Sketches & Applications

Chair
Dena Slothower
Stanford University

Contents

Chair
Dena Slothower
Stanford University

Program Coordinator
Vicki Caulfield
Sketches & Applications
Program Coordinator

Committee
Jonathan Gibbs
PDI/DreamWorks

Doug Roble
SIGGRAPH 2002
Sketches &
Applications Chair

Cheryl Stockton
Studio Firefly
Pratt Institute

Gary Templet
Sandia National
Laboratories

Jury
Simon Allardice
Lynda.com

Ed Angel
University of
New Mexico

Chris Bailey

Ronen Barzel
Pixar Animation Studios

Hisham Bizri
Center for Advanced
Visual Studies

David DeBry (grue)
Industrial Light + Magic

Randall Frank
Lawrence Livermore
National Laboratory

Darin Grant
Digital Domain

Marc Kessler
University of Michigan

Jacquelyn Martino
Philips Research, USA

Maureen Nappi
New York University

Kathy Neely
Fashion Institute of Technology

Cary Phillips
Industrial Light + Magic

Gary Pimentel
Evans & Sutherland

Introduction

"Draw your chair up close to the edge of a precipice and I'll tell you a story."
–F. Scott Fitzgerald

SIGGRAPH 2001 Sketches & Applications received close to 300 entries from 23 different countries, and over 130 companies, research facilities, educational institutions, and individuals. The final Sketches & Applications program featured 138 presentations, including emerging work from the technical, education, medical, arts, design, gaming, performing arts, and entertainment communities. Topics ranged from geology, character animation, and psychology to hardware acceleration, traffic patterns, puppetry, and beyond.

Technical sketches demonstrated the variety of disciplines and environments in which computer graphics research and production are developing. Artists and designers described virtual and augmented reality pieces from conception through production. These processes reveal the new and different ways we are choosing to perceive as well as the new ways we are choosing to communicate.

From the museum to the stage, the talks in the Art, Design, and Multimedia area offered fresh visions of the digital landscape and how interaction can occur within these invented spaces. Artists and designers described conception and production of virtual and augmented reality pieces, revealing the new and different ways we are choosing to perceive, while others showed us the new ways we are choosing to communicate.

Animation sketches explained the production process for everything from the cute and cuddly to the loud and scary, and revealed the inner workings of projects both large and small. As a group, the animation sketches demonstrated how confidently the artistry of the animator and the sophistication of the programmer are collaborating to present new and exciting visual experiences.

On behalf of the Sketches & Applications Committee, I would like to thank all of the contributors for giving us a glimpse behind the scenes. Numerous thanks also go to: the committee and jury for their incredibly hard work, SIGGRAPH 2002 committee members Simon Allardice and Doug Roble for pitching in, Lynn Pocock for her support, Tom Appoloni and Scott Senften for letting me follow them around and pick their brains, Vicki Caulfield and Carrie Ewert for making it a breeze to get everything together, and my colleagues in Freshman and Sophmore Programs at Stanford University for their patience and their willingness to give me the time to work on the Sketches & Applications program.

Dena Slothower
SIGGRAPH 2001 Sketches & Applications Chair
Stanford University

“2001: An MR-Space Odyssey:” Applying Mixed-Reality Technology to VFX in Filmmaking

Contact
Toshikazu Ohshima
Mixed Reality Systems
Laboratory Inc.
6-145 Hanasaki-cho,
Nishi-ku
Yokohama 220-0022 Japan
ohshima@mr-system.co.jp
www.mr-system.co.jp

Tsuyoshi Kuroki
Toshihiro Kobayashi
Hiroyuki Yamamoto
Hideyuki Tamura
Mixed Reality Systems
Laboratory Inc.

Introduction

Visual effects (VFX), which composite computer-generated imagery (CGI) onto real scenes in a feature film, usually require a sequence of images that is manually generated in the post-production process. In an alternative approach, mixed reality (MR) merges the real and virtual worlds to achieve real-time interaction between users and MR space[1,2]. In this sense, MR creates real-time VFX seen from an observer’s viewpoint. These two fields, which used to be considered independent, will rapidly affect one another. VFX techniques, especially photometric registration, are useful in MR systems. On the other hand, MR technology can be utilized in film production. This sketch introduces the first MR system that is specially implemented for filmmaking.

Advantages of MR in Filmmaking

The system uses the latest and highest level of MR technology. It calculates depth of objects that move around the real world in real-time so as to realize the dynamic mutual occlusion between the real and virtual objects. The MR system has the following advantages compared to virtual studio systems that are currently used in TV studios:

- No choroma-key technology. Blue or green backgrounds are no longer required. Computer-generated objects are placed not only in front of the real scene but also at any depth in the scene.

- Real-time composition of CGI and the real scene can be observed from an actor’s viewpoint with a head mounted display (HMD) and from a cinematographer’s viewpoint with a monitor.

These advantages can be applied to filmmaking in two ways:

1. In rehearsal, actors can view virtual characters in HMDs.

2. Depth data acquired in rehearsal can be used in the highly precise post-production process.

System Configuration

The system is based on a video-see-through MR system as shown in Figure 1.

- Using the Optotrak system, it tracks movements of a video camera and an HMD worn by the actor.

- Zooming is digitally encoded and transferred to a computer.

- Using a computer-vision method, a five-camera depth-detection system dynamically determines the depth of real objects in the scene.

“2001: An MR-Space Odyssey”

Using this system, we reproduced a few cuts taken from a short film produced by the film director Takashi Yamazaki, in which an actress interacts with a computer-generated creature in our MR studio as shown in Figure 2. Figure 3 shows a frame from the movie, and Figure 4 shows a real-time frame. The director and the audience see the panning and zooming operations of the camera in realtime. The system also allows the actress to change actions and positions as many times as necessary. Through her video-see-through HMD, the actress can see and fight with a virtual creature. This makes trial-and-error experimentation in the movie making much easier, and it allows audiences to see scenes seen from the actress’s viewpoint.

Conclusion

It may take a little while to practically apply this system to actual filmmaking. However, it may soon be utilized as a new type of entertainment in which a viewer can participate as an actor or actress.

References
1. Ohta, Y. & Tamura, H., Eds. (1999). *Mixed reality - Merging real and virtual worlds*, Ohm-sha & Springer-Verlag.
2. Feiner S., et al. (1999). Mixed reality: Where real and virtual worlds meet, in *SIGGRAPH 99 Conference Abstracts and Applications*, 156-158.

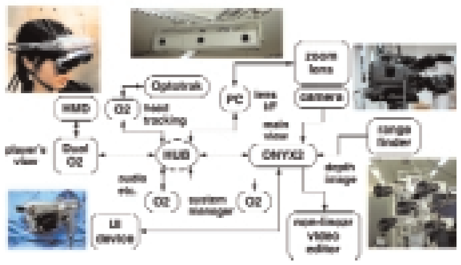


Figure 1. Concept illustration.

Figure 2. Tentative and final composites.

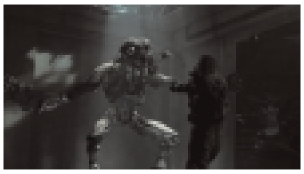


Figure 3. System configuration.

Figure 4.

2D Shape Interpolation Using a Hierarchical Approach

Contact
Henry Johan
Tomoyuki Nishita
University of Tokyo
henry@is.s.u-tokyo.ac.jp

Shape interpolation has been widely used for modeling and creating visual effects. Recent papers[1,2] have proved that interpolating two shapes taking into account their interiors can greatly improve the quality of the generated intermediate shapes. However, the algorithms are complex and hard to implement. Here, we present an easy and simple method to generate a smooth interpolation between two shapes. Given a vertex correspondence between the boundary of the two shapes, the compatible hierarchical representations of the shapes are constructed. Interpolating these compatible hierarchical representations generates the intermediate shapes. Unlike the previous approaches, our method can be used to interpolate polylines.

Construction of Compatible Hierarchical Representations

A hierarchical representation of a shape is defined as a set of triangles used to describe the shape with hierarchical relationships defined among the triangles. In the compatible hierarchical representations, each triangle in the source shape has exactly one corresponding triangle in the target shape. The outline of the compatible hierarchical representation construction is as follows:

First, we insert additional vertices into the two shapes: Figure 1(b). Note that we do not insert new vertices into short edges. Then we search for vertices to be removed. A vertex can be removed if its removal does not cause the two shapes to exhibit self-intersection and does not cause a folding problem. After that, we perform the vertex-removal operation and create triangles to represent the removed vertices. Finally, we create the source and the target shapes at the next lower hierarchical level: Figure 1(c). We repeat this operation until the shapes at the next lower level are either triangles or lines. We treat lines as degenerate triangles. We slightly change the algorithm in order to deal with open polylines. We do not remove the vertices at the end points of the polylines and do not perform the folding test.

Interpolation Between Two Compatible Hierarchical Representations

Our interpolation method proceeds as follows:

First, we interpolate the triangles at the lowest level of the representations: Figure 2(a). Then, we move to the next-higher level, interpolate all the triangles at this level, Figure 2(b), and finally we compute the final shape at this level: Figure 2(c). We perform this operation until we have finished processing the triangles at the highest level of the representations. To compute an intermediate triangle, we determine its ideal shape, its ideal orientation, and the ideal coordinates of its center. We define an ideal transformation between two triangles as a transformation that linearly changes the length of the edges.

Examples

Figure 3 shows four examples of interpolation between two shapes. In all of the examples, some part of the shapes undergoes rotations. Our method has produced smooth interpolation sequences for all these examples. There is no area deformation and local self-intersection in the intermediate shapes. The compatible hierarchical representations can be computed in less than 0.2 seconds. The interpolation sequences can be generated in real time. The computation is performed on a machine with Pentium III 800Mhz running Linux.

References
1. Shapira, M. & Rappoport, A. (1995). Shape interpolation using the star-skeleton representation. In *IEEE Computer Graphics and Application, 15*, 44-51.
2. Alexa, M., Cohen-Or, D., & Levin, D. (2000). As-rigid-as-possible shape interpolation. *Proceedings of SIGGRAPH 2000*, 157-164.

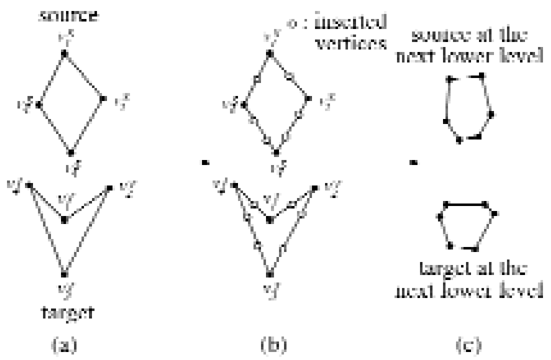

source
○ : inserted vertices
source at the next lower level
target
target at the next lower level
(a)
(b)
(c)

Figure 1. Constructing the compatible hierarchical representations of the source and the target shapes.

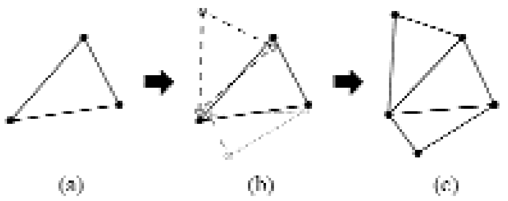


Figure 2. Generating the intermediate shapes.

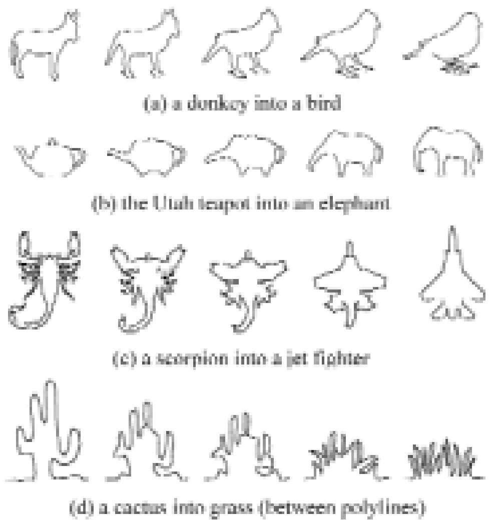

(a) a donkey into a bird
(b) the Utah teapot into an elephant
(c) a scorpion into a jet fighter
(d) a cactus into grass (between polylines)

Figure 3. Interpolation between two shapes.

3D Traffic Visualization in Real Time

Contact
Andre Gueziec
gueziec@computer.org
www.trianglesoftware.com

This sketch presents a system for generating a short narrated animation of highway traffic conditions in a large metropolitan area at a particular moment in time. The task is very challenging: an animated sequence ready for broadcast must be produced within minutes after collecting the most recent traffic data from sensors placed along the roads, police reports, etc.

An Animated Traffic Report vs. Live Footage

Our animation system incurs significantly lower costs than shooting live footage. Filming traffic from helicopters (which are not as safe as other aircraft) is expensive and may not cover the necessary territory with the required frequency. Fixed cameras provide limited coverage. Despite their popularity among TV audiences, broadcasting real-life accidents or traffic jams is more voyeuristic than informative.

Although this may seem counter-intuitive, live images can misrepresent actual traffic conditions. Traffic experts generally agree that:

- Traffic is significantly random.
- Probably as much as 15 minutes of continuous observation is required to gather reliable traffic flow information for a specific location. Drivers in stop-and-go traffic situations may attest that instantaneous snapshots of speed and density are useless in portraying vehicle flow.

Data-Driven Visualization in an Animation

Meaningful ways of presenting information on still images (paper) have been carefully studied. But for moving images, there is little published work on how best to visualize information. Further, some visualization scientists turned away from the moving image, arguing that information may be misrepresented in video or film.

At the other end of the spectrum, in artistic animation, animators are faced with a related problem: conveying "information" about what characters feel, think, do, or will do. The elements and tools they have developed to convey this information may have broader applicability. Some of those elements have been applied in this project: appeal, exaggeration and staging. For instance, we found that using realistic vehicle models, despite the effort and strain on the graphics system, provided more appeal to the resulting animations, and thus better served our data visualization goals (Figure 2).

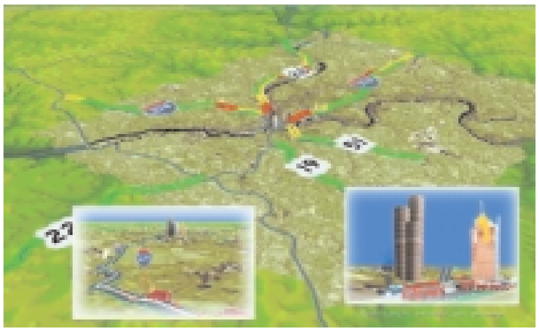


Figure 1. Virtual world used to produce animated road traffic reports for a particular metropolitan area. Colored segments overlaid on major highways indicate traffic conditions.

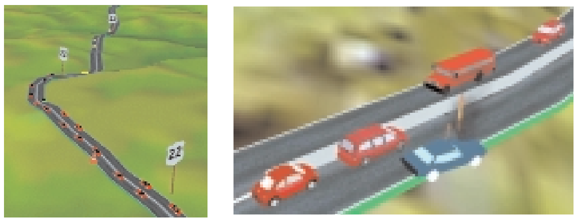


Figure 2. Using realistically modeled vehicles that are randomly selected from a pool (right) has much more appeal, and is more effective, than repeatedly using the same generic car (left).

Our System

Our system includes:
1. A modeling environment.
2. A rendering and simulation environment that is tightly coupled with modeling.
3. Interactive movie-making capabilities.

Built on MultiGen tools, our modeling is highly automated. It limits modeler input to artistic and design choices. We use geo-referenced spatial data, and we exaggerate selected elements, to produce minimal and cartoon-like polygonal modeling. The rendering uses Vega and Performer. For interactive movie directing, previewing, and generation of uncompressed video footage, we have engineered our system to render at 30 frames per second on a PC running Windows.

Some Technical Highlights

Vehicles. We opted for visualizing traffic flow using animated vehicles as an intuitive and, we believe, effective solution. We have built a physically based and data-directed vehicle traffic model.

Timing and exaggeration. We want to be able to survey a major highway section that is tens of kilometers long in about 10 seconds, by flying a virtual helicopter and observing animated traffic. This requires exaggerating the sizes and speeds of vehicles and simplifying the road network. (In traditional animation, exaggeration is a proven and effective technique.)

2D map. To transition smoothly from a 2D map to an animated virtual world, we overlay map-like colored symbols and textures on our virtual world. Placed on the scene graph with proper LODs, they are in turn visible, translucent, or invisible.

Dynamic scene. At run time, we regenerate the above polygonal symbols and textures depending upon current traffic data.

Acknowledgements

From MultiGen-Paradigm, we want to thank Tom Dowgiallo, Mark Ippolito, Nelson Iwai, Paul Levy, Binoy Marvar, Bill Preskar, Steve Thompson, Elizabeth Fox, and John Perser (for a great voice-over of the animation). Jeff Bott and Matt Barlow from Viewpoint have provided superb vehicle models, as well as the Three River Stadium Model. Thanks to Traffic.com. Thanks to Matthias Wloka.

Reference
1. Gueziec, A. Architecture of a system for producing animated traffic reports, April 2001.

ALVIN on the Web: Distributive Mission Rehearsal for a Deep Submersible Vehicle

Contact
Jan Jungclaus
Fraunhofer Center for Research
in Computer Graphics, Inc.
Providence, Rhode Island 02903
USA
jjungcla@crcg.edu

L. Miguel Encarnação
Robert J. Barton III
Petar Horvatic
Fraunhofer Center for Research
in Computer Graphics

Dudley Foster
Woods Hole
Oceanographic Institution

Introduction

The Woods Hole Oceanographic Institution (WHOI) operates the ALVIN Deep Submergence Vehicle (DSV) for scientific research at extreme ocean depths.[1] A key element that determines the length of such dives is the efficiency of the energy budgeting (getting the most out of the two batteries' overall capacity). Energy consumption is mainly determined by the use of outside lights, propulsion tasks, manipulator usage, and the use of experimental devices. Since the ALVIN submersible is a unique research tool that is completely reserved for years in advance, training has been conducted "on the job." To overcome the limited training opportunities, Fraunhofer CRCG constructed the ALVIN Web-based distributive simulator,[2] which will allow scientists to plan and rehearse their dives more efficiently.

Technical Requirements

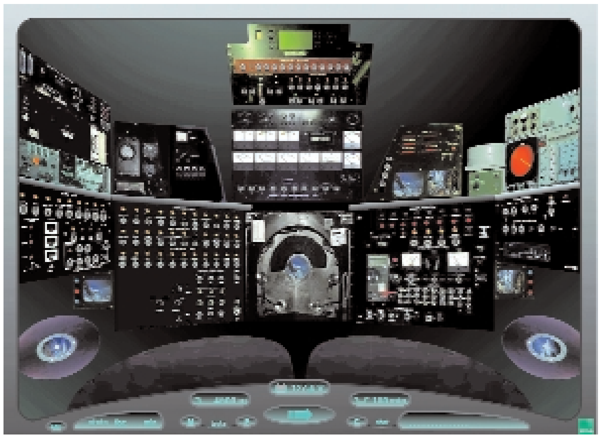
In order to implement the ALVIN simulator, Web-interface development had to be pushed into several directions at the same time. One of the main challenges was the requirement to fulfill both scientific and educational tasks. In order to meet this challenge, WHOI and Fraunhofer collaborated to implement:

- Full scientific applicability.
- Distributive operative collaboration.
- Electrical and logical evaluation of experimental and navigational operations.
- Web delivery with state-of-the-art performance.
- Visual feedback for ongoing processes.
- Analog and digital rendering of crucial variables.
- Photo-realistic visual representation.
- 3D look and navigational features.
- Intuitive interaction requiring only basic knowledge of the submarine's functional design.
- Display of impressions from the oceanographic surroundings of the submarine.
- Visual appeal for operational facilitation and to promote educational interest.
- A basic chat tool for supervision and counseling purposes.
- Maintainability of the interface and variables for future design changes and refined power modeling of the submersible.

Project History

Throughout the duration of the project, the discussion on which approach to take for the distributed ALVIN simulator and training interface was strongly influenced by rapidly changing and improving interface and communication technologies, on the one hand, and slowly maturing virtual reality technologies on the other. These technological developments, in turn, led to a variety of different conceptual designs that exemplify the different approaches to solve these problems from an interface-design point of view.

Initially conceptualized as a distributed collaborative virtual environment (cf. Figure 1, upper left), the immense hardware and software prerequisites for each of the participants prohibited continuation of such a high-realism (but low-accessibility) implementation. Consequently, and supported by professional design expertise and high-performance graphics software for the



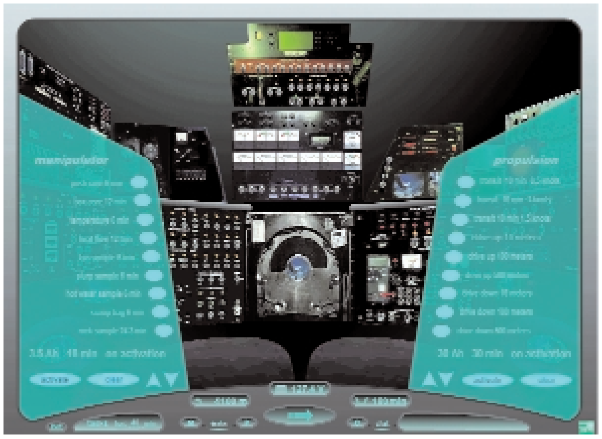


Figure 1. Virtual ALVIN mock-up (upper left) and final Web-based simulator interface in Flash.[3]

Web, a 3D-like interface was designed based on photographs taken from the original ALVIN submersible. The interface combines interactivity and visual appeal with a high degree of portability and maintainability (cf. Figure 1, center) and supports (employing Java servlet technology) all the requirements listed above.

References
1. Deep submergence vehicle ALVIN. (2000). Woods Hole Oceanographic Institution, URL: http://www.marine.whoi.edu/ships/alvin/alvin.htm
2. ALVIN simulator. (2001). Woods Hole Oceanographic Institution and Fraunhofer CRCG, Inc. URL: http://alvin.crcg.edu
3. Franklin, D. & Patton, B. (2001). *Flash 5! creative Web animation*, Macromedia Press, Berkeley, 2001.

Analysis and Simulation of Facial Movements in Elicited and Posed Expressions Using A High-Speed Camera

Contact
Tatsuo Yotsukura
ATR Media Integration & Communications Research Laboratories
Seikei University
2-2-2 Hikaridai, Seika-cho
Soraku-gun
Kyoto 6190288 Japan
yotsu@mic.atr.co.jp

Contributors
Hideko Uchida
San Francisco State Universtiy

Hiroshi Yamada
Nihon University

Nobuji Tetsutani

Shigeru Akamatsu
Hosei University/ATR Integration & Communications Research Laboratories

Shigeo Morishima
Seikei University

Introduction

The purpose of this research was to examine dynamic aspects of facial movements involving "posed" (intended) facial expressions versus "elicited" (unintended) emotional facial expressions. Participants were shown Gross & Levenson's set of standardized emotional film stimuli1, and their facial expressions were recorded by a high-speed video camera (250 frames/second), which allowed us to analyze facial movements very closely in image sequences. These movements cannot be seen with a regular video camera (30 frames/second). In addition, participants were asked to produce facial expressions of happiness, surprise, and disgust based on the Facial Action Coding System (FACS). The findings suggested that the patterns of facial movements in posed facial expressions and elicited emotional facial expressions are not significantly different, but that there are differences in the intensity of the facial expressiveness.

Method

Twenty-four participants (12 Japanese female, 12 Japanese male) were recorded by a high-speed video camera hidden behind a 21- inch prompter. The film stimuli were adapted from Gross & Levenson's set of standardized emotional film stimuli[1]. The protocol for posed expressions was utilized. We also used Ekman & Friesen's facial action coding system (FACS), an objective method for quantifying facial movements. FACS is an anatomically based coding scheme that codes facial muscular movements in terms of 44 action units (AUs) or action unit combinations (AU combinations). Participants were instructed to perform six basic facial emotional expressions based on combinations of AUs.

Analysis and Simulation of Facial Movements

Images of disgust, happiness, and surprise expressions were analyzed via the high-speed camera. We developed a feature-point tracking tool to analyze the facial movements of emotions. The images were downloaded onto a computer, and 28 dots were manually plotted on the face (four dots for an outline of the eyebrows, four dots for an outline of both eyes, five dots for an outline of the nose, six dots for an outline of the mouth, and one dot at a corner of the jaw) in the initial image. These 28 dots were selected because they were necessary to recreate the movements of basic facial expressions and to design computer graphics programs to simulate natural facial movements.

We examined patterns of facial movements by measuring the movements of the feature points, analyzed the feature points by the facial regions (eyebrows, eyes, and mouth), and computed the mean ratings of each region per emotion for all of the subjects.

Overall, our results indicated that the mean ratings of the eye region were higher than those of the eyebrows and mouth regardless of the emotional category or experimental conditions. Furthermore, we compared the total duration (from neutral to the start of the expression to attainment of peak intensity) between posed and elicited emotion conditions.

Our results demonstrated that the total duration was shorter for the posed expressions for all emotions. Figure 1 shows an example of the expression of "happiness." We also found differences in the magnitudes of facial movements between posed and elicited emotion conditions. The data showed that the magnitudes of posed expressions were greater than those of elicited emotions in all categories. In addition, the data revealed that the facial movements were nonlinear as a function of time.

We also simulated facial synthesis using Morishima's system.[2] Figure 2 shows a reconstructed facial movement of all frames. The generic facial animation was created through linear animation by morphing. The animation used by our technique is more natural than conventional generic animation.

References
1. Gross, J.J. & Levenson, R.W. (1995). Emotion elicitation using films. *Cognition and Emotion*, 9 89-108.
2. Morishima, S. (1996). Modeling of facial expression and emotion for human communication system. *Displays 17,* 15-25, Elsevier.

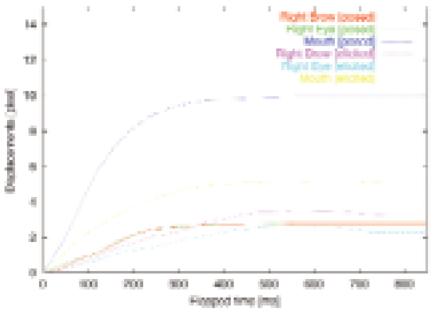


Figure 1. The average number of movements by facial part.

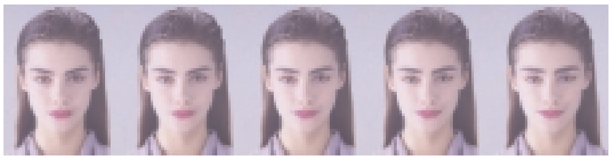
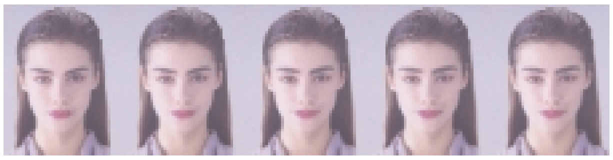


Figure 2. Reconstructed face (expression of "happiness").
(a) Linear animation (30 frames/second).



(b) Nonlinear animation (30 frames/second).

The Animator-Oriented Motion Generator, Animanium, Based on a Humanoid Robot-Control Algorithm

Contact
Fumio Sumi
Fujitsu SSL
sumi@ssl.fujitsu.co.jp

Hirotaka Imagawa
SEGA Corporation

Introduction
Currently, 3D computer graphics is the most popular method for making motion pictures, and it is a very powerful tool. But it is still immature compared to the tools of conventional celluloid animation. Some examples:

1. Three-dimensional computer graphics provides few methods for moving shapes. It offers many good modeling tools, but few shape model-handling tools.

2. Poor motion-generation techniques. Motion-capture systems and direct handling of the triple-axis rotation joints are only two methods that provide motion to hierarchical human structures such as hands or the head.

3. Poor operation of CG software tools. Existing CG software has too many functions, and it takes too much time to learn the operation. This is one of the reasons why it is difficult to train staff to use modeling and motion generation.

To solve the above problems (especially 2. and 3.), we developed Animanium, a new animation tool that uses a humanoid robot-control algorithm.

Implementation
1. Simplification of human-character pose adjustment. The human character is a very complicated link structure, and it has a large degree of freedom. It is difficult to manipulate a whole body at once by the ordinal inverse kinematics method. For this reason, we use the Jacobean matrix of inverse kinematics, which is used to control humanoid robots.

2. We used conventional celluloid animation, a kind of keyframe animation. The process of creating 2D celluloid animation is divided into two phases: making keyframes and creating interpolated frames. Animators make only keyframes, and computers generate interpolating frames.

System Overview
First, we developed the total motion-generation system based on humanoid robot-control algorithm shown in Figure 1.

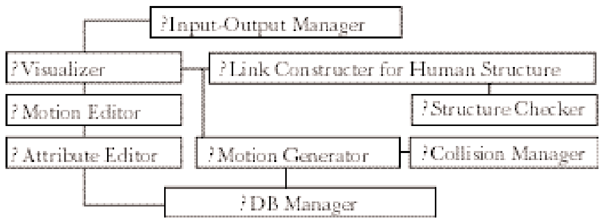


Figure 1. Functional Diagram.

The functions of this system are very useful for an experienced, professional animator. For a less experienced animator, additional steps are helpful functions for generating frames automatically.

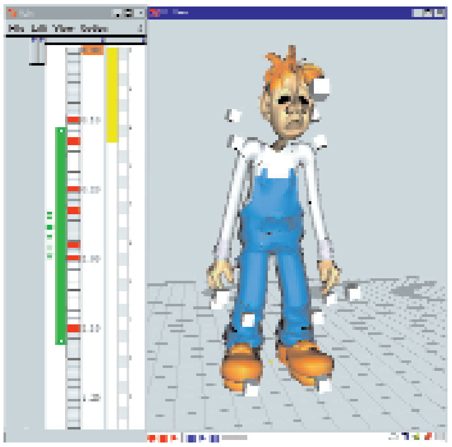
Technology
We developed a new interface method "Pin-and-Drag" for handling the entire body of a human character. This allows the user to drag a link to an arbitrary position with any number of links pinned in the global frame. Inverse and forward kinematics are used for generating and optimizing motion.

We also developed a character-motion adjustment method as a real-time operation, which can correct poses during playback. Using this technique, it is very easy to make create series of derivative motions.

Result
1. The time and number of steps required to interactively create poses for a human character is greatly decreased compared to other tools.
2. This tool is useful for both skilled and unskilled animators.

Acknowledgement
This work was supported by the Information Technology Promotion Agency, Japan.

References
1. Nakamura, Y. & Hanafusa, H. (1986). Inverse kinematics solutions with singularity robustness for robot manipulator control. *Journal of Dynamic Systems, Measurement and Control*.
2. Nagashima, F. & Nakamura, Y. (1992). Efficient computer scheme for the kinematics and inverse dynamics of a satellite- based manipulator. *Proceedings of IEEE International Conference on Robotics and Automation.*
3. Popovic, Z. (2000). Editing dynamic properties of captured human motion. *Proceedings of IEEE International Conference on Robotics and Automation*.
4. Yamane, K. & Nakamura, Y. (2000). Dynamics filter – concept and implementation of on-line motion generator for human figures. *Proceedings of IEEE International Conference on Robotics and Automation.*



AntiAliasing Perlin Noise

Contact
Ian Stephenson
National Centre for
Computer Animation
Bournemouth University
Poole, Dorset BH12 5BB
United Kingdom

Introduction

The Perlin Noise function is a key tool in procedural texturing, where it provides the controlled randomness required to create visual interest. However, as with all periodic functions, it is prone to aliasing when sampled at a frequency below the Nyquist frequency. The main approaches used to limit these artifacts are super-sampling, and frequency clamping, at the cost of render time, and shader complexity respectively.

The preferred solution to any aliasing problem is to convolve the continuous signal with a sampling kernel. If we accept a simple box filter, then this reduces to calculating the integral of the signal over the sampling area. However, the random nature of noise, and a lack of understanding of its implementation has led many shader writers to believe that this is not viable.

This sketch demonstrates that integration of noise is relatively simple, provided that lattice gradients are available.

The 1D Perlin Noise Function

Perlin Noise is a form of gradient lattice noise. A random gradient is generated for each integer lattice point (where the value of the noise is zero), and the gradients are smoothly interpolated between these values.

Interpolation is performed through the use of wavelets. These wavelets are defined to be zero outside the range ± 1, and integrate to zero over that range. When integrating 1D noise, it is therefore only necessary to consider the two wavelets that intersect the sample point. These wavelets are simple polynomials, so they can be trivially integrated. In forming the definite integral, these must be evaluated at each end of the sampling area. Hence, integrated noise is approximately twice as expensive as point sampling.

The 2D Case

In two dimensions, the wavelets are arranged in a grid. Once again, wavelets fully outside or fully inside the area being integrated over will sum to zero, and may be safely ignored. Those wavelets that intersect the corners of the area may be integrated in a similar fashion to the 1D case (at a total cost approximately four times that of point sampling). However, in the 2D case, we now need to consider those wavelets that intersect the edges of the area (or the faces of the volume in higher dimensional cases). Fortunately, the wavelet functions are of a form that allows integration of an entire edge to be evaluated as a simple summation of gradients, multiplied by a single polynomial. The computational cost of these edge integrals can, therefore, be kept to a minimum.

This approach generalizes to higher dimensions, and though the cost of the boundry evaluation increases with both dimension, and size of the summed area, it does so more slowly than super-sampling.

Implementation

The integrated noise (Inoise) functions have been implemented, both in a custom renderer and as DSO shadeops for Pixar's PRMan. A naïve implementation of the mathematics is numerically unstable, due to the mixing of floating-point operations with the division of space into lattice cells. However, a stable implementation has been developed and has been shown to accurately reproduce the standard noise function while greatly reducing aliasing artifacts.

Standard noise (left) vs. INoise (right).

The Attack on Pearl Harbor Battleship Row: Everything including the kitchen sink

David Frederic Horsley
Technical Director
Industrial Light + Magic

ILM was given a series of complex scenes for the bombing of Pearl Harbor. As you can imagine, there were numerous actions going on all at one time: explosions, smoke, debris flying through the air, people running every which way to action. In other words: chaos. We had our work cut out for us in trying to determine how to convey that feeling of pandemonium.

We started by going through our extensive library of effects developed for numerous productions that we've done over the years. We then began experimenting with different techniques, in varying combinations, layer upon layer, to obtain just the right look and feel for each sequence. In one instance, we used up to 30 effects elements to complete the final sequence. For example in the FA sequence, the following elements were used to create smoke:

- With the help of Effects Supervisor Ed Hirsh and his team, ILM created numerous practical smoke elements of different scales, under different wind conditions, of different velocities.

- In Baja California, director Michael Bay and his crew created numerous elements to add to our collection of practical smoke elements.

- To achieve correct perspective, computer graphics scientist John Anderson and his software team developed a fluid-dynamic simulation of large plumes of smoke based on our fluid dynamics software. These was rendered as particles.

- In addition to thick smoke plumes, we created lighter, airier plumes to fill the atmosphere around burning battleships.

- Then we added color and variation.

In all, we had 10 different simulations of light smoke and six different simulations of thick plumes.

Maya provided us with more tools to develop different simulations using turbulence plug-ins developed by our production software group. We were also able to emit particles from smoke textures using the same Maya software, to create layers of atmospheric smoke. This method of painting a texture gave us maximum control to create exact shapes.

We pasted smoke simulations on 3D planes that were in turn rendered as smoke elements, like moving images on sprites. Two dimensional compositing software to add smoke-like noise to elements to enhance practical smoke.

The Battleship Row bombing sequence involved the following list of elements and methods:

1. Maya particle simulation of smoke and fire.
2. Proprietary fluid dynamic simulation of gigantic plumes of smoke towering 600 feet into the air.
3. Crowds of particle-animated CG people, running, swimming, and fighting.
4. Accurate depiction of World War II battleships, destroyers, and cruisers.
5. Global illumination rendering of large complex models of battleships.
6. CG water using "Perfect Storm" simulation technology.
7. Global illumination rendering of Japanese planes.
8. Restoration of photographed background plates.
9. Compositing of explosions, fire, and smoke elements shot on stage.
10. Mental Ray and RenderMan for image rendering.
11. "Delayed read archive" RenderMan baked ribs for storage of battleship and airplane geometry and materials, easy access to geometry for efficient loading , and memory allocation.
12. Level of detail for the airplane animation rendering. This was combined with our optimized global illumination method.
12. Baking of ambient occlusion texture maps on our rigid body battleships.
13. Maya particle simulation of water spray and sparks.
14. Rigid-body dynamic simulation engines (proprietary software) for exploding airplanes and debris.
15. Simulated fire and smoke on water.
16. Fluid dynamic simulation torpedo wakes and explosion shock waves on water.
17. Instanced flying debris in air.
18. Splashing of water as debris hits the surface.
19. Boat wakes and bow splashes of moving ships.
20. Painted detailed maps of 10 different battle ships, with color, bump, opacity, rust, scorch, and area maps.
21. Two series of battleships, one before battle in peacetime and one series of destroyed battleships during the attack.
22. The kitchen sink.

Averaged Area Tables for Texture Filtering

Contact
Uwe Behrens
Digital Domain
300 Rose Avenue
Venice, California 90291 USA
ubehrens@d2.com

Attaining high-quality imagery requires that texture samples are properly filtered over the footprint of a pixel in order to reduce aliasing artifacts. Often, accuracy is traded for speed by using a specific kernel to prefilter the texture. At runtime, an approximation of the true filter response is calculated from one or more of the precalculated filter values.

Summed area tables (SAT)[1] are one such prefiltered representation of texture images. They allow for fast calculation of box-filtered texture integrals over axis-aligned rectangular regions. Unlike MIP maps,[2] they do not require the filter region to be square, which reduces blur for filter regions with unequal width and height.

Let T be a 2D texture of integral size $(T_w; T_h)$, and let $T(x, y)$ be the value of T at location (x, y); $x \in [1; T_w]$; $y \in [1; T_h]$. Let r be a convex region, completely within the bounds of T, for which a filtered texture value must be computed. A simple method to approximate the filtered value of T over r uses the smallest, axis-aligned bounding rectangle $R = (x_l; y_l; x_h; y_h)$ of r, and computes the average, or *box filtered value* $\mathbf{B}_T(R)$, of T over all points in R:

$$\mathbf{B}_T(R) = \frac{1}{\|R\|} \sum_{i=x_l}^{x_h} \sum_{j=y_l}^{y_h} T(i,j)$$

with $\|R\| = (x_h - x_l + 1)(y_h - y_l + 1)$ being the size of R in texture space. $\mathbf{B}_T(R)$ usually gives an acceptable approximation to the integral of T over r in terms of quality, although by no means a perfect result. To calculate $\mathbf{B}_T(R)$ quickly, we precalculate the SAT of T, S_T as a texture of the same size as T with:

$$S_T(x,y) = \sum_{i=1}^{x} \sum_{j=1}^{y} T(i,j)$$

which is the sum of all values in the rectangle $(1; 1; x; y)$. Now, $\mathbf{B}_T(R)$ is simply (with $S_T(x, y) = 0$, if $x \notin [1; T_w] \vee y \notin [1; T_h]$):

$$\begin{aligned}\mathbf{B}_T(R) = {} & [S_T(x_h, y_h) - S_T(x_h, y_l - 1) - \\ & S_T(x_l - 1, y_h) + S_T(x_l - 1, y_l - 1)] / \|R\|\end{aligned}$$

The major problem with summed area tables is that the values to be stored in $S_T(x, y)$ can become arbitrarily large (cf. Figure 1(b)), so that a SAT for an eight-bit texture must often be stored in floating-point format, requiring 32 or more bits per pixel and time-consuming floating-point operations. This is unacceptable in production environments, where texture sizes of 4,096 x 4,096 pixels are not uncommon. To overcome this, we developed the *averaged area table* (AAT). The AAT of T, A_T is defined as a texture of the same size as T, but with:

$$A_T(x,y) = \frac{S_T(x,y)}{xy} = \frac{1}{xy} \sum_{i=1}^{x} \sum_{j=1}^{y} T(i,j)$$

storing the average, rather than the sum of the values in the rectangle $(1; 1; x, y)$. Clearly, $\forall x; y : A_T(x, y) <= \max(T(x, y))$, hence A_T can be stored in the same amount of memory, and with the same precision as T. Rounding errors on the order of $1/\max(T(x, y))$ will be introduced, but since R is already an approximation of the real filter support r, in most cases the additional error is negligible. In the example figure, only one entry (marked as *) was rounded from the true value 62.5 to 63. Calculating $\mathbf{B}_T$ is only slightly more complicated from the AAT than from the SAT[1]:

$$\begin{aligned}\mathbf{B}_T(R) = {} & [x_h y_h A_T(x_h, y_h) - \\ & x_h(y_l - 1) A_T(x_h, y_l - 1) - \\ & (x_l - 1) y_h A_T(x_l - 1, y_h) + \\ & (x_l - 1)(y_l - 1) A_T(x_l - 1, y_l - 1)] / \|R\|\end{aligned}$$

The AAT allows efficient calculation of the filtered value of a rectangular region in texture space. Unlike the SAT, the AAT does not require any additional memory, surmounting even the 33-percent memory overhead of MIP maps. Also, the AAT, like the SAT, avoids the strong blur of MIP maps, which is caused by the limitation to square filter regions. In addition, the SAT/AAT requires only four texture lookups per computation, as compared to eight lookups needed for typical trilinear MIP map interpolation.

Figure 1 (a) shows a simple 6 x 6 checkerboard texture, and its corresponding SAT (b), and AAT (c).

References
1. Crow, F.C. (1984). Summed-area tables for texture mapping. In *Proceedings of SIGGRAPH '84, 18* (3), 207-212.
2. Williams, L. (1983). Pyramidal parametrics. In *Proceedings of SIGGRAPH '83, Computer Graphics 17* (3), 1-11.

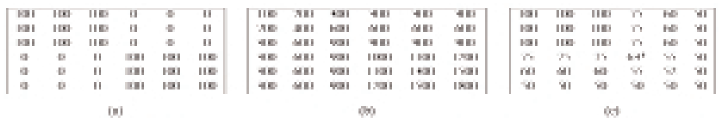


Figure 1. From texture (a) to summed area table (b) to averaged area able (c).

Bringing PhotoRealism to Fantasy: The Hybrid Approach of a Digital Matte Artist

Paul Huston
Senior Digital Matte Artist
Industrial Light + Magic

The discipline of matte painting has survived, and thrived, in the digital age because filmmakers still need ways to expand the scope of their films without proportionately expanding their budgets. Matte artists are called upon to create images that are grand, glorious, and fantastic, and at the same time completely convincing. These demands have caused matte artists to develop techniques that combine the strengths of various approaches in order to deliver highly complex photorealistic scenes within ever-shrinking production schedules.

This overview of these techniques follows the development of digital matte painting from the early 1990s.

2D Image Editing

With the advent of 2D image-editing software, it became possible to shortcut many of the tedious methods of traditional matte painting. This is done by sampling colors directly from scanned film using cut and paste, levels controls, alpha channels, direct painting, and other approaches to manipulate and combine scanned images for creating and blending additions to pre-photographed scenes. These additions are combined with live-action footage using post-production compositing systems, a direct transition from the traditional method of using photo-reference transferred onto panels using projectors, painting by hand and brush, and compositing with photo-chemical processes.

Early Examples: "Baby's Day Out" and the Young Indiana Jones Adventures

Location and Miniature Photography
The ease of using photographs as a base for creating paintings led to use of location photography and miniature photography. Before and after examples show how miniatures and location photographs were combined to create images for "101 Dalmatians" and "Star Wars Special Edition." One "Star Wars Special Edition" shot is shown as storyboard, concept painting, rough computer model, miniature, live-action elements and final to illustrate the planning process and progressive refinement of the image.

2.5D
The combination of photography-based paintings, simple geometry, and 3D animation led to what was known colloquially as 2.5D and more technically as image-based rendering, which was used in shots for "Mission Impossible" and "Star Wars Special Edition." When the illusion of 3D space created by camera movement and simple image planes placed in depth in the 3D environment was merged with the visual subtleties inherent in photographs, it produced an impressive combination of great realism and low set-up and render times. The approach has its precedents in "multi-plane" camera set-ups developed in earlier matte painting photography and in films using cell animation such as Disney's "Pinnochio" and others. This technique was used to good advantage in creating backgrounds for the Pod race in "Star Wars: The Phantom Menace." It was an extremely efficient technique given the unique conditions for that sequence, which featured point-of-view shots of extremely fast-moving vehicles through an entirely imaginary landscape.

3D CG and Image-Based Rendering
Many matte shots require extensive architecture, and in some cases computer models and shading become the solution of choice. Mechanical and architectural subjects can be rendered synthetically and then animated with image-based techniques (using painting to create complex detail and atmospheric effects). Image-based rendering was used to create geometry and shading as a basis for a painting in shots showing Mos Eiseley for "Star Wars Special Edition" and the princesses palace in "Star Wars: The Phantom Menace."

Conclusion: "Space Cowboys" HM040

As shots become more complex, variations and combinations of techniques are combined to create the desired effect. The closing shot for "Space Cowboys" combines 3D Phong-shaded imagery, image-based rendering of moon surfaces based on NASA photos, image maps created using height fields and shading, and image hulls created from painted depth maps mapped with live-action photography to bring the viewer from moon orbit down into the face-plate of astronaut Hawk.

“Budweiser Alien” Commercial

Leandro Estebecorena
Lead Technical Director
Industrial Light + Magic

Jonathan C. Lyons
Animator
Industrial Light + Magic

During October 2000, Budweiser asked the Computer Graphic Commercials department of Industrial Light + Magic to produce a new commercial. Several different cutting-edge and traditional techniques were combined to achieve the final images of a huge, totally CG stadium filled with aliens.

The director, Rick Schulze, wanted a sea below the stadium, and various techniques involving displacement maps and particles in Maya were used for this. He also wanted rocks surrounding the view to the ocean, and a fractal system was used to generate CG rocks at the bottom of the stadium. Caustics were implemented to light those rocks with a simulated reflection of the water.

According to the artwork provided by art director Randy Gaul, the stadium was supposed to be filled with a blue fog. Shaders were written to simulate the mist, and CG rigs were set up to support practical smoke elements on top of the stadium torches. Also, volumetric light-and-shadow tools were developed in order to integrate the aliens into the misty stadium. The stadium had to be crowded with aliens, and a special pipeline involving particles and different cycles in three different levels of resolution was set up by technical director Doug Sutton to solve this problem.

The biggest challenge for Carlos Huante, ILM art director and designer of the aliens, was to invent a creature that would be scary at the beginning of the spot but likeable by the end. Reference material included a small maquette, which is seldom done for commercial work. The maquette was quite valuable to ILM modeller Izzy Acar, who modeled the alien using Softimage and ILM proprietary sculpting tools.

Primary animation controls were built by CG commericals artist Todd Krish, using Softimage’s inverse kinematics (IK). The alien’s legs had three joints, so standard IK solving had a snakelike action. A second IK chain of two joints was added to control the bending, and this provided a more natural motion.

The alien’s head design included six “wattles” hanging below the chin. Initially these were to be animated using hair simulation, but the alien’s anatomy proved uncooperative, so joint chains were added and keyframed by hand. In pre-production, Paul Griffin, animation director for the show, was also developing animatics for the main establishing shots and setting up a battery of shapes that would later help to implement better alien performance.

Animators gathered for two meetings prior to beginning animation. The first was around a conference table to generate ideas for the behavior of the creatures. The animation was also expected to emphasize the creatures’ transition from frightening to friendly. The second meeting was a videotape session using props and interactive lighting, in which animators took turns acting the scenes From these tapes, the director chose takes that were edited into a rough cut of the spot for reference. The reference also included shots of the “Wassup” actors from the original spot.

After extensive pre-production and development, the remaining production time was a scant three weeks. The production team decided to bring on additional animators to maintain quality. Each of the seven animators was asked to animate six different cycles for the alien crowd in the stadium in one day, three of quiet action and three of broad cheering that would be used at the climax. The clients were extremely pleased with the animation in this production.

Using ILM proprietary tools, paint artists Richard Moore and Rebecca Heskes generated the maps and textures for the alien skin, and added details for each shot’s camera. Painter Drew Klausner was assigned the task of painting all the details in the spaceship, while technical director Dean Foster was responsible for painting all the architectural details in the stadium (modeled by Larry Tan).

We started to receive plates by the beginning of December 2000. The plates from the set (shot under the supervision of VFX supervisor Kevin Rafferty) were promptly match-moved by Luke Longin and Ingrid Overgard using Softimage and ILM proprietary software.

Once production started, the 12 CGC technical directors assigned to the show faced, and solved, a host of problems to light the crowded stadium and render those believable aliens (for example: volumetric shadow generation, defining the look of the eyes in the middle of production, changes in choreography, modification of caustic light patterns, etc.). To explore different possibilities for the appearance of the alien’s eyes, an interactive setup in our compositing tools was developed in the middle of production.

Scripts were set up by assistant TD Michael Muir to generate proof sheets for color continuity evaluation among all the spot’s shots on a daily basis. Additional problems (for example, interaction between live elements and CG elements in the “beheading” shot) were solved using Sabre (an ILM high-speed compositing system that incorporates Discreet’s inferno-flame software).

The spot started production in December 2000 and required only three weeks of production. It was completed on 16 January, just in time for the Superbowl telecast, 28 January 2001.

Calibration-Free, Easy 3D Modeling from Turntable Image Sequences

Contact
Sumit Gupta
National University of Singapore
10 Kent Ridge Crescent,
Singapore 119260
eleks@nus.edu.sg

Kuntal Sengupta
National University of Singapore

Introduction

Image-based 3D modeling techniques are gaining popularity in computer graphics and virtual reality applications. However, most existing techniques require precise camera calibration, which restricts their application to special environments and can only be performed by experts. We propose a new 3D modeling technique using silhouette-based volume intersection with the following features:

1. The method works without the need of camera calibration.
2. It uses a simple and effective intersection test.
3. It is computationally very efficient because it reduces the number of calculations required to compute the projection from 3D to 2D and requires no additional hardware.
4. Because it is silhouette-based, there is no need to compute dense point matches across views.

Using our technique, designers and modelers who are not experts in the field of computer vision can easily acquire 3D models from multiple views of an object undergoing rotational motion.

Theoretical foundations

In silhouette-based volume intersection techniques, multiple views of an object are captured by a camera, and then silhouettes are extracted for each view. If the positions of the cameras are known, we can estimate the conic volumes that encompass the objects. The volume model that best represents each object is computed by intersecting all these cones. Accurate calibration of the cameras, which is required by all volume-intersection techniques, is a specialized task. Also, each time the setup is changed, the arduous calibration task must be repeated. Hence, these techniques have not gained much popularity outside the computer vision community.

Our contribution to this volume-intersection technique has been in the development of the special calibration matrix P of the form shown below, after an extensive analytical and experimental study:

$$P = \begin{bmatrix} c & 0 & u_0 & cu_0 \\ 0 & c & v_0 & cv_0 \\ 0 & 0 & 1 & c \end{bmatrix}$$

The parameters c and v0 only influence the relative size and the position of the estimated 3D model, respectively. Hence, they can be chosen arbitrarily. Also, u0 is determined from the images directly. Hence, there is no need to perform camera calibration. Our method has been tested successfully on a wide range of objects such as shoes, video cameras, torsos, cups, toys, etc. Some of the estimated models are shown in Figures 1 and 2.

See also: www.ece.nus.edu.sg/stfpage/eleks/silo.htm

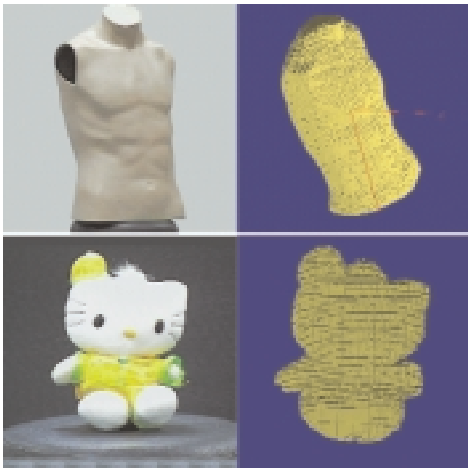


Figure 1. Original objects and the estimated models of a torso and a toy.

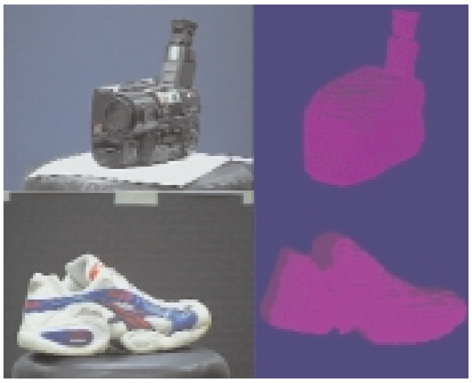


Figure 2. Original objects and the estimated models of a video camera and a shoe.

Change Blindness with Varying Rendering Fidelity: Looking but Not Seeing

Contact
Kirsten Cater
University of Bristol
cater@cs.bris.ac.uk

Alan Chalmers
Colin Dalton
University of Bristol

A major challenge in virtual reality is to achieve realistic images at interactive rates. However, the computation time required for realistic image synthesis is significant, which precludes such realism in real time. One possible way of producing perceptually high-fidelity images in reasonable times is to exploit the "flaws" in the human eye, for although the human eye is good, it isn't perfect! This sketch shows how the concept of "change blindness" may be exploited to produce realistic images of complex scenes in real time for use on an Explorer/1 motion platform.

Change blindness is the inability of the human eye to detect what should be obvious changes in a scene. Humans can miss large changes in their fields of view when they occur simultaneously with brief visual disruptions, such as an eye saccade, a flicker, or a blink. This concept has long been used by stunt-doubles in film.

We have used the onset of a blank field, mudsplashes, or masking blocks each time an image is changed to create the visual disruption. This swamps the user's local motion signals caused by a change, short-circuiting the automatic system that normally draws attention to its location. Without automatic control, attention is controlled entirely by slower, higher-level mechanisms in the visual system, which search the scene, object by object, until attention finally focuses on the object that is changing. Once attention has latched onto the appropriate object, the change is easy to see, but this occurs only after exhaustive serial inspection of the scene.

The experiment involved 45 images rendered in Radiance.[4] We asked a group of six "judges" to give a short description of the scenes. Their descriptions enabled us to define, for each scene, several aspects that are termed "Central Interest" aspects:[2] those aspects that were mentioned by at least three of the judges. Central interest aspects tended to concern what one would be tempted to call the main theme of the scene. Similarly, we noted several aspects that are termed "Marginal Interest" aspects:[2] those aspects that were mentioned by none of the judges. In the experiment, by manipulating changes in Central or Marginal Interest aspects, we controlled the degree of attention that our subjects would be expected to pay to the changes.

Figure 1 shows an example of the experiments we have conducted with 20 subjects. Figure 1(a) shows a scene rendered in Radiance to a high level of realism, while 1(c) is the same scene rendered with less fidelity for some objects and at significantly reduced computational cost. Figure 1(b) shows the same image as 1(c), but with the mudsplashes.[3] Each original image is displayed under strictly controlled lighting conditions for approximately 400 milliseconds, followed by the mud-splashed image for 60 milliseconds, and finally the modified image for a further 400 milliseconds. This sequence is continually repeated until the user has spotted the change. If no change was observed after 60 seconds, the experiment was stopped. Similar experiments were carried out with the flicker and masking-blocks paradigms.

The results showed that a significant amount of time was required for the user to spot the differences in the images. Future work will involve developing these techniques to dynamic scenes for eventual incorporation on the motion platform.

References
1. May, J. (2000). Perceptual principles and computer graphics. *Computer Graphics Forum, 19* (4).
2. O'Regan, J.K., Deubel, H., Clark, J.J., & Rensink, R.A. (1999). Picture changes during blinks: Looking without seeing and seeing without looking. *Visual Cognition*.
3. O'Regan, J.K., Rensink, R.A., & Clark J.J. (1999). Change blindness as a result of mudsplashes. *Nature*, 398.
4. Ward Larson, G.(1998). *Rendering with RADIANCE: The art and science of lighting simulation*. San Francisco: Morgan Kauffman.

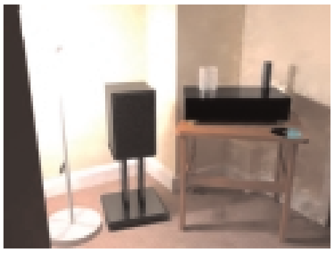
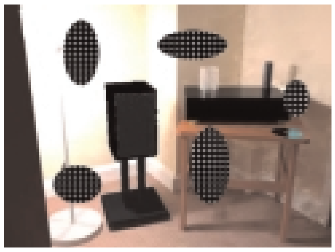
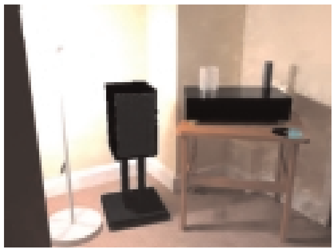


Figure 1. (a) High-quality image;



(b) low-quality imagewith mud-splashes;



(c) low-quality image.

Character Setup and Animation for "3-2-1 Penguins!"

Contact
Michael B. Comet
Big Idea Productions
206 Yorktown Center
Lombard, Illinois 60187 USA
michael.comet@bigidea.com

This animation sketch presents how the Penguins animation team at Big Idea Productions dealt with cartoon-style rigging and animation issues for their new 3D series: "3-2-1 Penguins!" The video blends the traditional quality of animated films, such as exaggeration and cartoon-style action, with a 3D universe.

Technology

All characters for "3-2-1 Penguins!" were rigged with typical inverse kinematics (IK) setups that included blending for FK and IK arms. In addition, custom controls were written to allow squash and stretch for the limbs and body. Custom MEL scripts were developed to create an easy user interface for the animators that included the ability to select and animate one part of the character while different controls displayed keyframe information on the timeline.

Characters were also set up with a custom "light rig" that illuminated each character individually. A custom light user interface allowed animators to quickly and easily adjust the lighting for all characters at once or to tweak lights independently. The rim light for each character was controlled by a custom plug-in. The plug-in automatically adjusted the rim-lights position and intensity based on the camera and character positions. The characters were also shaded with a variation of the default shader to yield a more traditionally animated appearance. The shader itself integrated into the lighting user interface and controls.

Other utilities included tools to automatically adjust spline curve interpolation to remove unwanted peaks, tools to automatically allow characters to grab and let go of props, and a way to save and reload vertex skin weighting among models with different vertex counts.

Style

In addition to technical hurdles, the animation team faced the challenge of creating an exaggerated style of animation on a tight deadline. The Penguins team departed from the traditional department approach used in most larger studios and used a smaller "do-it-all" artist workflow and "team-approach" process. The animation process was researched and developed to include a rough "pop-through" stage, which allowed the animators and director to review the blocking quickly and in a way that was geared toward snappy cartoon animation.

Penguins Core Animation Team

Producer
Jon Gadsby

Director
Ron Smith

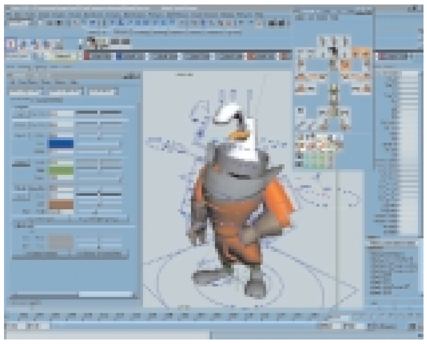
Artists
Mark Behm
Michael Comet
Jeff Croke
Rob Dollase
Everett Downing
Aaron Hartline
Bill Jacoby
Keith Lango
Tim Lannon
Michael Laubach
Joe Shaw

Production Assistants
Katie Carnes
Melody Chesney
Amanda Fragale

Production Coordinator
Matt Garbera

Jason and Michelle, stranded at Granmum's for the summer, explore the attic.

Zidgel, Midgel, Fidgel, and Kevin, the wackiest space penguins ever!


The character rig for Zidgel, showing control curves, the character UI, and the lighting UI.

Collisions and Adaptive Levels of Detail

Contact
John Dingliana
Trinity College Dublin
John.Dingliana@cs.tcd.ie

Carol O'Sullivan
Gareth Bradshaw
Trinity College Dublin

Collision detection, contact modeling, and collision response are vital but inherently expensive features of a physically based animation system. As scene complexity increases, collision handling quickly becomes a major bottleneck in the simulation process. A trade-off between speed and accuracy is often required in order to achieve interactive frame-rates[1]. Our goal in ReACT (Real-time Adaptive Collision Toolkit) is to optimize this trade-off by making simplifications as invisible as possible to the viewer.

Many applications simply address the frame-rate problem through pre-emptive simplification, reducing the complexity of the simulation to a pre-determined "safe" level. However, when complexity changes often over the course of a simulation, such an approach suffers from one of two problems: over simplification of the whole for the sake of relatively few snapshots of high complexity, or a drop in frame rate when the computational workload has been underestimated. We can avoid this through adaptive and interactive simplification of the simulation as it evolves. A popular approach to reducing workload in interactive animation is visibility-based culling, where parts of the scene not inside the visible volume are excluded from normal processing[2]. This is taken a step further in ReACT by applying varying levels of simplification over different regions within the viewable area. Such steps are an improvement, but they do not implicitly guarantee target frame rates if static rules are used to determine the levels of detail for different regions. If, for instance, the higher-priority regions should ever encounter computationally complex situations themselves, then problems similar to those in the pre-emptive simplification approach arise.

Target frame rates can be guaranteed by using time-critical mechanisms for computationally expensive parts of the simulation process. A time-critical (or interruptible) mechanism is halted when a scheduler decides that enough time has been spent on any particular task. For such an approach to work, however, we must ensure that some result is obtained and that some degree of correctness is maintained in the system, regardless of when processing is interrupted. This is achieved by using incremental mechanisms, which generate results of increasing accuracy as more time is spent on processing[3]. One example of this is Hubbard's sphere-tree collision detection system, which is extended in ReACT to return increasingly accurate approximations of contact data for collision response calculations (see Figure 1).

Although an interruptible system may guarantee target frame rates, it does not in itself ensure that the trade-off between accuracy and processing time is optimized. For this, we must incorporate some form of prioritization within the process. Certain events or specific parts of the scene are categorized as being more important and, as a result, given more processing time. Prioritization of the scene is based upon factors related to visibility and perceptibility of approximations within different parts of the scene. In ReACT, for instance, priority can be based on a weighted combination of factors such as eccentricity, occlusion, or projected distance from the user's fixation point determined interactively with the use of an eye-tracker (Figure 2).

References
1. Hubbard, P. M. (1995). Real-time collision detection and time-critical computing. *In Proceedings of the First ACM Workshop on Simulation and Interaction in Virtual Environments,* July 1995, 92-96.
2. Chenny, S. & Forsyth, D. (1997). View-dependent culling of dynamic systems in virtual environments. In *Proceedings 1997 Symposium on Interactive 3D Graphics,* 55-58.
3. Dingliana, J. & O'Sullivan, C. (2000). Graceful degradation of collision handling in physically based animation. In *Computer Graphics Forum,* 19 (3), 239-247.

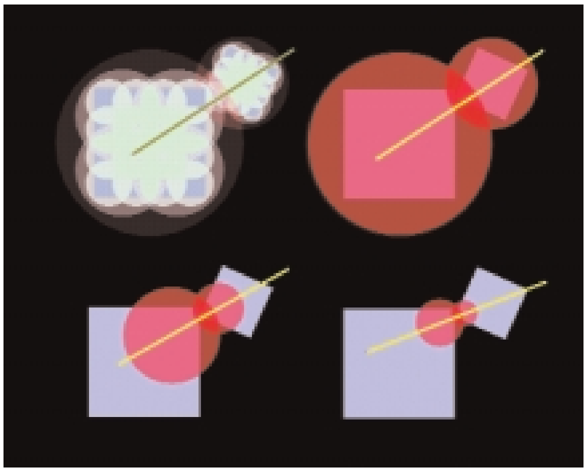


Figure 1. A sphere-tree-based collision-processing system. Shown are the full sphere trees of two objects and the process halted at three different stages. Better approximations of the collision points directly affect the computed response as seen by the direction of the impulse vector in yellow. Red spheres are colliding nodes.

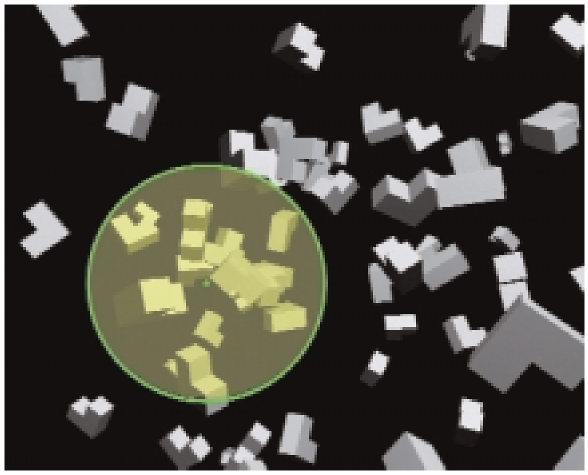


Figure 2. The screenshot shows a projected region on the screen around the user's fixation point, which is given a higher priority, and objects within the region are processed at a higher level of detail.

Colored-Paper Mosaic Rendering

Contact
Sanghyun Seo
Chung-Ang University
ddipdduk@cglab.cse.cau.ac.kr

Youngsup Park
Kyunghyun Yoon
Chung-Ang University

Sungye Kim
Electronics and
Telecommunications
Research Institute

Various features must be considered to produce mosaic effects in colored paper, including paper texture, how the paper is attached by hand, the irregular lines of torn paper, and a white section that shows a cutting plane (see Figure 1).

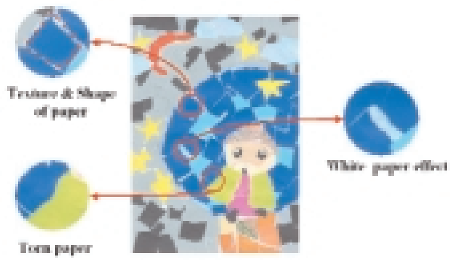


Figure 1. Features of the mosaic work.

Paper Texture and Shape of Attached Paper

Perlin's noise function is used to get the texture of the paper. This function utilizes an image-coordinate value and the noise function value at each coordinate to create the normal vector. Light vector is used on a random spot of the surface to generate the texture. The Voronoi diagram is used to generate the shape of an attached paper. The input image is divided into even grid sizes to generate the Voronoi polygon. Random points are generated in each divided area. After creating a Voronoi diagram using these input points, the Voronoi polygons are passed to the next stage.

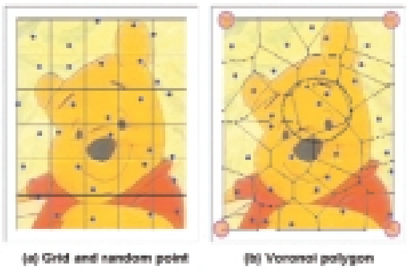


Figure 2. Determination of attached paper shape.

Torn Paper

The random mid-point displacement algorithm is used to generate a more natural torn-paper effect. This method continually adds a random height to the center point between the two points that form the edge of the paper, which is repeatedly applied to generate the resulting random line (Figure 3a).

White Paper Effect

A piece of colored paper consists of many different layers of fiber. Therefore, when torn, it exhibits a rough white surface. For the purposes of this sketch, we assume that a piece of colored paper consists of two different layers. The top layer is the colored layer, and the bottom layer is the white paper. The sizes and shapes of each paper layer (using the random midpoint displacement algorithm) are different. As random pieces of paper are applied, eventually the white layer is larger than the colored layer, which is the method used to express the white-paper effect of the torn paper (Figure 3b).

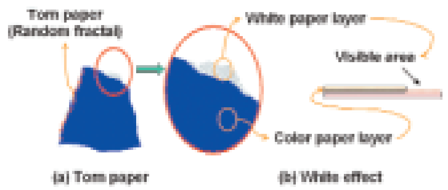


Figure 3. Irregular torn paper and white effect.

Conclusion and Future Work

The colored-paper mosaic-rendering algorithm proposed in this sketch enables creation of a hand-made mosaic work by using texture as well as the torn-paper effect. The resulting mosaic image is more natural than any other image generated by conventional software. However, the colored-paper mosaic effect uses polygons of the same scale, which reduces thickness and spatiality of the mosaic work. In future work, an adaptive-space (image region) subdivision technique that considers features of input image should be considered to resolve the problem.
See also: cglab.cse.cau.ac.kr/npr

References
1. Non-photorealistic rendering. In SIGGRAPH 99 Course #17, 1999.
2. Barnsley, M.F. (1993). *Fractals Everywhere*, 2nd Edition, AP Professional.
3. de Berg, M., Kerveld, M. V., Overmars, M., & Schwarzkopf, O. (1997). *Computational Geometry Algorithms and Applications*, Springer.

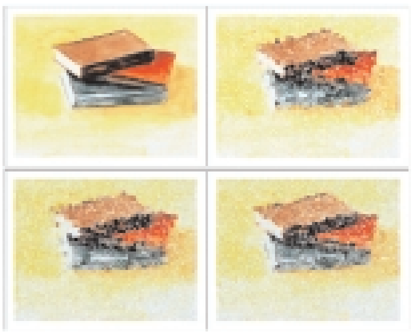


Figure 4. Images of colored-paper mosaic rendering.

ComicDiary: Representing Individual Experiences in Comics Style

Contact
Ryuuki Sakamoto
ATR Media Integration & Communications Research Laboratories
Japan Advanced Institute of Science and Technology
skmt@jaist.ac.jp

Yasuyuki Sumi
CREST, Japan Science and Technology Corporation
ATR Media Intergration & Communications Research Laboratories

Keiko Nakao
Kenji Mase
ATR Media Integration & Communications Research Laboratories

This sketch describes ComicDiary, a system that automatically creates a personal diary in a comics style. ComicDiary is built as a sub-system of our ongoing project (C-MAP) to develop a personal guidance system for touring museums, trade shows, academic conferences, cities, etc.[1] The aim of the C-MAP system is to provide users with personalized guidance in temporal and spatial situations, as well as for individual interests. We intend that ComicDiary will be used as a casual tool for augmenting each individual user's memory and encouraging users to exchange personal memories.

ComicDiary: What and Why

The goal of ComicDiary is to allegorize individual experiences by creating comics from users' touring records. Exhibitions are visited by people of all generations. A comics-style representation of personal diaries drawn with amiable expressions[2] is appropriate for such places. The comic, composed of 12 frames, is automatically generated as a diary. Users can view their diaries at information kiosks. They can also choose to print the information at the exit of the exhibition and/or access the information as an online service.

The figure is an example of a comic created by ComicDiary. It shows a user's personal diary during a Japanese academic conference. The comic describes where the conference was held, which presentation was the most interesting, what events the user attended, and so on.

The advantages of using comics for representation of diaries are as follows:

- Comics are a casual medium for exchanging personal experiences among users as well as for personal use.
- Comics make it easy to grasp the entire structure of sequential events.[3]
- Comics enable the use of powerful expressions for emphasizing important episodes.

The Technique

The user's activities and interest data (generated by a PalmGuide device, information kiosks, and an infrared badge) are stored in the central user database as the user explores an exhibition site equipped with the C-MAP system. The central database also includes an exhibition database, which stores exhibition data such as abstracts of exhibits and timetables of specific events. ComicDiary exploits these stored databases as story sources. ComicDiary generates a comic in three phases:

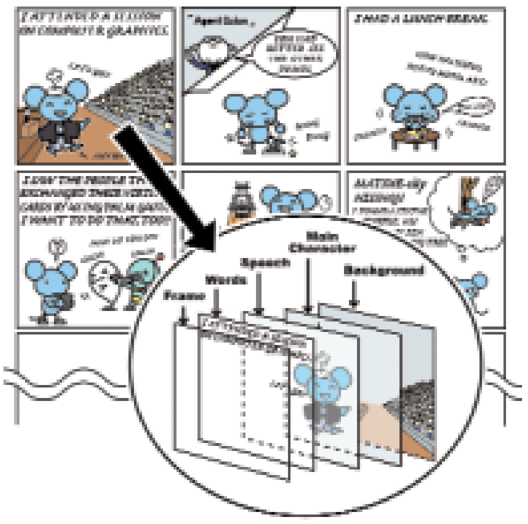
1. When it receives a user query via CGI, the generative engine in the server-side module considers the user's actual history of exploring the exhibition site from user-database logs involving other C-MAP services.
2. The engine determines the type of user by examining personal data and chooses an adequate story in the comic knowledge base, which contains generative rules for comics. For example, if the user is a child, comic expressions and structure are created to be simple and easily understood.
3. The server-side module sends the user's illustration data, which is drawn in advance, to the client-side module, which lays out the illustration data according to the story.

For the prototype, we have chosen Macromedia Flash 5.0 as the platform for publishing the comic on the client side because it can handle vector data, and it operates on popular Web browsers. Users can view their comics at information kiosks and at home.

Future Work

This prototype uses only a quadrilateral frame. However, conventional comics provide descriptions with a large variety of frame shapes to provide dynamic effects. To overcome this limitation, we plan to analyze the frame structure of several published comics. We also plan to evaluate the effectiveness of the system in order to redesign the story and structure of each comic. Automatic story generation and frame composition for arbitrary comics rendering are other remaining issues.

References
1. Sumi, Y. & Mase, K. (2000) Communityware situated in real-world contexts: Knowledge media augmented by context-aware personal agents. In *Proceedings of PAAM 2000,* 311-326.
2. McCloud, S. (1994). *Understanding Comic,* Kitchen Sink Press.
3. Kurlander, D., Skelly, T., & Salesin, D.H. (1996). Comic chat. In *Proceedings of the 23rd Annual Conference on SIGGRAPH 96 Computer Graphics*, 225-236.



This is a horizontal half-page of a sample comic and the layer structure of the first frame. The main character is the guide agent used in the C-MAP system.

Compressing Large Polygonal Models

Contact
Jeffrey Ho
Beckman Institute
University of Illinois
at Urbana-Champaign
Urbana, Illinois 61801 USA
j-ho1@dizzy.ai.uiuc.edu

Kuang-Chih Lee
David Kriegman
Beckman Institute
University of Illinois
at Urbana-Champaign

With the recent and rapid advances in digital acquisition technology, meshes with millions if not billions of vertices are becoming increasingly common. Existing mesh compression/decompression algorithms are only effective if a representation of the mesh's entire topological and geometric structures (and other attributes) is small enough to fit in memory. Yet for a mesh with a few million vertices, one faces the possibility that there is insufficient memory on a regular desktop computer for the entire model. Our approach to compressing these large models is to automatically partition the mesh into submeshes of smaller size, depending on available local memory, and then compress them separately.

The main purpose of the mesh partitioning is to divide the input mesh into submeshes of roughly equal sizes (i.e., the partition should be balanced). However, from the compression standpoint, it is also desirable that each region of the partition is "localized" somewhere in the model, and the boundary of each region is as simple as possible. Straightforward mesh partitions using x; y; z coordinate axes or the level sets of some other linear functions generally do not satisfy these requirements (see Figure 1). Instead, we propose a simple partitioning scheme based on a simplified mesh. Using vertex clustering,[1] we obtained a simplified mesh that is typically 100 to 200 times smaller than the original. Each vertex of the simplified mesh has a weight that is the number of corresponding vertices in the original mesh that cluster to the given vertex, and each edge carries a weight giving the number of collapsed triangles that form the edge. The simplified mesh can be partitioned as a weighted graph to obtain a balanced partition that minimizes the edge cuts. The partition of the simplified mesh then induces a balanced partition of the original mesh that usually satisfies the two requirements above. Therefore, we have an in-core representation of a simplified mesh which is used as a kind of blueprint for partitioning and compressing the original mesh.

Armed with this partitioning scheme, we have experimented with two similar methods for compressing large polygonal meshes. One ignores the boundary identifications, while the other compresses the connectivity losslessly.

For the first method, we simply partition the mesh and compress each submesh separately without regard to how different cut boundaries should be identified. By cut boundary, we mean the non-empty intersection between two neighboring regions of the partition. The advantage of this approach is that it is easy to implement and can be built immediately on top of existing mesh compression software.[2,3] The vertices belonging to the cut boundaries are encoded twice; hence, depending on the way the mesh is partitioned and the number of regions in the partition, the number of duplicated vertices can range from as few as one percent of all the vertices to as high as 20 percent; in the worst case, it is possible that more than half of the vertices will be encoded twice. This clearly illustrates the peril of using an arbitrary partition. Using a simplified mesh as a guide, our partitioning scheme will almost never produce the worst-case partitions. It has to be noted, however, that minimizing the number of vertices on the cut boundary does not necessarily guarantee smaller size for the compressed code. Nevertheless, we have observed through our experiments that the compressed output produced by our partition scheme ranges from two percent to eight percent with an average of four percent less than the compressed output from a "bad" partition.

For the second method, we developed an efficient approach to encoding (and decoding) cut boundaries based on identifying their connected components. The main idea is to use run-length encodings for the regular vertices of the cut boundaries, while for the singular vertices, we simply encode them directly. For meshes with mostly smooth cut boundaries, this part of the compressed code is generally negligible for very fragmented and complex cut boundaries.

Currently, we have only used the connectivity of the simplified mesh. Future research will be focused on how to utilize the geometry of the simplified mesh to define non-linear prediction rules for more efficient geometry compression.

Table 1 shows that a lossless compression ratio greater than 15 to 1 can be achieved on the meshes shown in Figure 2.

References
1. Rossignac, J. & Borrel, P. (1993). Multi-resolution 3d approximations for rendering complex scenes. *Modeling in Computer Graphics*, 455-465.
2. Rossignac, J. (1999). Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transaction on Visualization and Computer Graphics, 5* (1).
3. Touma, C. & Gotsman, C. (1998). Triangle mesh compression. *Proceedings of Graphics Interface '98*, 26-34.

Model	Original File Size	Compressed File Size	Ratio	Bits/ Vertex
David	173 MB	10.1 MB	17	19.4
Lucy	533 MB	35.6MB	15.2	20.3

Table 1: Compression Results: 16-bit coordinate quantization used for David and Lucy. All compressions were done on a Sparc workstation with 58MB of RAM. Each submesh is compressed using Edgebreaker[2] and Parallelogram Prediction.[3]

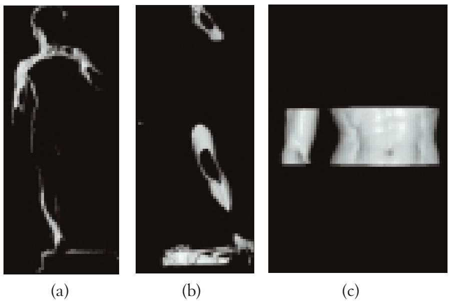


Figure 1. (a) This partition is induced from a simplified mesh; (b, c) partitions using z and y axes, respectively.

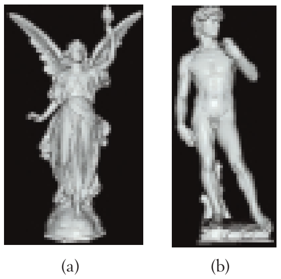


Figure 2. Two of our test models: (a) Lucy from Stanford 3D Scanning Repository. 28055742 triangles and 14027872 vertices. (b) David from the Digital Michelangelo Project. 8254152 triangles and 4129614 vertices.

A Computationally Efficient Framework for Modeling Soft Body Impact

Sarah F. Frisken
Ronald N. Perry
MERL

Introduction

While there has been significant progress in simulating collisions between rigid bodies,[1] much remains to be done for modeling interactions between soft bodies. Graphical techniques for representing and deforming soft bodies range from non-physical (e.g., control point-based) to physically plausible (e.g., FFD) to physically realistic (e.g., FEM).[2] All of these techniques require three operations to model interactions between soft bodies: Detecting collisions between deforming bodies, computing impact forces when bodies collide, and determining deformation forces or contact deformation of the bodies to initialize a deformation technique. In this sketch, we propose a new framework that performs all three operations quickly, with efficient use of memory, and more accurately than previous methods. The results of these operations can be used in any of the deformation techniques mentioned above.

Utilizing ADFs for Modeling Soft Body Impacts

We recently proposed adaptively sampled distance fields (ADFs) as a new shape representation and suggested that they might be useful for collision detection.[3] ADFs adaptively sample the signed distance field of an object and store the sample values in a spatial hierarchy (e.g., an octree) for fast processing. ADFs have several advantages for modeling impacts between soft bodies including: Compact representations of complex surfaces, trivial inside/outside and proximity tests, fast localization of potential contact regions, more accurate representation of the overlap region, and simple methods for computing material-dependent contact deformation.

Detecting Collisions, Penetration, and Proximity

The sign of the distance reconstructed from the ADF at any point in space provides a trivial inside/outside test. When detecting collisions between two ADFs, their spatial data structures can be exploited to quickly localize potential regions of overlap. Within these regions, a new ADF of the shape defined by the intersection of the two ADFs is locally generated as illustrated in Figure 1(a). (The intersection is a simple *min()* operation on the distance fields of the two ADFs.) If the intersection ADF is non-empty, a collision is detected and the region is further processed. To test for proximity rather than collisions, an ADF defined by the intersection of offset surfaces can be generated as illustrated in Figure 1(b).

Computing Impact Forces

There are a number of methods for computing impact forces between two interacting bodies.[4] Penalty-based methods compute impact forces based on how far objects penetrate each other during a discrete time step. Distance fields have been used[5,6] to determine penetration depth at sample points along the penetrating surface as well as to compute contact forces. From Figure 1(c), the total force vector, $\mathbf{F}_V$, acting on V due to penetration of U by V is approximated by the sum of forces $\mathbf{f}_{Vi} = k_U(\mathbf{x}_i)\text{dist}_U(\mathbf{x}_i)\mathbf{g}(\mathbf{x}_i)$, where $k_U(\mathbf{x})$ is the material stiffness of U at $\mathbf{x}$, $\text{dist}_U(\mathbf{x})$ is the closest distance from $\mathbf{x}$ to the surface of U, and $\mathbf{g(x)}$ is the normalized gradient vector of U's distance field at $\mathbf{x}$.

The intersection ADF represents the *volumetric* overlap region to high precision. Previous penalty methods compute $\mathbf{F}_V$ from a small number of points *restricted* to the penetrating surface. Using ADFs, penetration forces can be computed over the surface or the volume of the overlap region. Forces can be computed at an arbitrary number of well-spaced sample points seeded on the surface or throughout the volume, and we are investigating methods to analytically interpolate forces throughout the overlap region. These advantages of ADFs provide the opportunity to compute impact forces more accurately than previous methods.

Determining Contact Deformation

When using ADFs, there are two methods for determining the initial contact deformation. The first follows common practice and uses the impact forces computed above together with a deformation technique. The second uses the implicit nature of distance fields to compute contact deformation by combining the distance fields within the overlap region. Various methods can be used to combine the fields and achieve material dependent deformation[7,8] (see Figure 2). Figure 3 shows a simulation for the impact of two soft bodies.

References
See www.merl.com/reports/TR2001-11

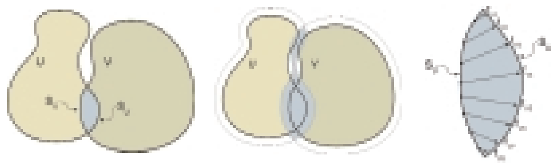


Figure 1. (a) The overlap region (in blue) of shapes U and V is determined by local generation of the intersection of U and V; (b) the overlap region of the offset surfaces of U and V can also be easily generated for determining proximity information; (c) forces acting on S_V in the overlap region of (a).

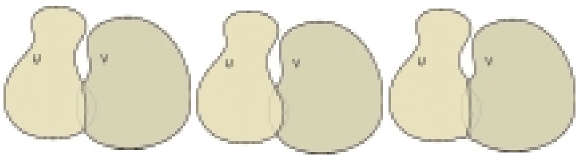


Figure 2. Contact deformation of the objects of Figure 1 with (a) similar material densities; (b) V softer than U; and (c) volume preserving deformation.[7,8]

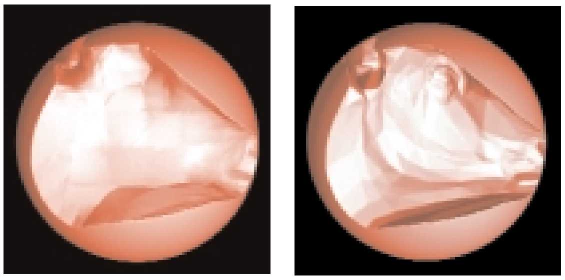


Figure 3. Indentation of a sphere after impact with an ADF of a complex cow model. Contact deformation of the sphere after impact with a soft cow (a) and a hard cow (b).

Computing 3D Geometry Directly from Range Images

Sarah F. Frisken
Ronald N. Perry
MERL

Introduction

Several techniques have been developed in research and industry for computing 3D geometry from sets of aligned range images.[1] Recent work has shown that volumetric methods are robust to scanner noise and alignment uncertainty and provide good quality, water-tight models.[2,3,4] However, these methods suffer from limited resolution, large memory requirements, and long processing times, and they produce excessively large triangle models.

Volumetric methods[2,3,4] construct range surfaces for each aligned range image and fill a (fixed-resolution) volumetric representation with signed distances from the range surfaces. The methods use various approaches to reduce the time required to fill and access this volume data, including run-length encoding of the distance values, binary encoding of regions outside a bounded region of the surface, and a three-color octree representation of the volume. The distance values from multiple scans are combined probabilistically using order-independent or incremental updating. Finally, these methods build a triangle model of the iso-surface of the distance volume using Marching Cubes.

In this sketch, we propose a new volumetric method for computing geometry from range data that:

- Computes distances directly from range images rather than from range surfaces.
- Generates an adaptively sampled distance field (ADF) rather than a distance volume or a three-color octree, resulting in a significant savings in memory and distance computations.
- Provides an intuitive interface for manually correcting the generated ADF.
- Generates optimal triangle models (with fewer triangles in flat regions and more triangles where needed to represent surface detail) from the generated ADF octree using a fast new triangulation method.[6]

Corrected, Projected Distance Images

Constructing 3D range surfaces and computing distances from these surfaces contribute significantly to the computational requirements of volumetric methods.[2,3,4] If, instead, the distance field could be generated directly from 2D range images, model generation times could be reduced. However, range images do not provide true distance data. In the simplest case, a range image records the perpendicular projected distance from the object surface to the image plane. The projected distance field is the same as the true distance field in two circumstances: Throughout the field for a planar surface parallel to the image plane, and at the surface (where both distances are zero) for any surface. Except for the first case, the projected distance field differs from the true distance field for points off the surface, resulting in artifacts when combining projected distance fields from different viewpoints.

For a planar surface, it can be shown mathematically that the difference between the true distance and the projected distance at a location $\mathbf{x}$ is inversely proportional to the magnitude of the distance field gradient at $\mathbf{x}$ when the gradient is computed using central differences. Here we propose to *correct* the 3D projected distance field by dividing sampled distances by the local gradient magnitude. This results in a better approximation of the true distance field near the surface, yielding better results when combining projected distance fields. Computing the local 3D gradient to make this correction could be prohibitive (it requires six additional distance computations). Instead, we derive the 3D gradient from a 2D gradient image generated once during preprocessing, resulting in significantly faster generation.

Adaptively Sampled Distance Fields

We recently proposed ADFs as a new representation for shape.[5] ADFs adaptively sample the signed distance field of an object and store the sample values in a spatial hierarchy (e.g., an octree) for fast processing. ADFs are memory-efficient and detail-directed, so that distance values are computed from the range images only where needed (mostly near highly detailed regions of the surface). Even in 2D, ADFs have been found to require 20 times fewer distance computations than a comparable three-color quadtree representation.[5] Finally, ADFs can be interactively edited via a sculpting interface[6] so that holes and other surface anomalies from occlusions and sensor noise can be easily corrected.

The ADF is generated from sequential or order-independent range images using a tiled generator.[6] Currently, distance values from the range images are combined as though carving the shape from a solid cube of material using a Boolean differencing operator; we have also begun experimenting with adding probabilistic combining functions[2,3,4] for robustness to sensor noise.

References
See www.merl.com/reports/TR2001-10

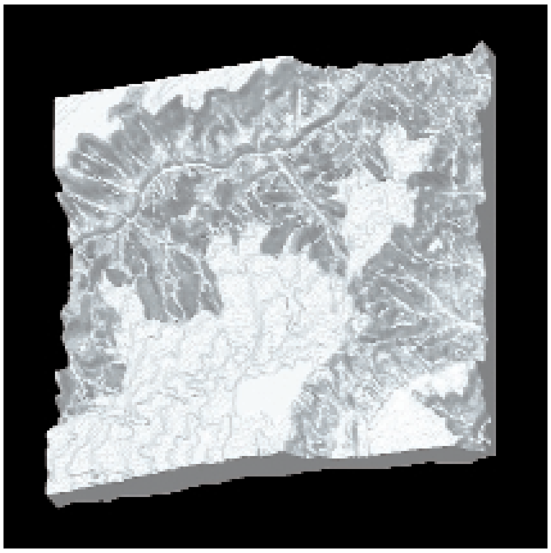


Figure 1. An ADF generated from an 800 x 800 elevation image of the Grand Canyon (data courtesy of USGS). Generation time (from range image to rendered model): 15 seconds.

Computing Nearly Exact Visible Sets within a Shaft with 4D Hierarchical Z-Buffering

Contact
Ned Greene
NVIDIA
ned@ngreene.com

We introduce a method for determining visibility within a shaft that exploits nearly all available occluder fusion and culls nearly all occluded primitives, failing to cull only when an occluded primitive is within a user-specified distance epsilon of being visible. Typically, the resulting epsilon-visible set of primitives only slightly overestimates the true visible set, culling more effectively than other practical methods. One important application is determining nearly minimal sets of primitives that are visible from a viewing volume. The method produces a 4D z-pyramid that can later be used for efficient on-the-fly visibility queries.

Within a shaft having quadrilateral endcaps, a primitive P is visible if there is a ray originating on the viewing endcap and piercing the other endcap (a "shaft ray") that intersects P before intersecting any other primitive. We will say that a primitive is epsilon-visible if there is a shaft ray connecting it to the viewing endcap, and occlusion of this ray by other primitives, if any, could be eliminated by moving each of them a distance of epsilon or less.

We construct an epsilon-visible set by subdividing a shaft until its subshafts are occluded by individual primitives, permitting deeper primitives to be culled, as illustrated for a 2D shaft in Figure 1. In this example, subshafts A-C and A-D are occluded by P1, subshaft B-C is occluded by P2, and additional subdivision would establish that shaft B-D is occluded by P2 and P3, and that the red primitives are occluded. Similar hierarchical methods have been applied to radiosity computations (e.g., Teller & Hanrahan, Global Visibility Algorithms for Illumination Computations, SIGGRAPH 93).

The recursive subdivision procedure that computes an epsilon-visible set takes as input the shaft's endcaps and a depth-sorted list of primitives, which it steps through in near-to-far order, testing each to see if it occludes the entire shaft. If so, a farthest depth value for the shaft is established, and any primitive whose nearest depth is deeper is occluded and therefore culled from the primitive list. The nearest primitive within the shaft and the nearest primitives on each of the shaft's "edge rays" are marked visible. After processing all primitives on the list, if not all have been culled or marked visible, the shaft is subdivided into the 16 subshafts formed by connecting all combinations of the four sub-quadrilaterals of one endcap with those of the other. This same procedure is then called with each subshaft's endcaps and the primitives on the current list that intersect that subshaft. Depth-first subdivision proceeds until all remaining primitives within a shaft have been culled or marked visible, or until the shaft width falls below the user-specified distance epsilon, whereupon primitives that have not already been marked visible are marked ambiguous. When the procedure finishes, the visible and ambiguous primitives are the epsilon-visible set, and all other primitives are known to be occluded.

This procedure could be accelerated with graphics hardware by using the method of Durand et al. to detect occlusion of shafts by collections of primitives (Conservative Visibility Preprocessing using Extended Projections, SIGGRAPH 2000). Efficiency could also be enhanced by processing in multiple passes, where the level of subdivision increases from pass to pass, and by organizing the scene in nested bounding boxes and testing boxes for visibility before testing the primitives that they contain.

In the example of Figure 2, epsilon corresponds to seven levels of subdivision, and our procedure established that 50 polygons were visible (blue/green), one was ambiguous, and 64 were occluded (red). Down to subdivision level five, we saved farthest depth values of shafts in the 4D z-pyramid of Figure 3 (simultaneous 2D subdivision of endcaps produces a 4D image).

Saving a z-pyramid enables fast on-the-fly conservative culling later of other objects within the shaft, using a visibility-query procedure analogous to conventional 2D hierarchical z-buffering (Greene, Kass, & Miller, Hierarchical Z-Buffer Visibility, SIGGRAPH 93). Just the coarsest levels of the pyramid provide a compact occlusion image, which we used to determine that square S was occluded in Figure 2.

In conclusion, our method is essentially 4D hierarchical z-buffering that finds visible geometry by adaptive subdivision, doing only as much work as necessary to establish visibility within a specified tolerance. We have found this approach to be a practical way to compute nearly exact visible sets within shafts containing many thousands of polygons.

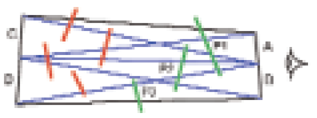


Figure 1. First level of subdivision of a 2D shaft, which established that primitives P1 and P2 occlude sub-shafts A-C, A-D, and B-C.

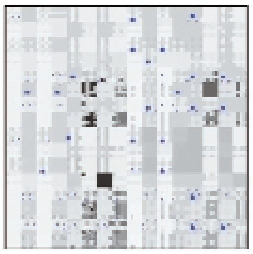


Figure 3. Corresponding "4D z-pyramid" encodes depth of occlusion within shafts.

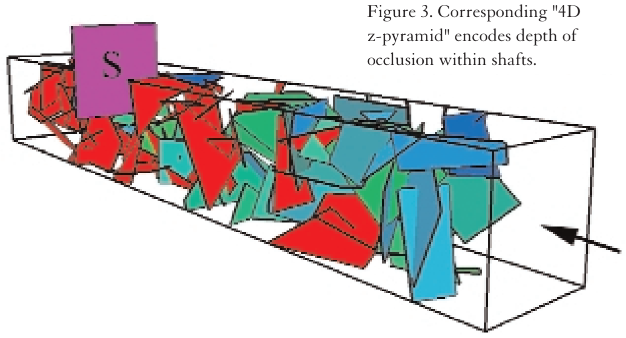


Figure 2. Blue-green polygons are visible and red polygons are occluded with respect to the set of rays originating on the near endcap and ending on the far endcap.

CONTIGRA: A High-Level XML-Based Approach to Interactive 3D Components

Contact
Raimund Dachselt
Dresden University of Techbnology
dachselt@inf.tu-dresden.de

Motivation

This sketch reports on a new approach to facilitating construction of interactive 3D graphics applications for the Web and introduces the component architecture and underlying high-level languages.

Three-dimensional Web graphics technology is used in an increasing number of application areas such as product presentations, teaching, or 3D navigation (see Figure 1). Considering the huge improvements in graphics hardware and the fast-evolving Internet technologies, it seems surprising that 3D applications are still not widely used. One problem is dependence on proprietary 3D formats or the less successful VRML standard. Even with its promising successor, X3D[1], the wide variety of Web 3D formats will persist. The second major problem is lack of design standards, authoring tools, and concepts of reuse. This is why building 3D applications is still time-consuming and heavily dependent on programming skills. However, interdisciplinary development of 3D solutions (for example, for entertainment or shopping) using building blocks or 3D components is inevitably necessary.

The few existing component approaches like 3D-Beans[2] or i4D[3] have the disadvantage of 3D format dependency or dependence on proprietary technologies. Also, component technologies like JavaBeans are inherently code-centered and thus difficult for non-programmers to use. This sketch addresses these problems and proposes a flexible solution.

The CONTIGRA Approach

The acronym stands for Component OrieNted Three-Dimensional Interactive GRaphical Applications. The approach introduces a 3D component concept that is largely independent of implementation issues and allows easy, declarative, and interdisciplinary authoring of 3D applications. It is based on structured documents describing the component implementation, their interfaces, and assembly and configuration. The core of the architecture consists of markup languages based on XML, allowing consistent, declarative description of complex 3D scenes. XML has the advantages of hierarchical descriptions matching scene graph concepts, powerful transformation capabilities using DOM or XSLT, and interoperability with other Web technologies. The three multi-layered CONTIGRA markup languages are coded with XML schema.

CONTIGRA SceneGraph

As an extension to X3D, this schema allows implementation of a 3D component in terms of geometry and behavior, which are described separately. The set of scene graph nodes is extensible. Using XLST or DOM, the documents can be translated to any 3D scene graph format. In addition to the SceneGraph component implementation, there are resource files (sounds, textures, scripts, etc.), all referenced in a homogeneous way (see Figure 2).

CONTIGRA SceneComponent

This component-description language is used to define component interfaces separated from their SceneGraph implementations. It is well suited for distribution, search, and deployment. Different sections of an interface document allow a rich component description, from offered functionality and configurable parts up to deployment and authoring information. As an abstraction to implementation details, high-level parameters hide scene graph fields. Other components can be encapsulated. Pointers to SceneGraph documents and available editors are included.

CONTIGRA Scene

This is a high-level configuration language for component integration. Documents coded with this schema represent a declarative description of interactive 3D virtual environments. They contain a hierarchical assembly of configured component instances, component connections, and general scene parameters like cameras, lights, etc. How are these grammars applied? People with knowledge of scene graph concepts implement and describe 3D components using the SceneGraph and SceneComponent level. After distribution, the components can be independently deployed and configured with an intuitive 3D user-interface builder that is now under development. For this tool, CONTIGRA Scene documents also serve as an exchange format. All the documents are finally transformed into a running 3D application, either at configuration time or on the fly in a Web browser, adapted to the configuration of the client.

The declarative high-level languages and the run-time framework are still works in progress. Our preliminary results already show the flexibility and feasibility of the component-based approach to authoring 3D applications. Its major achievements are platform independence, abstraction to specific 3D formats, componentization, and a declarative approach that is well suited for visual tool support.

References
1. Extensible 3D (X3D). URL: www.web3d.org/x3d.html
2. Doerner, R. & Grimm, P. (2000). Three-dimensional beans - creating Web content using 3D components in a 3D authoring environment. Web3D/VRML 2000.
3. Geiger, C., Paelke, V., Reimann, C., & Rosenbach,W. (2000). A framework for the structured design of VR/AR content. VRST 2000.

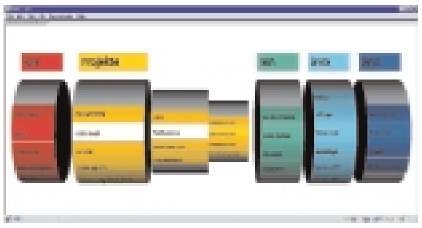


Figure 1. A 3D representation of a site map allows fast navigation among Web pages.

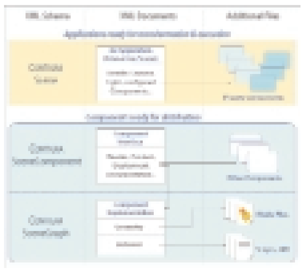


Figure 2. The multi-level Contigra XML suite and related component and application documents.

Crashing Planes the Easy Way: “Pearl Harbor”

Scott Benza
Lead Animator

Jim Hourihan
Principalz Engineer
Industrial Light + Magic

Aircraft destruction in “Pearl Harbor” was made possible by taking an alternative approach to rigid body systems. This presentation demonstrates ILM’s technique for creating realistic dynamic simulations for the film.

Problem: How do you animate destruction of a semi-rigid airplane as it collides with one or more other objects? The plane model may be over-constrained; it may be composed of more than 200 separate objects; and simulation compute time must be reasonable on a desktop workstation.

Solution: A number of algorithmic developments allowed us to achieve the required performance:

- Addition of “clustered” rigid body systems to maximize simulation speed and animation control. This allowed the animators to bypass the more traditional approach to crash simulations, which involve multiple simulation passes to achieve a performance. Clustering gave animators the control and speed they needed to set up an entire crash in a single simulation.
- Optimization of an existing simulation engine (SW1) to handle larger numbers of bodies.
- A robust animation framework.

We also developed a number of external physical force fields, including a pseudo-fluid air field and water field for floating objects. With air and water fields applied to the simulations, objects would flip and tumble through the air naturally, or float and sink once they hit the water.

Finally, a very tight connection was required to make data transfer between the external simulator and Maya as simple as possible. These connection tools gave animators copy/paste functionality directly between the software packages, which condensed an elaborate translation of huge amounts of data to a single mouse click.

Creating Tools For PlayStation2 Game Development

Contact
Kevin Rose
Sony Computer
Entertainment Europe
Cambridge, United Kingdom
kevin_rose@scee.sony.co.uk

Simon Brown
The Moving Picture Company
simon.brown@physics.org

Introduction

The tools department at Sony Computer Entertainment Europe Cambridge (SCEE Cambridge) created many tools needed for game creation during pre-production of our initial PlayStation2 game. The tools were built while programmers were working on the game engine and artists were creating concept art. The work is currently being used in production of our PS2 titles.

The required tools included a level editor with exporters, miscellaneous art tools, a pipeline, and a data-management system. We wanted to use off-the-shelf software wherever possible. We decided to use Alias|Wavefront's Maya for our editor, as it offers a proven platform with scripting facilities and an open API, and we used Microsoft's Visual SourceSafe for version control. These formed the foundation upon which our tools were built.

Game Object Specification

Game-play mechanics are built in the level editor using game objects. These objects describe collision areas, how characters behave and navigate, interactive lighting, etc.

In previous projects, writing changes to exporters and creating project specific tools was time-consuming, costly, and error-prone. To avoid these problems, we needed game objects that were customisable and extensible. We classified the many different types of game objects from previous games into six base types and created a scripting system to extend these specialised versions. Using object-oriented design principles, we considered special-case game objects to be derivations of the base types. This allowed us to focus on writing tools for a limited set of base objects, and new derived map-object types no longer required exporter or tool changes.

Our base object types are an extension of Maya DAG-nodes and utilise the MEL script language for adding parameters that describe the object. A limited set of parameters for deriving objects was defined. These described information such as floats, position, file resources, and storage requirements. Using our simple scripts, programmers can derive new game objects. Maya reads these scripts, and our tools automatically generate the appropriate GUI. The exporters, using the fundamental set of base objects and the allowable parameter types, build the new object-data structures.

Pipelines

Disparity between the code and art resource management and a lack of reliable version control for art resources caused problems on previous projects. We wanted a simple system with a minimum amount of custom coding for PS2 projects, so we chose Microsoft's Visual SourceSafe, as it is simple to use and maintain. The level editing is done in Maya using custom plug-ins and scripts. Code, game-object scripts, and resources are read into Maya from SourceSafe as required, ensuring that everything is up to date.

Many copies of each landscape are kept, some for artwork, and some for game objects, allowing game play and artwork to be edited concurrently by many people. Our exporters then combine all the exported data for use in the game. Using Maya's scripting system, the exporters also add exported data to SourceSafe, carry out "behind-the-scenes" housekeeping, and start up the game with the exported data. This pipeline is transparent to the user and has been kept relatively simple.

Art Tools

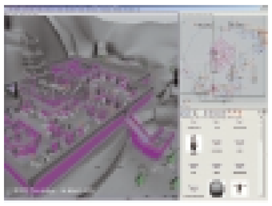
Many art tools have been created, ranging from tools to allow "vertex" colouring on NURBS surfaces to tools that check how suitable a given mesh is for game use.

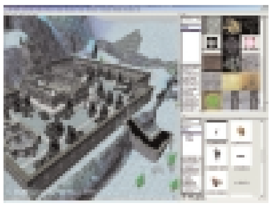
One particularly useful tool is the plant tool. Placing geometry accurately on a surface can be time-consuming, so a plug-in was written to "plant" a selected object where the user clicks on another surface. Variations on this approach allowed us to directly plant instances and duplicates of objects.

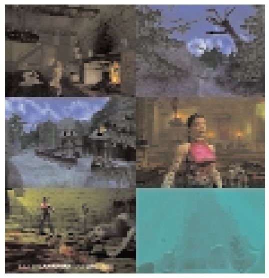
Custom GUIs were created with MEL scripting to present the artist with palettes of textures, geometry, and game objects. Our GUIs connect directly to SourceSafe, so they always retrieve the most recent files.

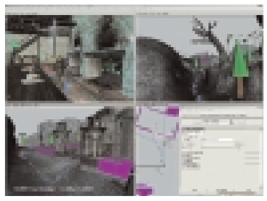
Acknowledgements

We would like to acknowledge Guillaume Raffy, Andrew Ostler and Adam Garman who worked with us to create these systems and tools.


Maya showing part of a level with game objects and the game-object browser.


Maya screen grab showing the same part of level, but working on the artwork with geometry and texture browsers.


PlayStation2 screen grabs. Real-time views from two games currently in development. Three of the images are from the same viewpoints shown in the Maya screen grab.


Maya screen grab showing three views of a level.

Creative Approaches to the Integration of CGI with Traditional Animation in Disney's "Atlantis: The Lost Empire"

Contact
Kiran Joshi
Walt Disney Feature Animation
kiran.joshi@disney.com

Marcus Hobbs
Walt Disney Feature Animation

Over the years, we have seen the advantages of CGI as a tool to artistically embellish and execute scenes that would be cost-prohibitive if done with traditional methods. The task of animating an action adventure feature of epic proportions, as in Disney's first action-adventure animated feature, "Atlantis: The Lost Empire," was only feasible by utilizing CGI. However, the challenges of when, how, and how much CG to use became a matter of creative, technical, and production balance. This sketch provides an overview of artistic and technical approaches to blending the two worlds.

Disney has always sought to enmesh the worlds of 2D and 3D to retain the company's traditional feel while at the same time taking advantage of the wonderful opportunities provided by computer graphics. For "Atlantis," the blend of techniques had to be seamless, or the comic book art direction would not work cohesively. We were embarking on a rather dramatic change in style, genre, and, ultimately, technique. Categorically, it would have not been possible to complete the movie without digital techniques.

The highly stylized, bold posterized look of the film had to be mastered by both the digital artists and the traditional artists in their rendering. The look and the animation style had to be resolved for each scene so the CG elements could be successfully integrated. Therefore, CG elements had to be developed with the same level of flexibility as hand-drawn elements. The details of a CG element were not baked into one level. Creation of many levels of artwork for a CG element provided the flexibility to blend its look to match hand-drawn artwork from scene to scene.

Due to the sheer volume of scenes, it was very clear that we would be required to animate elements both ways, so the method had to be transparent to viewers. Acknowledging the strength and weaknesses of digital mathematical perfection and the natural imperfections of traditional animation was the key to successfully blending of these worlds. Techniques were chosen for their appropriateness to the shot and subject.

To help the traditional animators animate their elements and register hand-drawn artwork to CG-rendered artwork, a custom tracking software was enhanced to utilize the plots of the rendered artwork. This allowed the animators to draw to a fixed, scaled image reference without having to worry about size and motion. The hand-drawn animation was tracked to the CG-animated element's motion path.

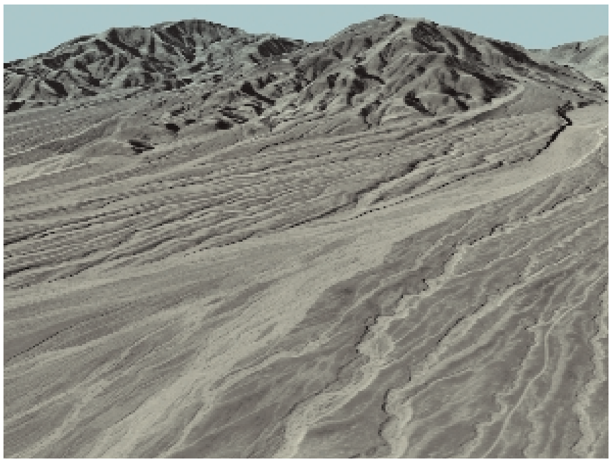
With a variety of video clips, this sketch illustrates how we integrated CGI to the creative process in each step of the production. We discuss the creative and technical integration challenges from visual development to production pipelines.

Death Valley Flyby: A Simulation Comparing Fault-Controlled Alluvial Fans Along the Black Mountain Range

Contact
Carlos Seligo
Stanford University
423 Sweet Hall
590 Escondido Mall
Stanford, California 94305-3091
USA
+1.650.736.1432
moth@stanford.edu

Charley Weiland
Academic Technology Specialists
Stanford University

This animation sketch presents how public US Geological Survey (USGS) data can be used to simulate complex geological phenomena for an undergraduate seminar. Our simulated flyby over Death Valley was sufficiently accurate and detailed to represent not only major fault-controlled deformations, but even the sparse vegetation along alluvial fans and evidence of recent seismic activity that was not yet eroded by flash flooding. Finding and downloading data of sufficient resolution, correctly registering 2D images on 3D elevations, and stitching together the quadrangles covered in the flight path proved challenging in a computing environment that was not dedicated to this task. Nonetheless, now that we have solved these problems, we hope the procedure might be more generally applied to rendering geological formations of any site for which similar data are available.



Technical Problems Rendering 3D Simulations From Public Data

In November 2000, a professor contacted us for technical support of a class on Death Valley. Half-jokingly, she requested an airplane to shoot the fault along the Black Mountain Range, which was beyond the budget of an undergraduate seminar, but we guessed that it would be possible to simulate such a flight from public data. Ten years ago, rendering a photographically realistic and geologically accurate flyby of a fault would have been a piece of experimental computer science that only the military or NASA could afford. Today, the major technical problems have all been solved, but applying these technical solutions in support of a seminar was not a trivial task.

The USGS no longer offers most of its public data for free, but we were able to find nine contiguous 7.5-arc-minute digital elevation models (DEMs) of Death Valley at 30-meter resolution. We downloaded the aerial photographs, or digital ortho quads (DOQs), that corresponded to each DEM from the Digital Alexandria Project. At one-meter resolution, four DOQs make one DEM, and stitched together, these image files were 170 megabytes each before compression. These proved cumbersome to match in brightness and contrast, and because we lacked the geographical tools to adjust them to the curvature of the earth's surface, the registration marks in the 2D images did not automatically correspond with the corners of the 3D models. Nonetheless, we were able to produce simulations that were scientifically accurate enough for our educational purpose using off-the-shelf software: World Construction Set, Lightwave, Photoshop, and Media Cleaner Pro.

How Simulations May be Applied More Generally to Serve a Wide Variety of Pedagogical Goals

Finally, we would like to suggest how these simulations might be applied more generally to serve the pedagogical aims of educators. For a price, the National Elevation Dataset will burn CDs of comparable data for anywhere in the contiguous United States. GIS Data Depot also provides file-format translators and an uneven selection of free models from all over the world. Not only geologists can benefit from accurate simulations of a landscape. If architects used these tools to preview the views from the windows of the buildings they design, an art historian could reconstruct Cezanne's "Mountains Seen from l'Estaque!"

Decomposing Animation into BlendShapes

Contact
J. P. Lewis
OHP
501 Glenoaks Boulevard
Suite 656
Glendale, California 91202 USA

This sketch describes some preliminary work on decomposing animated motion into a set of localized "blendshapes" (also called morph targets and shape interpolation) that can be combined to produce that motion. In computer animation, information traditionally flows in the opposite direction. Predefined blendshapes or other primitives are combined to produce the animated movement. To motivate blendshape decomposition (BSD), we note that several computer-vision techniques allow model streams to be acquired from cameras, and these techniques may become routine in the future. The form of the captured data (a stream of unlabeled meshes, often without frame-to-frame registration) does not lend itself to post-capture editing, however. This leads to the frequent comment that motion capture does not reduce the animator's workload. Some post-capture editing of the data is presumed necessary, but this requires constructing an animator-friendly model (often using blendshapes) and then animating this model to mimic the captured performance. Motion capture can obviously drive an animator-friendly model through some automated process, but producing this model remains an issue. Our work is motivated by this issue.

Viewed abstractly, BSD can be seen as looking for a decomposition

$$A = BW \quad (1)$$

where for each of N_p frames of performance the model containing N_m control points is unrolled into a column of A ($N_m \times N_p$), columns of B ($N_m \times N_b$) are the basis being sought, and columns of W ($N_b \times N_p$) are the weights (encoding) for corresponding frames of A; the rank of A is assumed to be N_b, so the blendshapes can be regarded as compressing the performance.

Decomposition techniques such as principle component analysis (PCA) and independent component analysis (ICA) provide a starting point for consideration of the BSD problem. The PCA of A is easily obtained from its SVD, $A = UDV^T$; the U matrix is the desired basis. PCA has well known characteristics that make it unsuitable for our purpose, however. The basis shapes are orthogonal, not localized, and usually have no intuitive interpretation, whereas an animator-friendly BSD needs bases with intuitive and preferably local control, such as "smile" or "raise right eyebrow." In other fields, such considerations (especially the lack of locality) has led to the search for alternative decompositions. ICA produces a decomposition in which the basis shapes are not required to be orthogonal, and locality has been demonstrated. On the other hand, ICA is formulated as the solution of the blind source-separation problem in which independently moving sources are added in an unknown combination. Although the BSD problem might be put in this form, at the outset the characterization of individual shapes as completely independent (and an additional requirement that they have non-Gaussian densities) seems unnatural and restrictive. Consideration of other recent literature suggests that this restriction can be avoided.

Two techniques provided inspiration for this work: the Atomizer basis pursuit system and positive matrix decomposition techniques. In Atomizer,[1] a large, overcomplete set of basis functions is generated prior to examining the data. A particular data set is represented by finding weights that minimize the 1-norm rather than the usual 2-norm. The authors observe that the 2-norm disproportionately penalizes large weights, resulting in many small weights that are roughly the same size. In BSD, as in Atomizer, we desire a sparse coding whereby a local deformation is preferably produced with a single (or few) delta shape(s) rather than an overlapping combination of many shapes. Unlike Atomizer, we want a tight rather than overcomplete basis, and we want to derive the bases functions rather than selecting from a large predefined set.

Several numerical techniques (LININPOS, PMF) provide a factorization of the form (1) when A,B,W are all restricted to be positive; software for the PMF algorithm is available commercially.[2] Discussion of these techniques yields a second useful observation: restricting the weights and bases to be positive encourages locality. When weights are unrestricted, a localized shape can be explained with broad bases combined in a cancelling fashion with positive and negative weights. Unlike PMF techniques, we will require both positive and negative excursions in the basis shapes, however.

BSD prototypes that explore locality principles, including those mentioned, above have been implemented. The algorithms perform gradient descent minimization of $tr(A - BW)^T (A - BW)$ starting from random B and W and reprojecting B on the subspace U during the descent iteration. Evaluations of these approaches are not complete, but many one-dimensional and a few three-dimensional simulations have been performed. In these simulations a synthetic dataset was created by randomly combining blendshapes, and the simulation then tries to recover similar local basis shapes without knowing the originals (Figure 1).

References

1. Chen, S. & Donoho, D. L. (1995). Atomic decomposition by basis pursuit. Stanford Dept. Statistics TR 479, May 1995.
2. Paatero, P. (1997). Least squares formulation of robust non-negative factor analysis. *Chemometrics and Intelligent Laboratory Systems 37*, 23-35.

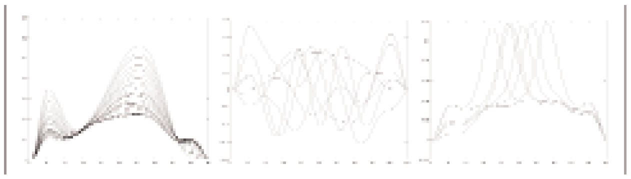


Figure 1. Left: 1D dataset; center: orthogonal basis is complex and not local; right: evolved local basis.

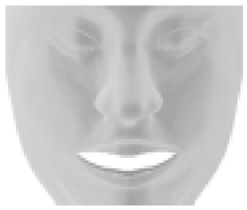


Figure 2. Superimposed original target and recovered delta shape. Note that this is a relatively easy result since most of the geometry here is obtained by keeping the solution B close to a subspace defined by only three shapes.

Destroying a Dam Without Getting Wet: Rule-Based Dynamic Simulation for "Wave of Death"

Contact
Stephan Trojansky
CA Scanline Production
Bavariafilmplatz 7
82031 München, Germany
+49.89.6498470
+49.89.64984711 fax
troja@scanline.de

Digital Artists
Stephan Trojansky
Florian Hu
Roland Langschwert
Fritz Beck
Sebastian Küchmeister
Albrecht Steinmetz
CA Scanline Production

The shots presented in this sketch show the breaching of a huge dam. Each shot has been completely computer generated, including the environment, the dam, chunks of concrete, fragments, dust, water, and mist. The work was done by using a rule-based dynamic simulation technology that provides a new technique for digital creation of natural phenomena or catastrophes.

The Challenge

The film "Wave of Death" revolves around a huge dam that is blown up by terrorists at the end of the story. The destruction of the dam was supposed to be shown in a two-minute effects sequence consisting of 17 shots, with people running on the dam while it slowly disintegrates under their feet.

Since the Scanline team is always looking for new challenges, the thought of a "dry" digital solution, where only the close-ups of the actors are live action, made our eyes sparkle. That alone would have been challenge enough, but after a close look at the very limited budget, we had the feeling of being flooded ourselves. We had a production schedule of six weeks and a team of six animators to get this job done. Our main concern at that point was finding a way to realize this project without getting more than "wet feet."

The Approach

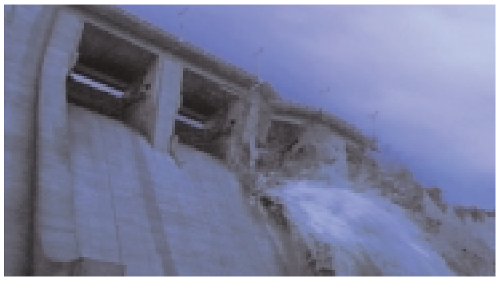
To achieve extraordinary results in such a short span of time, we had to automate as much as possible, to free the animators from doing work that could be done just as well by a software tool. At the same time, we didn't want to be at the mercy of a mathematical simulation. We wanted to keep the creative direction of the action in our hands. The fact that Scanline is working with proprietary animation software helped a lot. We were able to combine our know-how with the rule-based simulation idea of Cebas Computer, a well-known German software-development team.

The Solution

The tool we developed provides an efficient and at the same time flexible way of creating dynamic simulations. In contrast to conventional keyframe animation or pure dynamic simulation, rule-based dynamic simulation allows complete control of the dynamic reactions and interactions of all particles involved, without the necessity to determine each one of them separately.

Once we had set up a rule for the "behaviors" of one element (for example, chunks, fragments, dust, water, mist), we were able to apply it to all the other elements of the same kind. By using this strategy, our dam breaching sequence, consisting of a few hundred thousand varying parts, could be controlled by alterations of only a few parameters.

www.scanline.de/WaveOfDeath



The Development of the Virtual Human Abdomen: Algorithms and Methodologies

Contact
Kevin Chugh
Virtual Reality Lab
State University of
New York at Buffalo
Amherst, New York 14260 USA

T. Kesavadas
Virtual Reality Lab
State University of
New York at Buffalo

James Mayrose
Department of
Emergency Medicine
State University of
New York at Buffalo

Simulating soft tissue in haptic virtual reality applications presents two difficult obstacles:

1. The ability to compute force-displacement calculations in real time. There are two issues that contribute to this hurdle. The first is that for continuous haptic sensations, an output rate of 1000 Hz or better is required. Second, traditional methodologies for computing force-displacement relationships employ polynomially growing algorithms.

2. The need for physical realism dictates that actual tissue parameters be extracted from tissue and then used in the simulation.

At the Virtual Reality Lab, we have developed a virtual human abdomen that simulates soft tissue for a virtual palpation exam. The simulation overcomes both of these obstacles and, at the same time, provides a realistic and physically accurate model.

Methodology

To overcome the first obstacle (real-time performance), we have developed a force-displacement computation algorithm/methodology called the Atomic Unit Method.[1] In this methodology, each discrete volume is modeled as an atomic unit, and each atomic unit has its own behavior. Forces are handed off from atomic unit to atomic unit, and the global behavior of the tissue model is computed by aggregating the atomic unit behaviors. Figure 1 depicts a portion of the human abdomen modeled as a set of atomic units before and after palpation.

Data Collection

Creation of a physically accurate simulation requires tissue parameters. In order to non-invasively collect material properties on human organs, we created a data glove[2] that allows the user to collect information about a palpation exam (the 3D position of the finger as it displaces the tissue, as well as the force used to cause that displacement). This information was used to calculate the stiffness and viscoelastic properties of human soft tissue within the loading ranges of palpation. These viscoelastic properties were then fed into the haptic simulation.

The Virtual Human Abdomen

Combining the real-time haptic calculation methodology with the data collection device allowed for development of the Virtual Human Abdomen. This is a real-time, physically accurate abdominal palpation simulation that uses real human anatomical images and real human tissue parameters. Figure 2 shows the simulation.

Presentation

This sketch describes three facets of the development of the Virtual Human Abdomen:

1. The real-time algorithms, their development and implementation, and their asymptotic qualities.
2. The functionality of the data collection device and its supporting algorithms.
3. Validation results. These experiments involved physicians performing palpations on the Virtual Human Abdomen.

References
1. Chugh, K., Mayrose, J., & Kesavadas, T. (2000). The atomic unit method: A physically based volumetric model for interactive tissue simulation. World Congress on Medical Physics and Biomedical Engineering, 2000. Chicago.
2. Mayrose, J., Chugh, K., & Kesavadas, T. (2000). A non-invasive tool for quantitative measurement of soft tissue properties. World Congress on Medical Physics and Biomedical Engineering, Chicago, 2000.

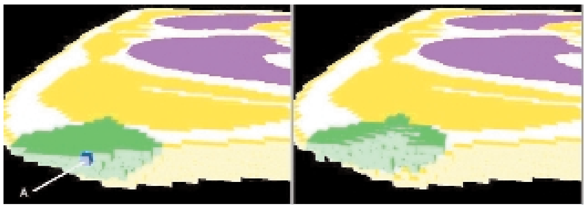


Figure 1. Before and after a palpation.

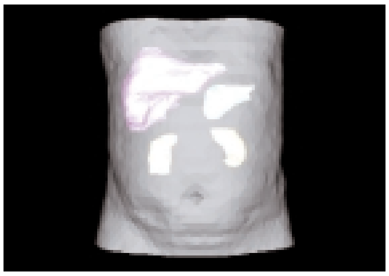


Figure 2. Surfaces are rendered for the Virtual Human Abdomen.

digital stupa: interactive animations

Contact
Daniel Pirofsky
pirofskyd@pdx.edu

Concept
These interactive animations are playful meditations on algorithmically discovered, non-objective images. Source animations are mapped to spatial grids, providing non-linear access to frames in any continuous or discontinuous sequence. Viewers can meditate on frames, play strands of continuous motion, or generate unique, spontaneous animations.

Application
Photographic virtual reality is a popular medium for interactions with panoramic or three-dimensional images. QuickTime VR panoramas present a screen-based immersive experience that places the user in the center of a scene. QuickTime VR object movies present a three-dimensional object in front of the user that is rotated to display alternative views. In panoramas, the viewer is the center of the universe; with objects, the user is on the periphery looking into and around the center.

Absolute object movies are an alternative, two-dimensional method of organizing multiple images in a non-linear format. Instead of rendering frames as multiple views from positions around the object, an absolute object movie positions images in a grid structure, allowing the user to click on any position to display a frame. This allows interactive access to multiple views and sequences of 2D or 3D images. In this framework, there is no center or circumference: space generates time as the matrix of frames is explored.

Exploiting this feature of QuickTime VR provides an interactive animation experience. Current work presents 12 interactive animations, each composed of 2,700 frames, the equivalent of three minutes of linear animation, mapped across a grid of 25 rows and 108 columns. When an animation loads, a short selection of looped animation plays to preview the imagery contained in the object. Clicking the mouse button down within the object stops the animation to display a single frame from a unique location in the grid. Dragging the mouse button down plays a sequence of frames. Moving horizontally across the grid isolates a single strand from among the 25 rows of 108 frames. Moving vertically down the grid selects frames from a variety of different strands. Stroking the animation in a series of interactions generates unique versions of the animation by displaying a series of continuous or discontinuous frames. Speed of the mouse motion controls the speed of the animation. This simple interaction in visual-temporal space can be intriguing, hypnotic, relaxing, or exciting as images appear, connect, and disappear. Across the limitless spectrum of algorithmically generated, non-objective imagery, traveling through these unique structures of visual space begins to trigger contemplative experience and insight.

Process
This project began by exploring the graphics technology of Artmatic software, which renders visual images from mathematical structures generated through a combinatorial algorithmic process. I was amazed at the richness of this natural visual order as I discovered, mutated, and rendered images, searching for novel and beautiful structures. From these structures, I rendered animations of sufficient length to explore these visual regions more deeply and systematically. Playing these animations, I stopped to gaze at key images that captured my imagination. Contemplation of their natural beauty led me to connect these frozen moments back to their original flow by replaying the animation. I meditated on these regions of visual space by first scrubbing across the timeline in an arbitrary, spontaneous, exploratory process, then interacting with specific locations to generate new sequences, revealing unique, tunable insights. This led to recomposing the linear animations into QuickTime VR absolute objects that offer a non-linear format to explore image sequences within the totality of visual space presented.

This enables free play with images and sequences, fast or slow, getting a feel for the totality of the animation and permutations of its structure. Temporal flow of images is presented in a spatial context. Time becomes a non-simultaneous experience of the totality of this visual space. Horizontal motion invokes continuity; vertical motion leaps to connect discontinuous location. The two-dimensional surface becomes an enjoyable play, and in that flow arises a meditation that is aesthetic and insightful in its reconfiguration of temporal into spatial media.

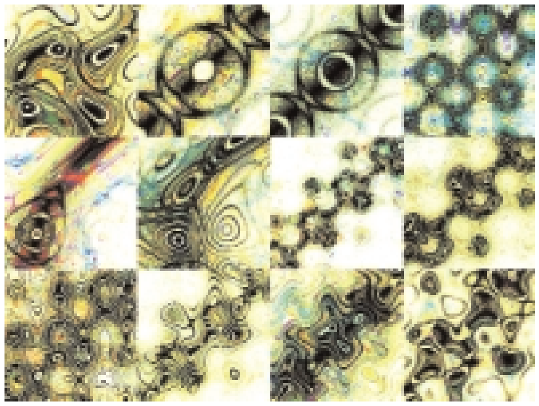


Figure 1. Representative frames from "horizons of unborn insight."

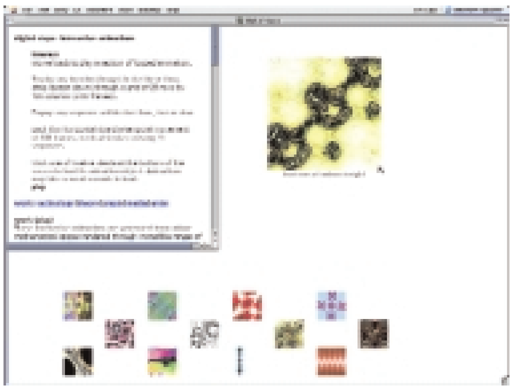


Figure 2. Screen image of digital stupa Web interface.

Digital Tricks for Japanese Cel Animation

Contact
Akio Kazumi
OLM Digital, Inc.
kazumi@ilm.co.jp

Megumi Kondo
Katsuaki Hiramitsu
Ken Anjyo
OLM Digital, Inc.

Yoshinori Dobashi
Hokkaido University

This sketch illustrates our unique and original uses of digital technology in creating Japanese cel animation. First we discuss cases that focus on merging 3D CG with cel-animated characters. Then we consider implicit the use of 3D techniques for making 2D painted, cel-based scenes more visually 3D and fantastic.

Merging a 3D Scene with 2D(Cel-Animated) Characters

In principle, the Western production pipeline is constructed to produce an exact realization of the storyboard of an animation piece. Each process in the pipeline is explicitly distinguished from other processes. This also means that the role of the creators is accurately prescribed. In other words, this production pipeline might be called "top-down," and it is an efficient way to achieve a desired result.

On the other hand, in many cases, Japanese animation is created from rough keyframes based on the storyboard, and then the keyframes are repeatedly arranged by checking test sequence rendering. So the Japanese pipeline is "bottom-up." For example, in the brand-new Pokémon movie, entitled "CELEBI A Timeless Encounter," many cuts are made with the bottom-up approach, where cel-animated characters are placed into a 3D background (see Figure 1). The bottom-up pipeline may force traditional cel animators to adopt a rather strict 3D sense, which represents an alternative way to get the desired result.

Fast rendering tools are indispensable in production work. This is very crucial for Japanese cel animation, because it must be produced on a tight schedule (as short as one year, even for a cel-animated film). In making "CELEBI", a custom fast volume renderer was developed for depicting shafts of light. This renderer is a non-photorealistic version of the fast volume renderer developed by Dobashi et al.[2] This tool is unique in the sense that it allows fine parameter tuning for non-photorealistic effects, such as decay of shafts, consistent with fast processing at high resolution rates.

Making 2D Painted Images More Visually 3D

In Japanese cel animation, implicit use of 3D techniques for cel-based 2D scenes is often preferable to explicit use of the cel-shaded 3D scene.[1] Camera projection mapping is a good digital technique for providing "visually 3D" effects: inputting a 2D-painted image of the scene to be animated. In the camera projection process, a pseudo-3D structure is constructed from the input 2D image, and then the resultant pseudo-3D model is animated with camera control restricted. An advanced use of this technique is illustrated in Figure 2, where a 3D blobby model is added to the pseudo-3D model of the painted bridge. Then mask animation of the pseudo-3D model is used to describe crystallization of the bridge.

Additional examples of our implicit uses of 3D techniques for cel animation include particles with patch-grid geometry for splash and wave expressions.

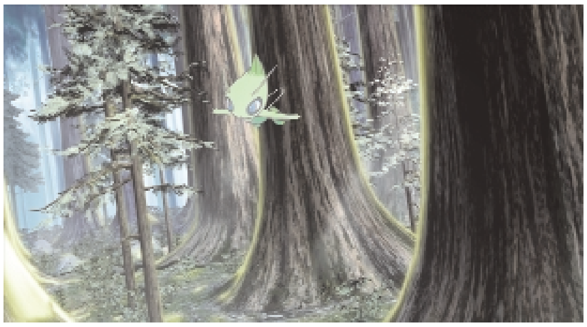


Figure 1. Frame from "CELEBI A Timeless Encounter."

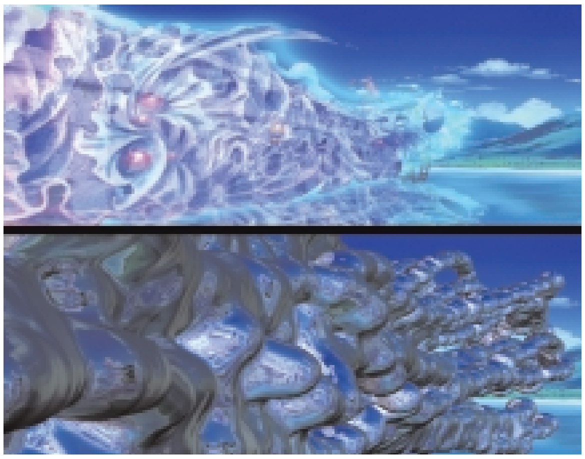


Figure 2. Frame from a promotional reel of "Lord of the Unknown Tower" (bottom) and 3D blobby hull of the bridge model (top).

References

1. Anjyo, K., Arias, M., Horry, Y., & Momose, Y. (2000). Digital cel animation in Japan. *SIGGRAPH 2000 Conference Abstracts & Applications*, 115-117.
2. Dobashi, Y., Kaneda, K., Yamashita, H., Okita, T., & Nishita, T. (2000). A simple, efficient method for realistic animation of clouds. *Proceedings SIGGRAPH 2000*, 19-28.

Dynamic Flesh and Muscle Simulation: "Jurassic Park III"

Dennis Turner
Technical Animation Supervisor
Industrial Light + Magic

Sebastian Marino
CG Software Engineer
Industrial Light + Magic

Problem

At first glance, creation of digital dinosaurs for "Jurassic Park III" was a problem with which ILM was well acquainted. Indeed, our digital creature pipeline is still based on the principles and techniques learned on the original "Jurassic Park" and honed on its sequel, "The Lost World." Beneath the surface, however, technological development and aesthetic expectation has increased unabated. ILM's work on projects such as "The Mummy" and "The Phantom Menace," and work from other studios, such as Disney's "Dinosaur," raised the bar that defines believable organic digital creatures.

Solution

We met the higher expectations for digital creatures by increasing the level of detail. Our performance criteria for the dinosaurs as actors remained the same. What changed was our expectation for how their bodies should react to the performance. In effect, the number of things that happen on the screen when the tyrannosaurus rex lunges forward after human prey increased exponentially.

Flesh simulation techniques developed for "The Mummy" were converted for use on dinosaurs. This allowed us to procedurally create the dynamic, or ballistic, effect of large organic bodies moving under the dinosaur effects of gravity and inertia. Within the dinosaur, we modeled and rigged objects, which represent muscles and bones. These objects make "guest appearances" in shots by providing the effect of masses moving under the skin.

Distinguishing physical differences of the dinosaurs were enhanced by the manner in which flesh simulations were employed. For corpulent creatures, such as the brachiosaurus, muscles and bones were rarely seen, if at all. The effect of dynamics applied to these bodies was nearly pure follow through with overlapping action. In contrast, the muscles and bones on lean creatures, such as the velociraptors, truly defined the form of the models as they moved. These two features were combined to full effect for the heavyweight prize fighter bodies of the tyrannosaurus rex and spinosaurus.

Artist Application

Additional detail was layered into ILM's existing creature-development and shot-production pipeline. The dinosaurs were modeled with additional surface detail and internal structures as needed, but otherwise were consistent with our standard methods for employing parametric surfaces. The models were then rigged with primary animation controls and enveloping. Additional muscle rigging and enveloping were developed in model sets that could cleanly replace the primary animation models at any time during shot production.

As we identified shots for which flesh simulation would provide perceivable effects, the simulation models replaced the primary animation model set. This was done only in a subsection of the shot pipeline, leaving the performance animation, lighting, and render procedures unencumbered.

Dynamic Meshing Using Adaptively Sampled Distance Fields

Jackson Pope
Sarah F. Frisken
Ronald N. Perry
MERL

Many models used in real-time graphics applications are generated automatically using techniques such as laser-range scanning. The resultant meshes typically contain one or more orders of magnitude more polygons than can be displayed by today's graphics hardware. Numerous methods have been proposed for automatically creating level-of-detail (LOD) meshes from large input meshes.[2] These techniques typically generate either one or more *static* LOD meshes, pre-computed before use in the application, or a *dynamic* mesh, where the LOD of the mesh adapts to frame rate requirements. We present a new dynamic LOD technique ideal for applications such as games and physical simulations based upon adaptively sampled distance fields (ADFs).[1] ADFs also provide fast collision detection as required by these applications.

Previous Work

Existing dynamic meshing algorithms such as view-dependent progressive meshes (VDPM)[3] and hierarchical dynamic simplification (HDS)[4] generate a hierarchy to efficiently process refinement and decimation operations. The hierarchy in VDPM is formed by creating a new parent vertex for every pair of vertices combined by an edge collapse operation. The HDS hierarchy is formed by spatially subdividing the scene into cells and grouping vertices in each cell into a single representative vertex. In both, the screen space error and normal cones (to detect back-facing and silhouette triangles) are used to determine when to refine and decimate the mesh. We present a new method that utilizes a spatial subdivision hierarchy, enables fast collision detection, and uses the distance field to position mesh vertices to optimize mesh shape.

Generating Meshes from ADFs

ADFs are a new shape representation that adaptively samples the signed distance field of an object and store the distance values in a spatial hierarchy (we use an octree).[1] We utilize a fast, new triangulation method that generates topologically consistent (orientable and closed) triangle meshes from the ADF structure.[5] Cells in the ADF octree that contain the object surface (where the distance field changes sign) are connected to their neighbors by triangles. The technique exploits the hierarchical nature of the octree to produce detail-directed triangles.

Algorithm

Our method creates a triangle mesh from the ADF, associating triangles with ADF cells, and then adapts the mesh in real time to viewing parameters to optimize visual quality (by using a high level of detail in visually important regions), while meeting user-defined frame rate criteria.

The algorithm is composed of two stages: a pre-processing stage and a real-time stage. The real-time stage is performed every frame or every few frames as required. The pre-processing stage initializes the data required for the real-time stage and creates an initial view-independent active cell list from which a triangle mesh is derived. Each active cell is associated with one ADF cell. Data initialization includes determining and storing normal cones in each boundary ADF cell; these cones bound the normal cones of all the cell's children. The hierarchical ADF structure enables fast view frustrum and back-face culling using normal cones.

The real-time stage consists of adapting and optimizing the existing active cell list and corresponding triangle mesh for current viewing conditions. During each adaptation, the active cells are considered to see if they contribute too many or too few triangles to the mesh according to view-dependent cell weights. If the number of triangles is appropriate, the cell is left alone. If the cell contributes too many triangles, triangles associated with the cell and its siblings are deleted from the mesh, the cell's parent is added to the active cell list, and triangles associated with the cell's parent are generated and added to the mesh. If the cell contributes too few triangles, the cell is added to an ordered list of such cells. To ensure that frame rate requirements are met, this cell list is processed in order only while there is frame time available. When processed, triangles associated with cells in the ordered list are deleted from the mesh, the cell's boundary child cells are added to the active cell list, and triangles associated with the cell's boundary child cells are generated and added to the mesh. The different treatment of cells with too many and too few triangles prevents the mesh from growing in size beyond the rendering capabilities of the graphics hardware.

Each cell is assigned a weight based on its contribution to the view. Currently, a cell is assigned a high weight if it is on the object's silhouette, and zero weight if the cell is back-facing or outside the view frustum. Other parameters could be considered, such as the projected screen size of the cell or whether the cell contains a specular highlight. In addition, our method uses the in-place cell error of the ADF as an indicator of surface roughness and curvature in the cell, and modulates the weight by this error.

Results

The technique produces detail-directed triangle meshes of high visual quality as viewed from the camera, while minimizing the number of triangles in non-visible portions of the object. It meets frame-rate criteria (currently, at 30 FPS it maintains ~25K triangles), even during viewpoint changes that lead to large differences in the visible portion of the object.

References
See www.merl.com/reports/TR2001-13

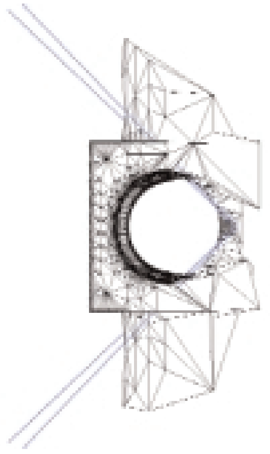


Figure 1. CSG object showing view frustrum (20364 triangle, 41 FPS). Note how areas outside the view frustrum are culled.

Efficient Update of Geometric Constraints in the Tapestry Evolving Mesh Representation

Maryann Simmons
Sara McMains
Carlo Séquin
University of California,
Berkeley

This sketch describes the use of motion bounds to optimize the evolving mesh representation used by the Tapestry[1] interactive rendering system. The tapestry dynamic display mesh is constrained to be a "spherical depth mesh" (i.e., its projection onto a sphere centered at the viewpoint has no overlapping elements). The mesh vertices are projected onto a sphere centered at the viewpoint to determine the mesh topology during incremental insertion of new points. This makes the insertion more efficient by reducing the problem to 2D. A Delaunay condition is maintained on the projected mesh to produce a good-quality image reconstruction. In addition to the depth mesh and Delaunay properties, minimum projected edge length and minimum projected vertex-edge separation constraints are maintained on the mesh to support a robust implementation. Here, we describe an efficient method for enforcing these constraints during view motion.

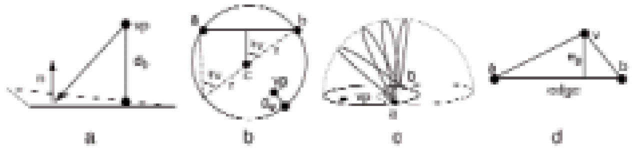
Calculating Conservative Motion Bounds

The mesh is constructed relative to an initial viewpoint. The geometric constraints are enforced by construction for this viewpoint. When the viewpoint changes, we want to retain as much of the 3D mesh as possible, while guaranteeing that all of the constraints are still satisfied. Since it is inefficient to test each triangle in every frame to make sure that its projected topology remains consistent, we assign conservative motion bounds to each triangle, stating how far the user can move before any constraints are invalidated.

For the depth-mesh constraint, this means that all triangles that will become back-facing in the new view need to be removed. We use the distance from the viewer to the plane of the triangle as a conservative bound d_b indicating how far the viewpoint could move before the triangle could be back-facing (see Figure a).

For minimum projected edge length, we note that in the plane, all points for which the angular extent of an oriented edge (a,b) relative to a viewpoint is greater than some minimum epsilon are contained in the circle that has (a,b) as a chord, and an interior angle of epsilon (Figure b). The same construction holds in 3D for the epsilon surface, which can be visualized by sweeping the circle about the edge in the half-space above the plane of the triangle to form a half torus with negative interior radius (Figure c). To determine the motion bounds for edge length, we calculate the minimum distance d_e to the perimeter of this surface.

For minimum edge-vertex separation, the same calculations described in the previous paragraph are performed, but the perpendicular e_p dropped from the vertex to edge (a,b) is used as input instead of edge (a,b) (Figure d).



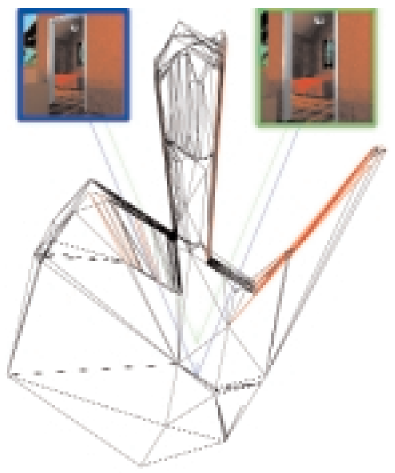
Results

We have utilized this update technique to evolve a fully dynamic display mesh used as a front end for ray-tracing.[2] We further optimize the identification of non-conforming triangles by binning the mesh triangles based on the conservative motion bounds. In this way, only a small number of triangles need to be examined each frame, and an even smaller number tested. During an interactive session with a moving observer, and a mesh with approximately 22K triangles, on average only nine percent of the triangles had to be examined, and of those only four percent needed to be removed. The end result is that on average over 99 percent of the mesh could be re-used across view motions, while still maintaining the depth-mesh and minimum feature size constraints.

To ensure a robust mesh and good-quality image reconstruction, the Delaunay condition relative to the new view must also be re-asserted for each triangle. This requires that each sample be re-projected relative to the new viewpoint. As these operations are expensive, in this application we have relaxed the Delaunay constraint for efficiency. Instead, samples are re-projected, and the Delaunay condition tested, only when triangles are encountered in some subsequent mesh operation. The derivation of a similar motion bound for triangles based on the Delaunay condition would greatly improve the performance and mesh quality after motion.

References

1. Simmons, M. & Séquin, C. H. (1999). Tapestry: A dynamic mesh-based display representation for interactive rendering. *Proceedings of Eurographics Rendering Workshop*, 1999.
2. Simmons, M. (2001). Tapestry: An efficient mesh-based display representation for interactive rendering. PhD thesis. University of California, Berkeley, May 2001.



Tapestry Evolving Mesh: A top-down view showing the mesh. The current view frustum is drawn in blue and the next desired view frustum in green. The red mesh edges show the triangles that would invalidate the constraints in the new view. The corresponding views (at higher resolution) for the blue and green frusta are shown on the above left and right, respectively.

Elastic Object Manipulation Using Coarse-to-Fine Representation of Mass-Spring Models

Contact
Hirofumi Kawai
Nara Institute of Science and Technology
(Presently with Network & Software Technology Center, SONY Corporation)
kawai@arch.sony.co.jp

Masatoshi Matsumiya
Naokazu Yokoya
Nara Institute of Science and Technology

Haruo Takemura
Nara Institute of Science and Technology
(Presently with Cybermedia Center, Osaka University)

This sketch describes an efficient method for elastic-object deformation using mass-spring models with coarse-to-fine representation based on the degree of deformation. One of the key technologies for creating realistic virtual environments is real-time manipulation of elastic objects with force feedback. Mass-spring models are widely used for representing elastic objects in virtual environments because they can be applied to both geometrical and topological deformation of objects in real time.[1] However, the increased number of mass points and springs representing more complex deformation usually makes it difficult to simulate the deformation in real time. The proposed method aims to reduce the computational cost for simulating such a complex deformation with mass-spring models.

Coarse-to-Fine Representation of Mass-Spring Models in Manipulation

The key idea of our method is based on the dynamic combination of different levels of mass-spring models according to the magnitude of object deformation in order to reduce the number of mass points and springs used for calculation. As shown in Figure 1, a largely deformed part of an object is represented and calculated by a fine mass-spring model. The other part is calculated using a coarse model. The velocity and position of mass points are calculated from these dynamic mass-spring models.

The proposed method requires building "hierarchical mass-spring models" as described in Figure 1. In this step, multiple mass-spring models are constructed with different numbers of cells. Note that a cell is composed of 2^3 connected mass points.

In interactive manipulation, the simulation is carried out as follows. First, a set of mass points for the calculation is chosen with the following criteria. When a manipulator contacts a mass point, the cell including the point is subdivided as in octree subdivision. When a diagonal spring of a cell stretches more than a certain threshold, a finer model is applied to the cell. By these steps, we get a minimum set of mass points and the level of these points in hierarchical mass-spring models. Secondly, we choose the springs connected to those points from hierarchical mass-spring models. Then the velocity and position of those points are calculated by using the chosen springs. Finally, the velocity and position of mass points that are not updated yet in each cell are calculated by linear interpolation of known values. In terms of topological deformation, a conventional method of deleting a spring that is stretched over a certain threshold in the finest mass-spring models is used.[2]

Results

In order to evaluate the effectiveness of the method, a simple elastic object manipulation system is developed with a force-feedback device on a SGI Octane (MIPS R10000 195MHz.). Figure 2 shows the results of elastic-object deformation. Figure 2 (b) and (c) illustrate coarse-to-fine models of a cubic object. Figure 2 (d) and (e) show examples of geometrical and topological deformations, respectively. Table 1 summarizes the number of springs and the time used for calculation of the models Figure 2 (b)-(e). The computation time is reduced 33.6-64.0 percent of conventional mass-spring models.

References
1. Miyazaki, S., Ueno, J., Yasuda, T., Yokoi, S., & Toriwaki, J. (1995). A study of virtual manipulation of elastic objects with destruction. IEEE Int. Workshop on Robot and Human Communication,.26-31.
2. Norton, A., Turk, G., Bacon, B., Gert, J., & Sweeney, P. (1991). Animation of fracture by physical modeling. *Visual Computer* 7, 210-219.

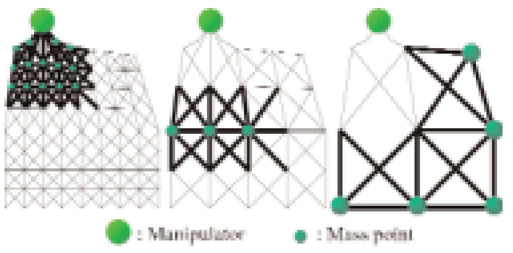


Figure 1. 2D illustration of coarse-to-fine representation of a mass-spring model.

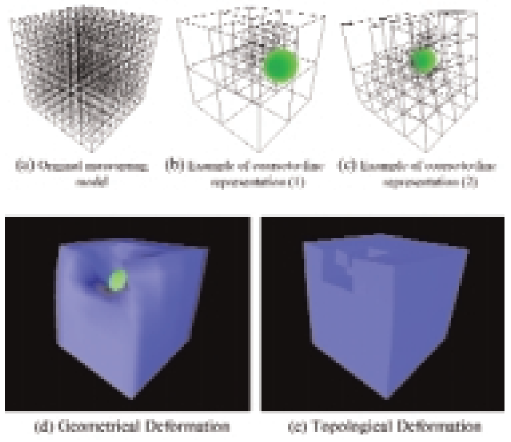

(a) Original mass-spring model
(b) Example of coarse-to-fine representation (1)
(c) Example of coarse-to-fine representation (2)
(d) Geometrical Deformation
(e) Topological Deformation

Figure 2. Elastic object deformations.

		Fig.2 (b)	Fig.2 (c)	Fig.2 (d)	Fig.2 (e)
Proposed Method	No. Springs	937	1753	1168	1170
	Time [msec]	4.5	8.3	5.3	5.5
Conventional Method	No. Springs	7448			
	Time [msec]	12.5			

Table 1. Computation time comparison with conventional method.

ELMO: A Head-Mounted Display for Real-Time Image Synthesis

Contact
Kiyoshi Kiyokawa
Communications Research Laboratory
kiyo@crl.go.jp
www.crl.go.jp/jt/a114/kiyo

Hiroyuki Ohno
Communications Research Laboratory

Yoshinori Kurata
Topcon Co

Display devices for mixed-reality systems are expected to produce synthesized images of high quality in real time. We have been developing displays that correct occlusion phenomena between real and virtual images, which is important to make synthesized images convincing.

In see-through optical displays, it is hard to represent mutual occlusion phenomena since virtual objects become transparent due to a half mirror, and the real image is always visible through the virtual image (ghost problem). However, we chose an optical approach because it preserves the intrinsic quality of real images, while the video see-through approach inevitably degrades the spatial and temporal resolution of real images.

We attacked the ghost problem with a novel optical system that has two convex lenses of the same focal length and an erecting prism, thereby making a kind of telescope of one magnification.[1] Then, by locating an LCD panel between the lenses, any portion of the real image can be blocked without defocus by closing respective pixels on the panel. We named the display ELMO (an Enhanced optical see-through display using an LCD panel for Mutual Occlusion).

Design and Features of ELMO

The latest prototype display, ELMO-3 (shown in Figure 1), is stereoscopic with folded optical paths and a real-time depth-sensing mechanism. We chose a camera-based passive approach[2] to retain two of ELMO's advantages: it does not affect the real environment, and it does not rely on any environmental presumptions.

Five cameras are used for image capturing with a focal length of 6mm and a baseline of 79mm. A high-speed FZ930 stereovision board from Komatsu Co. is used for disparity calculation, which produces 30 depth maps of 280x200 pixels per second with up to 30 disparities. For instance, if the number of disparities to be calculated are set from 0 to 29 pixels, detectable distance is from 831mm to infinity. At the minimum detectable distance, depth resolution is 5.5mm. Captured depth maps are then rendered three dimensionally and transparently to overwrite Z-buffer, from the left and right eyes' positions, to match the respective viewpoints' real images.

ELMO-3 has both an LCD panel for mask pattern and display modules for color virtual images in its body. Their images are optically combined by beam-splitters, whose transparencies are adjusted so that the real scene is bright.

Examples

Figure 2 shows four images made by and seen through ELMO-3. Conventional optical see-through displays present a combined image like Figure 2(A). By using the light blocking mechanism, virtual objects are made opaque, as in Figure 2(B). By not partially rendering virtual objects by using depth information of real objects, real objects are seen as if they occlude virtual objects: Figure 2(C). Yet virtual objects remain semitransparent. ELMO-3 can show an image like Figure 2(D) in real time, with mutual occlusion phenomena correctly processed and presented within unknown or dynamic environments.

Conclusion and Future Work

Our latest optical see-through HMD, ELMO-3, can present mixed-reality environments with correct mutual occlusion phenomena without any setups such as lighting conditions and trackers for real objects. Though the current display has a few problems (bulky body, slow response, and low resolution of the LCD panel) many of them can be improved with better LCD modules that have recently become commercially available.

References
1. Kiyokawa, K., Kurata, Y., & Ohno, H. (2000). An optical see-through display for mixed reality. *SIGGRAPH 2000 Conference Abstracts and Applications*, 241.
2. Kanade, T., Yoshida, A., Oda, K., Kano, H. & Tanaka, M. (1996). A stereo machine for video-rate dense depth mapping and its new applications. *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR) 1996*, 196-202.

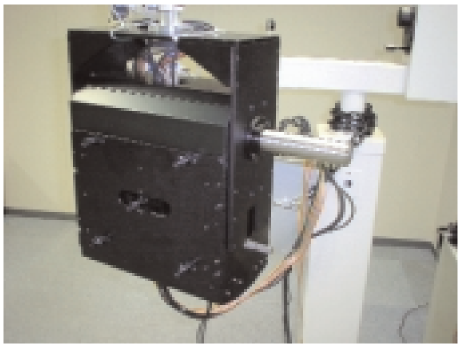


Figure 1. ELMO-3 with arm support.

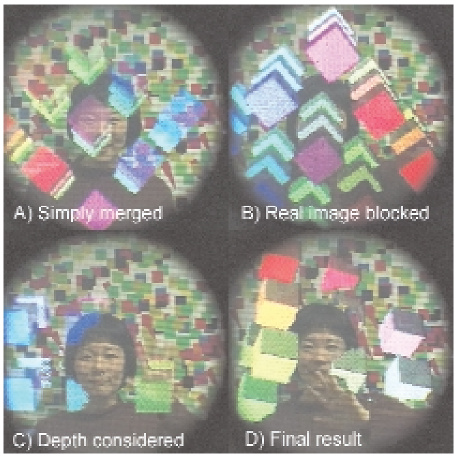


Figure 2. Four images seen through ELMO-3.

Elmo's World: Digital Puppetry on "Sesame Street"

Contact
Emre Yilmaz
PO Box 460022
San Francisco, California 94146
USA
emre@digitalpuppetry.com

Collaborators
Dave Satin
Les Rudner
SMA Video

Eric Gregory
Protozoa

Sesame Workshop

Recently, we received an unusual but exciting request: to make a drawer bark like a dog, to make a TV set cry like a baby, and to do it all live on set with Elmo.

Elmo's World is a daily 15-minute show-within-a-show on "Sesame Street." In his imaginary world, the popular red Muppet monster learns about a new concept each day (for instance, dogs, babies, or bananas). Among Elmo's friends in this world are four pieces of furniture. We bring these furniture creatures to life using real-time computer animation, or "digital puppetry." Specifically, we use an experimental motion-capture method.

The Need

As we worked live on set, our furniture characters were improvising their performances alongside the real puppets. Sesame Street Executive Producer Arlene Sherman further requested that the furniture not only be able to do basic functions, like opening and closing a drawer, but also act like living, thinking beings with personalities and quirks. The idea is that Elmo has drawn these creatures, but they have come to life, and they are not always cooperative. The window shade may be more interested in tickling Elmo than in opening up for him. The TV set may try to come to Elmo, but waddle and fall over like a baby unsuccessfully taking its first steps.

Digital Puppetry

The models were relatively easy to make, but finding a suitable performance method was difficult. We decided to try an experimental method for puppeteering the characters, using foam rubber blocks outfitted with magnetic motion-capture sensors. If a puppeteer twisted one foam rubber block, the table would twist. If they hopped the foam up and down across the floor, the door character would hop up and down across Elmo's floor. We can't use the captured motion quite straight; we do a lot of real-time modifications on the data to fit the character's proportions and position in the room. In addition to the foam rubber blocks and motion sensors, we provided foot pedals, joysticks, and sliders so the puppeteers could augment the performances. For instance, one foot pedal opened and closed the drawer. That way, one puppeteer could both make the drawer walk around and make its "mouth" open and close.

We felt this would be a very puppeteer-friendly approach, because it allows intuitive control and a great range of motion. The puppeteers could actually grab and manipulate the drawer, the TV set, and the door directly, just as if they were real puppets, watching the live video feedback to see how their characters were moving.

Production

During taping, the digital puppeteers watch their characters interacting live onscreen with Elmo. The camera sees Elmo in front of the real set, and a quick live composite allows the puppeteers and the director to see how the furniture will look in this shot. What they are seeing is very close to the final result. The puppeteering performance data are captured from the foam rubber blocks and other devices for later playback. Keeping the motion data, and waiting till post-production to commit it to videotape, allowed fixes as necessary.

Three of Sesame Street's regular Muppet performers enact the puppet furniture. The real Muppet performers are highly accomplished at bringing character to inanimate objects. They are also very experienced at interacting with Elmo! Many of the best interactions are improvised live on set. We wanted to enable them to bring all their experience and talent, including improvisation, to the digital characters as seamlessly as possible.

The Empathic Visualisation Algorithm: Chernoff Faces Revisited

Contact
Andreas Loizides
Department of Computer Science
University College London
A.Loizides@cs.ucl.ac.uk

Mel Slater
Department of Computer Science
University College London

These four pictures of faces represent the financial state of four different companies. In which one would you invest? Without knowing any details of what the facial expressions represent, it is very likely that you would choose the company represented by the bottom-right face as a good investment. In fact, more information can be gleaned from the faces by knowing that the degree of happiness represents profitability, the degree of fear represents liquidity, and so on.

The empathic visualization algorithm (EVA) is a fundamental extension of the type of data visualisation first introduced b y Chernoff,[1] who exploited the idea that people are hardwired to understand faces, and therefore can very quickly understand information encoded into facial features. In particular, it is very easy to cluster Chernoff faces into groups that represent similarities in the underlying data they represent.

Given an nxk data matrix of n observations on k variables, the original Chernoff method assigned each variable to correspond to a particular facial feature like shape of the nose or shape of the eyes. The mapping from data to visual structure was arbitrary, and the resulting face had no correspondence to the underlying semantics of the data. Such faces are good for understanding pattern, but any individual face seen in isolation does not readily convey anything about the data without knowledge of the specific mapping used.

EVA provides an automatic mapping from semantically important features of the data to emotionally or perceptually significant features of the corresponding visual structure, such as a face. In other words, a single glance at the visual structure informs the observer of the global state of the data, since the visual structure has an emotional impact on the observer that is designed to correspond to the impact that would have been generated had the observer been able to analyse the underlying data itself. Finer details concerning interpretation of the visual structure are then available through knowledge of the relationships between semantically important features of the data and emotionally significant aspects of the visual structure.

The Method

It is assumed that an nxk data matrix is to be represented by a visual structure, ideally one that is naturalistic, in the sense that humans can immediately and transparently interpret the meaning of this structure at a high level. A human face is our paradigmatic example, and we will stay with this example from now on. There are a number of global characteristics that can be used to describe the emotional expressions of a face, such as its degree of happiness, sadness, calmness, fear, or anger. There are also a number of features of a face, such as muscle tensions, that have been used in the Park and Waters model[2] to determine the overall facial expression.

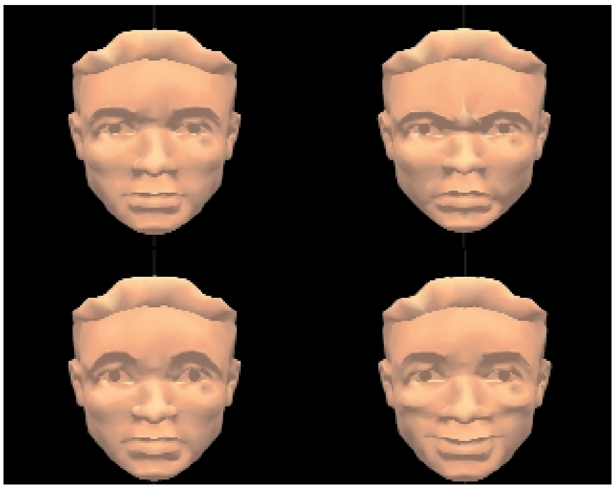
Correspondingly, there will be a number of important global characteristics that are of importance to the consumer of the data. For example, the magnitude of a particular combination of variables (representing, for example, company performance), the difference between a variable and a threshold value, and so on. These global characteristics of the data correspond to the global characteristics of the visual structure. Finally, features of the data (particular combinations of the variables) will determine the features of the visual structure. Thus, if all these features of the data were known, all the features of the visual structure would be known, and the visual structure could be rendered. The global characteristics of such a rendered visual structure could then be measured.

A genetic program is used to determine features of the data so as to minimise a fitness function, which measures the difference between the global characteristics of the data and the corresponding global characteristics of the rendered visual structure. The goal is to minimise the difference, so that the ideal visual structure is one in which its global characteristics correspond exactly to the global characteristics of the underlying data. This can easily be achieved by choosing random functions over the set of k variables to form the first generation of features from the data. Successive generations are formed in the usual way by measuring the fitness of each rendered face and then selecting feature functions with probability proportional to fitness. In our experience, the genetic program typically converges after 75 generations. Initial experiments have shown that even non-expert users can use our technique to quickly interpret the significance of the data.

Summary

We have introduced a new method for constructing an automatic mapping from data to visual structure, which enforces a homomorphism between important characteristics of the data and the emotional or perceptual impact of the visual structure. Such visual structures are informative “at a glance” but can also reveal important detailed information about the data.

References
1. Chernoff, H. (1971). The use of faces to represent points in n-dimensional space graphically. RN NR-042-993, Department of Statistics, Stanford University, December 1971.
2. Waters, K. & Parke, F. (1987). A muscle model for animating 3-dimensional facial expression. *SIGGRAPH 87 Conference Proceedings, Computer Graphics, 21* (4), 17-24.



Enhanced Reality: A New Frontier for Computer Entertainment

Contact
Richard Marks
Sony Computer
Entertainment America
richard_marks@playstation.sony.com

This sketch presents techniques for implementing enhanced reality, in which live video and computer graphics are combined to produce real-time, movie-like special effects in the home. The range of achievable effects is limitless, anywhere from something as simple as causing a participant's eyes to glow to something as complex as enabling interaction with an artificial character (for example, Stuart Little). The ability to create such effects enables a new form of computer entertainment that is both more personalized and more engaging than traditional computer entertainment (video games).

Related Work

Other researchers have considered the interactive modification of video with graphics; this topic of research is commonly referred to as augmented reality. Our work is targeted specifically at home computer entertainment, for use in a typical living room or family room. Whereas many other researchers have chosen to control the environment to achieve a desired visual effect (such as the Mandala system), we have instead chosen to equip the participant with simple props when necessary. By designing the props appropriately, we are able to deal with widely varying lighting conditions and unstructured backgrounds.

Techniques

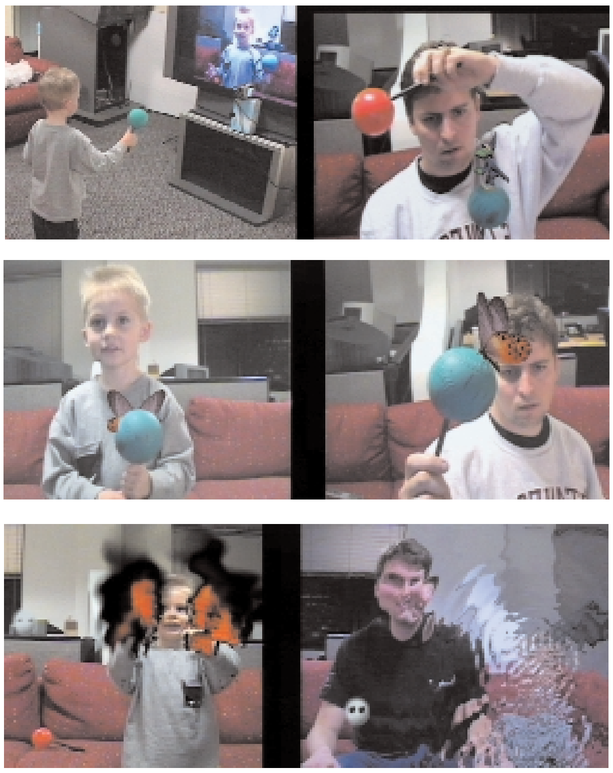
The techniques used to achieve enhanced reality fall into two categories: interpretation and enhancement. Interpretation consists of processing the video input to extract information about the participants and their environment. The type of information that is extracted depends entirely upon the particular application.
We use several techniques to obtain the following information: a per-pixel labeling that distinguishes users from their environment, a quantitative measure of the visual motion in the video stream, the camera-relative 3D location of special props, an estimated equation for the ground plane, and a model of the lighting of the scene.

Enhancement consists of modifying the video image to include the desired effect. Several rendering and compositing techniques make use of the information acquired via interpretation. These include: rendering to the z-buffer to assign depth cues to parts of the video stream, rendering synthetic objects using the dynamically obtained lighting model of the environment, and rendering synthetic objects to an off-screen buffer and alpha-feathering the edges.

We present several examples of effective ways that information obtained from interpretation can be used effectively for enhancement. We also describe many other possibilities of what may be achievable in the near future.

Implementation

We have implemented our work on a PlayStation 2 connected to a standard NTSC television set for display and an inexpensive (<$100) IEEE 1394 Webcam for video input. The props are plastic and/or foam toys.


Participants interact with a virtual character…or a virtual pet.

Explicit Control of Topological Evolution in 3D Mesh Morphing

Contact
Shigeo Takahashi
Department of Graphics
and Computer Science
Graduate School of
Arts and Sciences
University of Tokyo
3-8-1 Komaba, Meguro-ku
Tokyo 153-8902 Japan
takahashis@acm.org

Kokojima Yoshiyuki
Toshiba Corp.

Ryutarou Ohbuchi
Yamanashi University

Existing 3D mesh-morphing techniques are often limited to cases in which 3D source meshes to be morphed are topologically equivalent. Previously, we published a 3D mesh-morphing scheme based on interpolation of 3D source meshes by using a 4D tetrahedral mesh.[1] While the algorithm could potentially morph source meshes of different topological types, it lacked an effective way to explicitly control evolution of topology. This sketch presents a new approach to explicitly specifying a path of topological evolution while morphing 3D meshes of different topological types. The formalism we employed, while concise, is expressive enough to precisely specify all the possible topological changes that could occur during such topology-altering mesh morphing.

The Formalism

Our shape-morphing algorithm directly interpolates 3D meshes by using a 4D tetrahedral mesh (a discretized version of 4D hyper-surface), which is embedded in 4D space spanned by x, y, z and t(time)-axis.[1] If a topological evolution occurs during a morph, it occurs at a critical point of the 4D hyper-surface that interpolates the source 3D surfaces. According to Fomenko, et al.,[2] any such topological evolution can be invoked by attaching one of four types of topological handles, listed in Figure 1, to the 3D surfaces involved. In the figure, the shaded part of each handle is "glued" to an existing surface and then eliminated. Careful examination of 4D hyper-surface (and their embedding in 3D space) shows that all the possible transitions in topological evolution are listed in Figure 2.

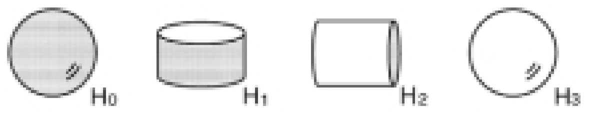


Figure 1. Four topological handles.

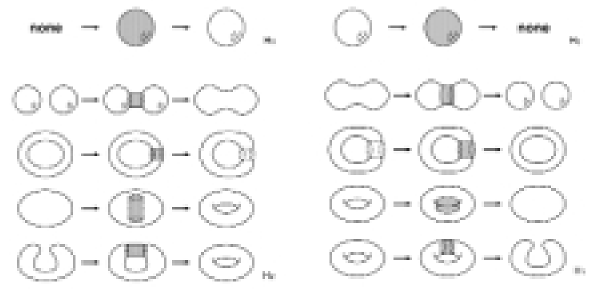


Figure 2. List of possible topological transitions.

The Mechanism

To explicitly specify a topological transition, we insert a topological "keyframe," a concrete version of one of the topological transitions above. A keyframe is a special mesh that consists of two meshes that are geometrically identical but topologically different. Two topologically different meshes in a keyframe relate 3D source meshes of different topological types. Obviously, it is not possible to prepare an infinite number of keyframes for every combination of complex source meshes and topological transitions. Instead, we prepare keyframes in their simplest forms.

As stated above, our algorithm morphs 3D source meshes by interpolating them with a 4D tetrahedral mesh.[1] To simplify the task of creating the interpolator 4D tetrahedral mesh, our algorithm starts from a set of simplified source meshes to create a coarse tetrahedral mesh. The tetrahedral mesh and source meshes are then refined so that the original details are regained in the source meshes. Taking advantage of this framework, we prepare and insert the key frames only at the coarsest of the resolution levels.

As an example, Figure 3 shows creation of two "faces" of a keyframe that relates a simplified torus and a simplified sphere. In the top figure, the hole of a torus is shrunk to a single critical point to make one "face" of a keyframe topologically equivalent to a torus. In the bottom figure, another "face" of the key frame, a mesh topologically equivalent to a sphere, is created by plugging the hole with a topological handle H_1.

Interpolating the keyframe with a pair of (simplified) source meshes of different topological types creates a 4D tetrahedral mesh as shown in Figure 4. Mesh refinement and 3D shape extraction create smooth shape-morphing sequences as shown in Figure 5.

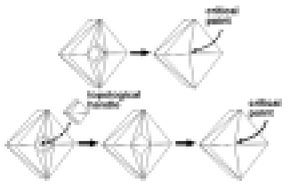


Figure 3. Keyframe generation.

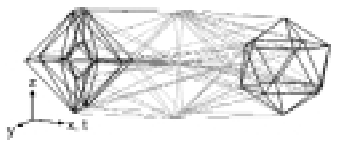


Figure 4. Tetrahedral mesh. (The t and x axes are overlaid.)

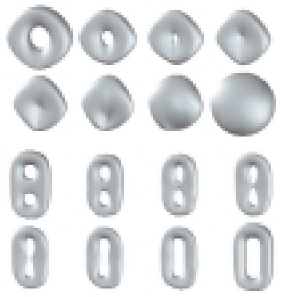


Figure 5. Results.

References
1. Ohbuchi, R., Kokojima, Y., & Takahashi, S. (2001). Blending shapes by using subdivision surfaces. *Computers and Graphics, 25* (1), 41-58.
2. Fomenko, A.T. & Kunii, T. L. (1997). *Topological modeling for visualization*. Springer-Verlag, 105-125.

Fair and Robust Curve Interpolation on the Sphere

Carlo H. Séquin
Jane Yen
EECS, CS Science Division
University of California,
Berkeley

In this sketch, we present a fair and robust interpolating scheme for curves on the sphere. This work was motivated by the desire to create fair curves on the sphere through a sparse set of data points, as might be used in smooth camera motions around an object of interest or in artistic designs. Ideally, the curves should exhibit fairness properties similar to the MVC[1] but would be generated without the need for costly optimization loops. We attempt to obtain such results with a generalization of the classical local four-point subdivision scheme.[2] In order to avoid sharp hairpin turns and cusps, and to obtain a more loopy characteristic of the curves, we forego affine invariance and aim for circular arcs wherever they are compatible with the given data points. Thus, rather than using the traditional cubic polynomial to calculate the position of the new subdivision point, we use a blend of circular arcs (as in Szilvasi-Nagy and Vendel[3]). These circle-splines or C-splines also yield pleasing interpolating curves in the plane and in 3-space.

Construction

We first describe the C-spline construction in the plane, which extends to the sphere. Given four consecutive data points A-B-C-D, a new subdivision point S halfway between B and C is computed. If A-B-C-D lie on a circle, S is placed onto the same circle at the appropriate intersection with the perpendicular bisector between points B and C. Otherwise, we construct two separate circles through A-B-C and B-C-D, respectively, and blend the two subdivision-points SL and SR proposed by the two arcs (Figure 1a). However, rather than linearly interpolating between SL and SR, we determine S by averaging the turning angles ?L and ?R generated at the points SL and SR (Figure 1b). This interpolating scheme is suggested by a local application of a discrete MVC functional, which aims at averaging the turning angles at consecutive vertices of a piecewise linear curve. To adapt this idea to the sphere, we bisect the angle between the planes through B, C, SL' and through B, C, SR', where SL' and SR', are projections of the arc midpoints onto a concentric sphere enlarged by d/2 = (B-C)/2. With this construct, we achieve that, as the distance BC becomes small compared to the radius of the sphere, the spherical construction transitions seamlessly into the planar construction.

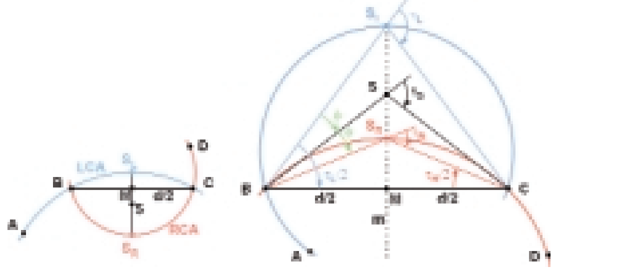


Figure 1. Subdivision of non-cocircular points by linear averaging (left) and by averaging turning angles (right).

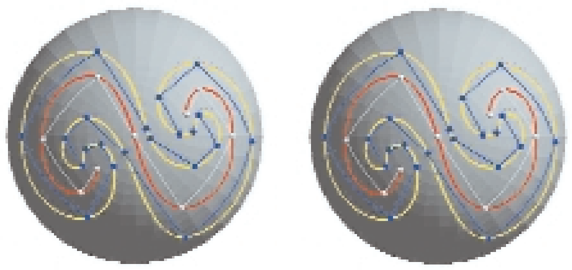


Figure 2. Circle splines on the sphere (left). Sculpture model (FDM) made with a sweep along a spherical C-spline (right).

Properties of C-Splines

The resulting curves have local support and exhibit linear and circular precision by construction. They preserve all symmetries exhibited by the original set of points. The curve is not dependent on evaluation order (unlike quaternion splines[3]), and it thus exhibits “front-to-back” symmetry.

Our construction averages the turning angles at subsequent points, so, on a local scale, it tries to minimize variations in curvature. This property gives it the desired similarity with the MVC. The globally optimized MVC has the undesirable property that it runs away to infinity when constrained by too few points. In our context, dynamic run-away is not a problem, since every subdivision point, once it has been placed, will stay there forever.

Discussion

The construction on the sphere also gives us an implicit technique to generate C-splines in 3D space. For each set of four consecutive points, an interpolating sphere is calculated, and the above construction is used to find the new subdivision point. The continuity properties for these curves are not yet fully understood. However, in many experiments with a variety of tricky test cases, C-splines have produced fair-looking, seemingly G2-continuous curves. C-splines on the sphere have been used to generate artistic designs that can be enjoyed in virtual form, and which also have been made real through rapid prototyping on a fused deposition modeling (FDM) machine (Figure 2b).

References

1. Moreton, H.P. & Séquin, C.H. (1992). Functional optimization for fair surface design. *Proceedings ACM SIGGRAPH 92*, 167-176.
2. Dyn, N., Gregory, J., & Levin, D. (1987). A four-point interpolatory subdivision scheme for curve design. *CAGD 4*, 257-268.
3. Szilvasi-Nagy, M. & Vendel, T.P. (2000). Generating curves and swept surfaces by blended circles. *CAGD 17*, 197-206.
4. Kim, M.J., Kim, M.S., & Shin, S.Y. (1995). A general construction scheme for unit quaternion curves with simple high order derivatives. *Proceedings ACM SIGGRAPH 95*, 369-376.

A Fast Simulating System for Realistic Motion and Realistic Appearance of Fluids

Contact
Yukio Watanabe
System LSI R&D Center
Toshiba Corporation
580-1 Horikawa-cho
Saiwai-ku
Kawasaki 212-8520 Japan
watanabe@sdel.toshiba.co.jp

Atsushi Kunimatsu
Takahiro Saito
Kazuhiro Hiwada
System LSI R&D Center
Toshiba Corporation

Hiroko Fujii
Tsunemi Takahashi
Corporate R&D Center
Toshiba Corporation

Heihachi Ueki
CAE Technology Department
Toshiba CAE Systems Inc.

This sketch presents a system for fast generation of animations that include realistic motion of fluids that obey physical laws and feature shadow and caustics due to the refraction of light at the water's surface. With this system, we can generate fluid animations easily and quickly.

Fluid Modeling System

Modeling of a scene that includes fluids, and simulation and rendering of the scene, are combined in one system, and they can be controlled through a GUI. In the modeling section, fluids, fluid inlets and outlets, and solid objects are placed. Each object can have its own initial velocity and other parameters, and this information affects the entire simulation. Though fluid simulation and its visualization are very time-consuming processes,[1] since our system can change the LODs of the simulation and the rendering easily by changing the number of grids of the simulation space and changing the level of division of the water surface polygons, the outline of the simulation and the visualization result can be checked very fast. Then we can move to detailed simulation and visualization. Also, we can change the lighting and the viewpoint for each frame while the simulation is running.

Fast Simulation for Realistic Motion

For realistic motion of fluids, including union and separation, 3D, time-dependent full Navier-Stokes equations have to be solved. To achieve a fast fluid simulation, we adopted two strategies:

1. Use a simulator which itself is fast. For this purpose, we have adopted the Eularian method with a uniform mesh structure. Since this method uses a fixed grid, the calculation efficiency is high. In addition, uniform mesh can speed up the process of solving the Poisson equation for pressure, which requires most of the computation power.

2. Reduce the size of the simulation space by using a low-resolution mesh. However, this also reduces the realism of the motion and appearance of the fluid. Hence, we have adopted the cubic interpolated propagation (CIP) method,[2] which has third-order precision for the fluid transportation problem. In addition, since the CIP method uses the information gradient of volume of fluid (VOF) values, we can reuse the information for the purpose of generating the iso-surfaces, which leads to a smoother surface. Moreover, to generate much smoother surfaces, the Catmull-Clark method[3] was used as a surface subdivision algorithm.

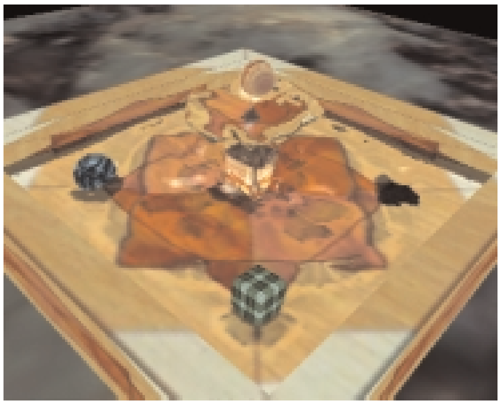
Fast Rendering for Realistic Appearance

Refractions of light on the water surface yield an aesthetically pleasing pattern of caustics, and refractions of eye vectors create effects such that objects appear to be dancing in the water. These effects are very impressive and important for realistic appearance of fluids. To calculate caustics patterns, two-pass algorithms are often used. In the first pass, distribution of the rays from the light sources is calculated. Then, in the second pass, the scene including caustics is rendered using the distribution information. In earlier methods, these calculations do not match existing rendering hardware and are time-consuming if implemented by software. Hence, we have developed a new rendering method that can utilize the existing polygon rendering hardware. For this purpose, the distribution of the rays is stored as textures (caustics textures) of each object in the scene. First of all, memory for the caustics textures is prepared and cleared to zero. For each water surface polygon, refraction of each vertex is calculated, and then the refracted polygon is drawn to the appropriate caustics texture region by using the alpha-blending function of the rendering hardware. When the water surface is rendered, by calculating the refractions of eye vectors and texture coordinates of objects, refracted eye vector intersections are given. The ordinary texture and the caustics texture of the objects are drawn on the water surface by using the multi-texture function of rendering hardware or the multi-pass approach.

Example and Evaluation

The figure shows one frame of an animation generated by our system. A fountain was designed with the modeler, and the scene was divided into 40x40x18 mesh for the simulation. Interactions between the fluid and solids are also calculated in the simulator. The caustics pattern due to the refraction of light can be seen on the bottom of the fountain. Even though the simulation space is relatively low, the resulting water surfaces are very smooth and highly realistic. Due to the strategies described above, the beginning 50 frame average simulation and rendering times were only 2.64 and 11.9 seconds, respectively, on a 933MHz PentiumIII PC with 512MB of memory and GeForce2 GTS DDR SGRAM.

References
1. Turner, J. A. & Mazzone, A.C. (1999). Multifluid finite volume Navier-Stokes solutions for realistic fluid animation. *SIGGRAPH 99 Conference Abstracts and Applications*, 259.
2. Yabe, T. (1997). Universal solver CIP for solid, liquid and gas. *Computational Fluid Dynamics Review 1997.* Eds. M. M. Hafez and K. Oshima. John Wiley & Sons, New York.
3. Catmull, E. & Clark, J. (1978). Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design, 10* (6), 350-355.


A fluid dynamics simulation and rendering result by our system.

Fast-moving Water in DreamWorks' "Spirit"

Contact
Saty Raghavachary
DreamWorks Feature Animation
1000 Flower Street
Glendale, California 91214 USA
saty@anim.dreamworks.com

Yancy Lindquist
DreamWorks Feature Animation

Contemporary animated feature films often employ a mixture of hand-drawn and 3D computer-generated elements to synthesize water imagery. DreamWorks Feature Animation's first two features, "The Prince of Egypt" and "The Road to El Dorado," both contain ocean water scenes created in this manner. The surfaces were generated procedurally using time-varying turbulence functions with appropriate controls for wave shapes and sizes.[1] In "El Dorado," a program called Spryticle layered foam shapes and handdrawn splashes to add visual complexity and integrate the elements with the rest of the scenes.[2]

For our third feature, "Spirit: Stallion of the Cimarron," a different kind of water surface is called for. Three-dimensional Effects Supervisor Wendy Rogers became responsible for designing the look of water flowing down a hill, rushing past piles of rocks. It was clear to us that the usual wave-based approach would not work in this situation.

Technique

First, we tried a rather novel technique based on scattered interpolation in two dimensions. A Maya plug-in, *SurfIt*, would interpolate the heights of a set of Maya particles along a regular grid and use the results to deform a NURBS surface whose projection lies on the grid. This did not give us enough control over the look of our surface, so we switched to an alternative technique:

A NURBS surface is sculpted to serve as a "snapshot" of the water surface. At a given frame at each CV of the NURBS surface, another deformer plug-in, *Pokemagnet*, converts the distance to each particle to a "strength," which is then used to find a "pull" on the CV. The pull can be positive or negative, depending on whether the particle is above or below the CV. The largest positive pull and largest negative pull are found by iterating through all the particles. These two values are finally combined with their own user-specified scale factors into a single resultant height offset which is then applied to the CV. This technique yields very precise control over the water surface.

We also came up with an efficient technique to animate particles (for foam and spray) along the generated surface. We sample the surface uniformly along its parametric u and v directions, and store the world-space position, normal, tangentU and tangentV values in custom attributes in a particle "cage." Now, using a particle system, we simply need to search this cage for the sampled surface point closest to each particle. At the first frame, the entire cage needs to be searched for each particle's closest neighbor. Once identified, the cage indices of each closest neighbor are stored in the particle itself, in additional channels. For subsequent frames, we start our search at the previously close neighbor and search only a small neighborhood in u and v, as specified by the user. By using a sampled version of the surface and further restricting the search to a small neighborhood on it, we are able to tackle rather complex surfaces and a large number of particles. The statistics about the closest surface neighbors are written back into the particle object in additional (user-defined) attributes, making it possible for the animator to utilize them in particle expressions to generate arbitrarily complex effects involving particle motion with respect to the surface.

The generated surfaces are environment mapped and rendered with appropriate RenderMan shaders (for example, refraction) to provide the necessary look. Additional foam and splash elements are generated with Spryticle. These elements work to further enhance the look and help blend the result with the rest of the imagery.

References
1. Yates, G. & Raghavachary, S. dwNoise and dwNoiseOffset, Houdini & Maya plug-ins, DreamWorks Feature Animation.
2. Ikeler, D. (2000). The use of Spryticle in the visual FX for "The Road to El Dorado." *SIGGRAPH 2000 Conference Abstracts and Applications*.

Feature-Based Topological Mesh Metamorphosis

Contact
Seungyong Lee
Department of Computer Science and Engineering
Pohang University of Science and Technology
leesy@postech.ac.kr
www.postech.ac.kr/~leesy

Minsu Ahn
Department of Computer Science and Engineering
Pohang University of Science and Technology

Metamorphosis, commonly referred to as morphing, deals with fluid transformation from one object to another. Three-dimensional mesh morphing handles two input polyhedral objects and generates an animation in which the source mesh gradually changes to the target through the in-between meshes.

Previous polygonal mesh-morphing techniques consist of two steps: correspondence establishment and geometry interpolation. In the first step, the vertices and edges of the source and target meshes are embedded onto a common domain, such as a sphere, a 2D polygon, and a base mesh. Next, a metamesh is created, where the vertex set contains the source and target mesh vertices, and the intersection points of the edges from the source and target meshes. In the interpolation step, an in-between mesh is generated by interpolating the vertices in the metamesh between the source and target positions.

The basic and common idea of previous mesh-morphing techniques can be summarized as the construction and interpolation of a metamesh. However, this approach has fundamental limitations. First, the metamesh usually has a more complicated structure than the source and target meshes. That is, the number of vertices, edges, and faces in the metamesh are much larger (usually five to 10 times) than those of the source and target meshes. Second, in a metamorphosis of two objects, we expect that all the attributes of the objects will be gradually transformed from one to the other. Hence, in mesh morphing, the topology and geometry of input meshes should change at the same time. However, a metamesh-based approach does not interpolate the topology of input meshes at all. The topology of an in-between mesh is always the same as that of the metamesh. Only the vertex positions are transformed from the source to target meshes.

This sketch presents a novel approach for 3D mesh morphing that is not based on a metamesh and overcomes the limitations of previous work. The approach simultaneously interpolates the topology and geometry of input meshes. With our approach, an in-between mesh contains only the vertices from the source and target meshes. Since no additional vertices are introduced, the number of vertices in an in-between mesh does not exceed the sum of those in the input meshes. Hence, the in-between meshes are much simpler than those generated by previous techniques.

The contributions of this sketch can be specified as follows:

Topological Mesh Transformation Algorithm
We present an algorithm that transforms the topology of a mesh into that of the other when two homeomorphic meshes are given. Hoppe et al. mentioned that two homeomorphic meshes can be transformed to each other by applying a sequence of three edge transformations: edge collapse, edge split, and edge swap.[1] However, they did not provide an algorithm to derive the sequence of edge transformations that realizes the topological transformation. This sketch presents the required algorithm, in which edge transformations are applied to create or remove edges in the process of establishing one-to-one mapping between the edges of two meshes.

Feature-Based Hierarchical Algorithm
To control the shape of in-between meshes, a user is allowed to specify the corresponding feature vertices on the source and target meshes. Then the vertex correspondences should be preserved in the topology transformation from the source to target mesh. Further, the topology transformation should uniformly happen in the regions between the feature vertices. We present a hierarchical algorithm based on mesh simplification for uniform topology transformation that reflects the feature vertex correspondences.

Topology and Geometry Interpolation Algorithm
Once the edge transformation sequence from the source to target mesh has been obtained, we can derive an animation by applying geomorphs[2] to the edge transformations. However, in this case, the topology and geometry do not change uniformly all over an in-between mesh, but only near the transformed edges. We present an algorithm that produces uniform topology and geometry changes over an in-between mesh. The algorithm rearranges the edge transformations into groups and performs the transformations in each group at the same time.

Figures 1 and 2 show morphing examples from our preliminary implementation of the proposed algorithms. In each example, an in-between mesh simultaneously interpolates the topology and geometry of input meshes by using edge transformations and geomorphs. Note that no additional vertices, other than from the source and target meshes, are included in an in-between mesh because the proposed approach does not construct a metamesh.

References
1. Hoppe, H., DeRose, T., Dunchamp, T., McDonald, J., & Stuetzle, W. (1993). Mesh optimization. *Proceedings of SIGGRAPH 93*, 19-26.
2. Hoppe, H. (1996). Progressive meshes. *Proceedings of SIGGRAPH 96*, 99-108.

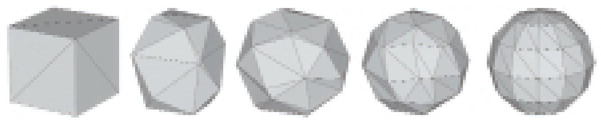


Figure 1. Morph from a cube to a sphere.

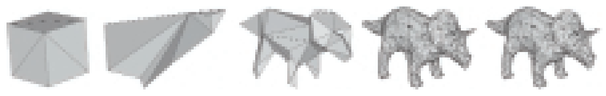


Figure 2. Morph from a cube to a triceratop.

A Flexible Approach To 3D Reconstruction From Single Images

Contact
S.F. El-Hakim
Visual Information Technology
National Research Council
Ottawa, Ontario K1A 0R6
Canada
www.vit.iit.nrc.ca/elhakim/3dmodels.html

Traditional image-based 3D reconstruction methods use multiple images to extract 3D geometry. However, it is not always possible to obtain such images – for example, when reconstructing destroyed structures using existing photographs or paintings with proper perspective (Figure 1) and reconstructing objects without actually visiting the site, using images from the Web or postcards (Figure 2). Even when multiple images are possible, parts of the scene appear in only one image due to occlusions and/or lack of features that match between images. Methods for 3D reconstruction from a single image do exist.[1,2] We present a new more accurate, more flexible method that can model a wider variety of sites and structures than existing methods. Using this approach, we reconstructed in 3D many destroyed structures using old photographs and paintings. Sites all over the world have been reconstructed from tourist pictures, Web pages, and postcards.

The approach does not need models of the objects nor known internal camera calibration.[1] It also does not use vanishing lines or vanishing points,[2] which may be unavailable or hard to extract. We use several types of constraints: point/coordinate constraints, surface constraints, and topological constraints. We solve first for internal and external camera parameters using one set of constraints, then use additional constraints to obtain 3D coordinates for reconstruction. The camera parameters and 3D coordinates are computed from photogrammetric bundle adjustment: a simultaneous triangulation of all data. Each point p, extracted from an image i, has two image coordinates, xp and yp and contributes two equations (representing the condition that the projection center, image point, and object point all fall on a straight line):

$$x_p = F_x(f_o, x_o, y_o, X_p, Y_p, Z_p, X_i, Y_i, Z_i, pitch_i, yaw_i, roll_i)$$
$$y_p = F_y(f_o, x_o, y_o, X_p, Y_p, Z_p, X_i, Y_i, Z_i, pitch_i, yaw_i, roll_i)$$

The parameters are the internal camera parameters (focal length fo, and principal point x_o, y_o), 3D object coordinates of point p (Xp, Yp, , and the camera position and orientation. Those six camera parameters are the same for all points measured in the same image. However, each point adds three unknown XYZ coordinates. The image coordinates xp and yp may include parameters for distortion. In the calibration phase, certain constraints are used. Points with the same X coordinates, points with the same Y coordinates, points with the same Z coordinates, one point with zero coordinates to define the origin of the object coordinate system, and one point with a zero Y and Z to define the orientation. Arbitrary distance is assigned between two points to define an arbitrary scale. When sufficient constraint equations are combined with equations,[1] solution of the internal and external camera parameters is possible. In the econstruction phase, more constraints are combined with equations.[1] These include shapes such as planes, cylinders, quadrics, spheres, and circles in addition to topological relationships such as perpendicularity, parallelism, and symmetry.

References
1. Debevec, P.E., Taylor, C.J., & Malik, J. (1996). Modeling and rendering architecture from photographs: A hybrid geometry and image-based approach. *Proceedings of SIGGRAPH 96*, 11-20.
2. Criminisi, A. & Zisserman, A. (2000). Single view metrology. *IJCV2000, 40* (2), 123-148.

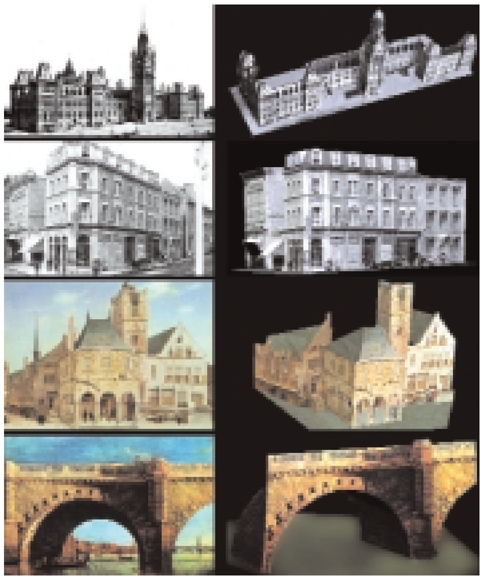


Figure 1. 3D Reconstruction from old photos and paintings.

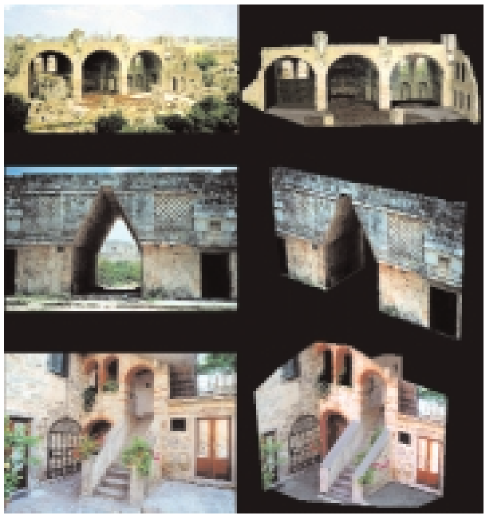


Figure 2. Examples of 3D reconstruction from tourist pictures.

Flow Noise

Contact
Ken Perlin
New York University
www.mrl.nyu.edu

Fabrice Neyret
IMAGIS

Flow textures defined with shaders that use Perlin Noise can look great, but they don't "flow" right, because they lack the swirling and advection of real flow. We extend Perlin Noise so that shaders that use it can be animated over time to produce flow textures with a "swirling"quality. We also show how to visually approximate advected flow within shaders.

Rotating Gradients

The classic Perlin Noise function[1] can be described as a sum of overlapping pseudo-random "wiggle" functions. Each wiggle, centered at a different integer lattice point (i; j; k), consists of a product of a weight kernel K and a linear function (a; b; c)i;j;k. K smoothly drops off away from (i,j,k), reaching 0 in both value and gradient at unit distance. Each (a; b; c)i;j;k = a(x_i)+b(y_j)+ c(z _ j) that has a value of 0 at (i,j,k). The result of summing all these overlapping wiggle functions: noise has a characteristic random yet smooth appearance.

Our modification is to rotate all the linear vectors (a; b; c)i;j;k over time, which causes each wiggle function to rotate in place. Because all the (a; b; c) vectors were uncorrelated before the rotation, they will remain uncorrelated after the rotation, so at every moment the result will look like Perlin Noise. Yet over time, the result will impart a "swirling" quality to flow. When multiple scales of noise are summed together, we make the rotation proportional to spatial frequency (finer noise is rotated faster), which visually models real flow.

Pseudo-Advection

Beyond swirling, fluids also contain advection of small features by larger ones, such as ripples on waves. This effect tends to stretch persistent features (for example, foam), but not newly created ones (for example, billowing), to varying degrees, according to their rate of regeneration, or structure memory M.

Traditional Perlin Turbulence is an independent sum of scaled noise, where the scaled noise is bi(x) = noise(2ix)=2i, and the turbulence is tN(x) =PN i=0 bi(x). This can define a displacement texture color(x) = C(x + ItN(x)), where C is a color table and I controls amplitude. Our pseudo-advection displaces features at scale i + 1 and location x0 in the noise domain to x1 = x0 + k ti(x0), where k is the amplitude of the displacement (see below). For small displacements, this can be approximated by x1_k ti(x1), so displacement k is proportional to an amplitude I specified by the user. We can scale this by desired structure memory M, since passive structures are totally advected, when M = 1, while very active structures are instantaneously generated, thus unstretched, when M = 0. Our advected turbulence function is defined by modifying the scaled noise to: bi(x) = b(2i(x_IM ti_1(x)))=2i and using this to construct the sum tN(x) = PN i=0 bi(x).

Results

Many flow textures can be created; some can be viewed at mrl.nyu.edu/flownoise/

Reference
1. Ebert, D. et al. (1998). *Texture and modeling*. Morgan Kaufmann Publishers, July 1998.

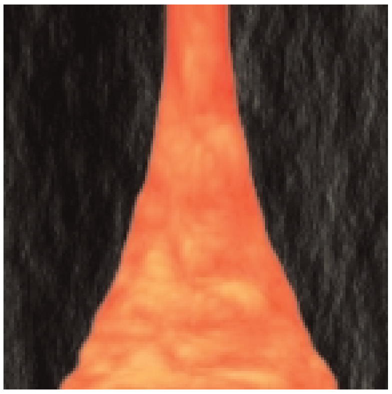


Figure 1. Lava flow.

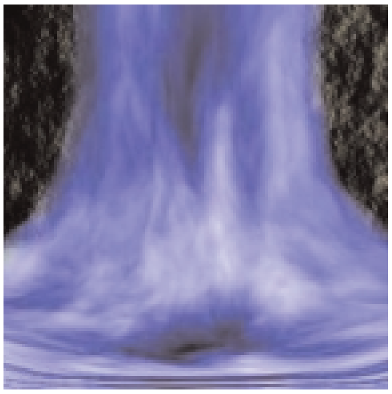


Figure 2. Waterfall.

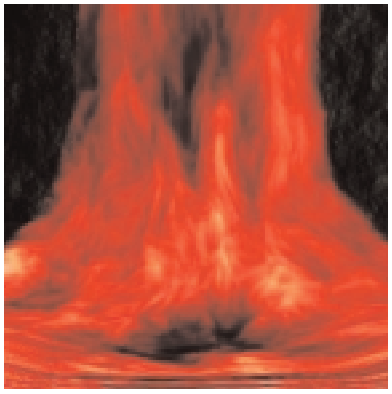


Figure 3. Waterfall with lava-flow texture.

French Surface: A New Technique for Surface Design

Contact
Zita Cheng
Human Communications Technology Laboratory
Electrical and Computer Engineering
University of British Columbia
zitast@math.ubc.ca

Sidney Fels
Human Communications Technology Laboratory
Electrical and Computer Engineering
University of British Columbia

We present a new user-friendly interface for surface design. The goal is to overcome limitations associated with traditional methods: dependence on freehand controling and rigid mathematical structures. Our solution is motivated by the drafter's tool called French curves, which are used as templates for tracing curves.

Digital French curves are, for example, investigated by Singh.[2] Our project considers its analogue in one higher dimension, namely 3D surfaces. A predrawn set of surfaces, called French surfaces, is given to the user. The selected surfaces are then connected together to form the final model. The advantage of this approach is that the only input from users is their artistic expression, not their mathematical skills. Initial testing shows that the system is easy to learn and can be quickly used to construct models. Examples using our technique are shown in Figure 1.

General Scheme

Models are contructed using a stepwise refinement process. In each step, users select a surface from the initial set of French surfaces on the control panel as a starting point for their search. Then they find the exact one they want by adjusting scale bars and action buttons. For example, users can stretch or crop a cone, extend the ring of a torus, tilt the angle of a wedge, or bend a cylinder. The selected surface is then positioned over the ongoing model and blending may be applied.

The Set of French Surfaces

The set of French surfaces must be large enough to give any shape, but not too large, to allow for easy searching. We have come up with a minimal set of primitive surfaces (for some examples, see Figure 2) ranging from those with a rounded base to those with an elongated base, and from those with a smooth top to those with a pointed top. The set also runs through surfaces of different scale, tilt, and extent. Theoretically, this set, together with the powerful blending technique, is enough to generate any surface. Our system is different from methods in which users manipulate or draw surfaces, such as Welch's approach.[3] In our system, users navigate surfaces rather than manipulate control or surface points to get the surface they want.

Blending Surfaces

Blending is vital to our system. The blend between our triangle tesselated shapes is formed by a family of Bézier curves. This is similar to Welch's method.[3] except the degree of blending is user-adjustable. Rays from the rim serve as a blending guide (Figure 2). In our system, blending preserves the original shape of both the surface and the model.

What's Next?

Our project suggests a new direction for surface modelling. We have developed a set of French surfaces that can be blended to create any surface features. The idea of using predefined, modifiable surfaces leverages the advantages of the French curves. Our technique applies to applications such as creating predefined libraries of shapes for CSG, spline models, or even freeform shapes for Teddy.[1] Future research includes improvements to the blending algorithm, more user testing, integration of our technique with various surface modelling packages, support for creating personalized libraries, and use of AI techniques to learn commonly used surfaces for easy access.

References

1. Igarashi, T., Matsuoka, S., & Tanaka, H. (1999). Teddy: A sketching interface for 3D freeform design. *Proceedings of SIGGRAPH 99*, 409-416.
2. Singh, K. (1999). Interactive curve design using digital French curves. 1999 Symposium on Interactive 3D Graphics, 23-30.
3. Welch,W. & Witkin, A. (1994). Free-form shape design using triangulated surfaces. *Proceedings of SIGGRAPH 94*, 247-256.

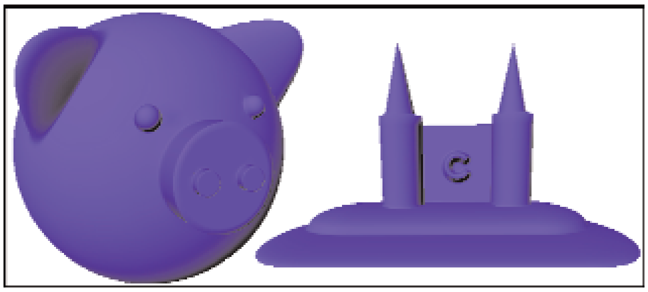


Figure 1. Models created using French Surface: the piggy and the castle.

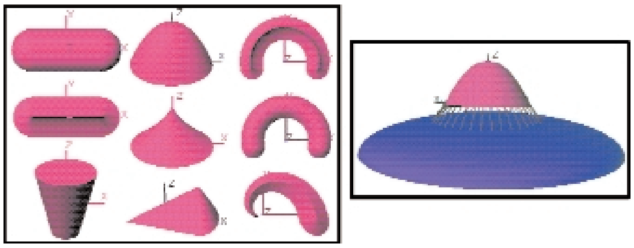


Figure 2. Examples of primitive surfaces and the blending technique.

genieBottles: An Interactive Narrative in Bottles

Contact
Ali Mazalek
Massachusetts Institute of Technology
mazalek@media.mit.edu

Ali Wood
Hiroshi Ishii
Media Laboratory
Massachusetts Institute of Technology

The genieBottles system presents a story that is told by three genies who live in glass bottles. When a bottle is opened, the genie contained inside is released and begins to talk to the user. If several genies are released at once, they converse with each other. The physical bottles can be seen as graspable "containers" and "controls" for the digital story information, and wireless tag sensing technology is used to determine their open and closed states. This interface was first used in the musicBottles project, in which sets of glass bottles were filled with musical trios.[1] The genieBottles project explores the application of the bottle interface to the interactive storytelling.

The Application

Storytelling is an important part of human culture, both in entertainment and in education. We find great pleasure in experiencing good stories, and they enable us to learn about our society and history. By creating stories, we structure our perceptions and understandings of the world in a form that can be passed on to others. Over the past 20 years, the increasing accessibility and stability of digital technology has enabled new computational approaches to storytelling. We felt that by applying a tangible interface to the field of interactive narratives, we could provide stories with a means of escaping from the computer box and into our physical environment. Our genieBottles provide an engaging interactive story experience in which the audience can go beyond the visual and auditory senses, and make better use of their sense of touch.

Story Content and Narrative Model

The genieBottles story is based on the lives of three genies (Junar, Opo, and Seala). Each has a distinct personality and background that defines the way they talk and interact with other genies. When users interact with the system, they captures the genies at a particular moment in time, during which they talk about their state of being in bottles, about their pasts, and about their expectations or desires for the future. Depending on which genie they listen to most, users will get a slightly different story tailored to that genie's particular history, desires, and beliefs.

The genieBottles use a simple state transition model for interactive storytelling, in which the system plays back the appropriate segment of audio depending on the state it is in, as well as the appropriate segment(s) of audio to transition from one state to another. State changes are caused by user interactions. For instance, if a user opens a bottle, a new genie is brought into the conversation, while if a user closes a bottle, that genie leaves the conversation. The story is organized into many short segments of text ordered according to a narrative progression. Transitioning into a new system state causes the first unused story segment for that state to be played back. This ensures that a new portion of the story will be played back even if the same sequence of interactions is repeated multiple times, allowing the story to maintain a continuous narrative progression.

Future Extensions

The genieBottles system gives a concrete example of how the use of glass bottles as an interface for digital information can be applied to interactive storytelling. In the future, we plan to explore alternative narrative models for the bottle interface. We would also like to extend the current narrative model to support different types of story content that could be used for educational purposes. For instance, a set of bottles containing important historical or political figures might be used to teach children about how peace treaties are made. Or perhaps children could fill the bottles with their own stories in order to help them learn different ways of structuring conversation-based narratives.

Acknowledgements

We would like to acknowledge the Tangible Media Group, the Responsive Environments Group, and the Physics and Media Group at the MIT Media Lab. We would also like to thank storytellers Kevin Brooks, Laura Packer, and Raelinda Woad.

Reference
1. Ishii, H., Mazalek, A., & Lee, J. (2001). Bottles as a minimal interface to access digital information. In *CHI 2001 Extended Abstracts*, ACM Press.

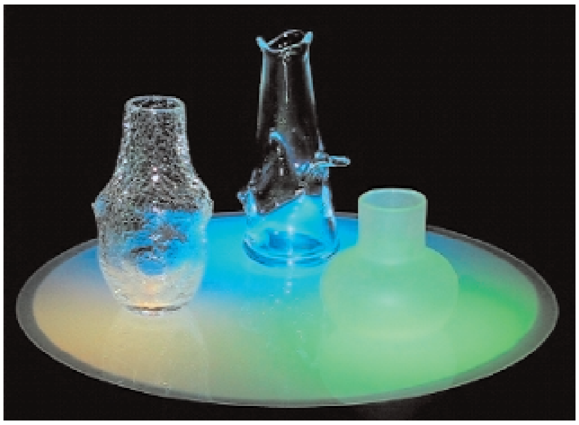


Figure 1. The three genie bottles (Junar on the left, Seala in the center, and Opo on the right).

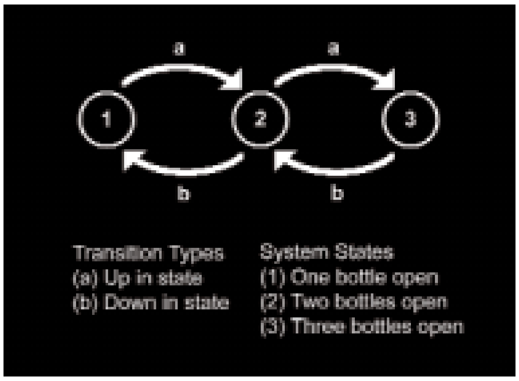


Figure 2. The two types of state transition in the genieBottles system.

Grooming and Rendering Cats and Dogs

Contact
Ivan Neulander
Rhythm & Hues Studios
ivan@rhythm.com

Pedro Huang
Hans Rijpkema
Rhythm & Hues Studios

The digital fur in "Cats and Dogs" requires fully CG animals to be seamlessly intercut with their live-action cat, dog, and mouse counterparts. Realistically groomed, animated, and rendered fur is essential to achieving this effect for various types of fur coats. The ninja cats have short sleek hair closely matted against the skin. In contrast, the villainous Mr. Tinkles is a Persian with long and fluffy plumes. Not only must these animals talk, but the furry cast must also perform martial arts and operate tools and machinery such as drills, guns, and log-loaders.

Grooming

We begin with a partition of the polygonal skin geometry. The "element normal" and "uv" parameters define a spatially and temporally consistent field of coordinate systems providing for precise local positioning and grooming control. Spline-based control hairs are rooted onto the skin surface through density maps or modified through direct modeling. An interactive 3D fur grooming tool sculpts and grooms the control hair geometry based on global and texture-based parameters such as length, scruffiness, and curvature. The same tool is used to control additional attributes of the instantiated fur. Noise controls generate the irregularity necessary for a natural appearance. Efficient clumping is achieved by correlating the hundreds of thousands of hairs using a clump-control image. Hairs are assigned to various coats such as a thick, dark undercoat, or a sparse set of specular hairs.

Animation

As geometry, the control hairs can fully utilize the transformation tools in our proprietary animation program. For example, blend shapes target specific emotions. Control is added to constrain the hairs to the surface of the deforming skin. Specialized dynamics modules are written to support concepts such as cohesion and collision using the hair root and tip connectivity.

Rendering

Each hair strand is converted by the renderer into a polygonal ribbon representing a generalized cylinder traveling along a Catmull-Rom spline. The ribbon, automatically oriented to squarely face the camera, consists of trapezoidal segments whose density depends on hair length and curvature, as well as camera distance. This arrangement minimizes the hair's polygon count while maintaining frame-to-frame coherence.

The true geometric normals of a hair ribbon are useless for shading. Instead, we use the hair's tangent vectors, which are easily obtained from its spline path. We chose the cylinder-based shading model originally presented by Kajiya and Kay. It produces realistic shading that is free of aliasing and correctly handles backlighting, the phenomenon where edge hair lights up when placed between the light and camera.

The gradual self-shadowing by layers of semi-transparent hair is essential to a photo-realistic render. To capture this, we compute a normal vector and depth value for each hair control point. These define an imaginary sphere that encloses the control point. The distance along any direction from this point to the sphere surface represents the amount of hair material that shadows the control point.

To obtain the self-shadowing normals and depth values, all hair control points are inserted into a voxel grid. A ray-marcher then determines, in each voxel, the ray that escapes the point cloud while encountering the fewest points along the way. We specially mark voxels containing skin polygons, so as to prevent rays from escaping through the skin. After computing the normal and depth values at each voxel, we interpolate them over all the control points. The resulting self-shadowing accurately portrays hair clumping and bald spots.

Growing Fitted Textures

Contact
Gabriele Gorla
gorlik@acm.org

Victoria Interrante
Guillermo Sapiro
University of Minnesota

Introduction

Adding texture to the surface of a polygonal model can profoundly enhance its visual richness. Given a texture pattern and a surface model, the historical challenge has been to determine how to apply the pattern to the surface in an appropriate manner, minimizing the visual impact of seams and projective distortion while orienting the pattern so that it flows over the shape in a desirable way.

In this sketch, we address the problem of how to seamlessly, and without repetition cover the artifacts or visible projective distortion cover the surface of a polygonally defined model with a texture pattern derived from an acquired 2D image so that the dominant orientation of the pattern will everywhere follow the surface shape in an aesthetically pleasing way. Specifically, we propose an efficient, automatic method for synthesizing, from a small sample swatch, patches of perceptually similar texture in which the pattern orientation may locally follow a specified vector field, such as the principal directions of curvature, at a per-pixel level, and in which the continuity of large and small-scale features of the pattern is generally preserved across adjacent patches

Proposed Method

The method that we propose has the advantages of being essentially automatic (requiring no manual intervention), reasonably efficient, fairly straightforward to implement, and applicable across a wide variety of texture types and models. In addition, the resulting textured objects can be easily displayed at interactive frame rates using a conventional renderer on a standard PC with texture mapping hardware.

Our technique consists of the following main steps:

- Partition the polygons of the model into contiguous patches, as nearly planar as reasonably possible.
- Compute a vector field over the object or read a pre-defined field from a file.
- Synthesize the texture pattern over each patch, using an efficient, orientation-adaptive variation of the non-parametric sampling method proposed by Efros and Leung.[1]

The constant direction field used for Figure 1 produces good results in most obvious cases. Of greater intrinsic interest to our ongoing research is the possibility of applying an oriented texture pattern to the surface of an object so that it is everywhere aligned with the principal directions of curvature. Recent results in vision research support the idea that the principal directions play an important role in surface shape understanding.

Applications and Future Work

There are many promising applications for this system and many directions for future work. One of the most interesting is multi-texturing. On a per-pixel basis it is possible to change not only the direction of the synthesized texture, but also the texture itself according to any arbitrary function. This multi-texturing method has the potential to be useful for important applications in scientific visualization. Other direction fields, such as gradient descent, hold promise for different applications, such as non-photorealistic rendering of terrain models. The methods that we have proposed can also be used for visualization of scientifically computed vector fields over surfaces.

Reference
1. Efros, A. & Leung, T. (1999).Texture synthesis by non-parametric sampling. *Proceedings International Conference on Computer Vision, 2*, 1033 -1038.

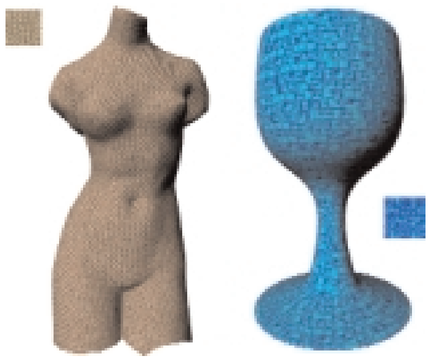


Figure 1. Examples of synthesized surface texture produced by our method. No manual intervention of any kind was employed. The textures were grown following a vector field locally defined by the projection of (0,1,0) onto the tangent plane at each point. The entire process required 12 minutes for the Venus and 20 minutes for the goblet.

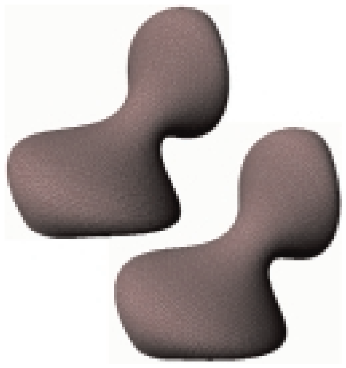


Figure 2. The orientation of a directed pattern over a curved surface can influence our perception of the surface's 3D shape. On the left, the bricks are oriented in the direction of least signed normal curvature, and on the right they are oriented in the same constant "up" direction used for the models in Figure 1.

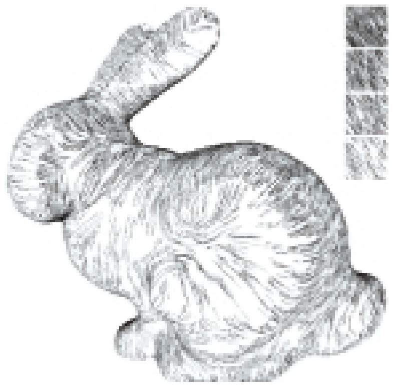


Figure 3. Multiple textures, indexed by illumination, applied to an automatically defined smooth vector field approximating the first principal directions over the Stanford bunny.

Hand-Held Torque Feedback Device

Contact
Yukio Fukui
Seiichi Nishihara
University of Tsukuba
fukui@is.tsukuba.ac.jp

Kentaro Nakata
NEC corporation

Norio Nakamura
Juli Yamashita
National Institute of Advanced Industrial Science and Technology (AIST)

Background and Problem

Portable-virtual-environment technology is a recent trend for extending and merging the virtual world into real space. The downsizing of computers and their interface devices has spurred this portability. Portable force feedback devices, in principle, must have some basis to support the reactive force. Conventional techniques in reactive force support utilize some part of the user's body such as the back or upper arm. These methods leave the sensation felt by the user incomplete or unsatisfactory because of the closed force loop.

Technical Description of Proposed System

The proposed technique requires no support to display force to the user. The key is using the angular momentum transition of rotating wheels. Changing the speed of the wheel generates torque toward the outer housing due to the law of action and reaction. Three motor-driven wheels whose axes are orthogonal to each other generate three momentum components that compound into a single momentum of arbitrary direction and magnitude. Figure 1 shows the composition of momentum and the configuration of the system implemented by the proposed technique. Figure 2 shows the developed torque feedback device: the Gyro-Cube. The table shows the correct ratio of direction that subjects felt as output torque varied by magnitude and duration. The colored region (relatively high score) shows that human sensation of torque is not always proportional to the magnitude of the stimulus.

Future Directions

We will upgrade the device by making it smaller and lighter, and implementing position sensors. It will then be applied to navigation tools for the visually handicapped or products for outdoor games.

Reference
Burdea, G. C. (1996). *Force and touch feedback for virtual reality*. John Wiley & Sons, New York.

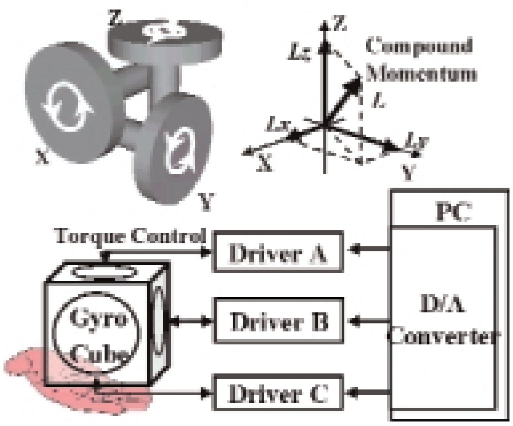


Figure 1. Torque generation of arbitrary direction.

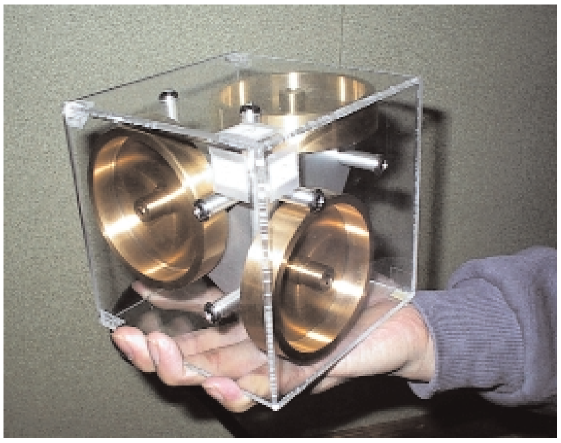
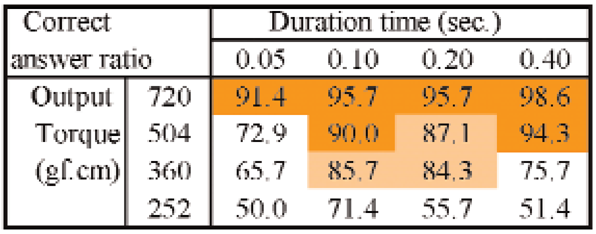


Figure 2. Hand-held torque feedback device.



Correct answer ratio		Duration time (sec.)			
		0.05	0.10	0.20	0.40
Output Torque (gf.cm)	720	91.4	95.7	95.7	98.6
	504	72.9	90.0	87.1	94.3
	360	65.7	85.7	84.3	75.7
	252	50.0	71.4	55.7	51.4

Result of psychophysical experiments.

Hardware Acceleration for Physical Modeling of Deformable Objects

Contact
Benjamin Bishop
Department of Computer Science
University of Georgia
Athens, Georgia 30602 USA
bishop@cs.uga.eud

Thomas P. Kelliher
Department of Mathematics and
Computer Science
Goucher College

Recently, there has been a great deal of interest in interactive physical modeling. Most current work has focused on rigid body simulation, since it is typically much faster than simulation of deformable objects. Interactive simulation of scenes containing complex deformable objects on consumer-grade PCs appears to be several years away if we wait for Moore's Law.

Our Work

We believe that it is possible to use specialized hardware to bring complex interactive physical modeling to the consumer in the very near future, similar to what has happened with the specialized consumer 3D card market. In order to prove this idea, we built a proof-of-concept system that involves a high-density FPGA on a custom board connected to a host machine via parallel cable. It implements forward Euler integration of the spring equations and only simple collision detection in two dimensions. It is non-pipelined and uses reduced precision functional units. An image of the board is shown in Figure 1. The architecture that was implemented in the FPGA is outlined in Figure 2.

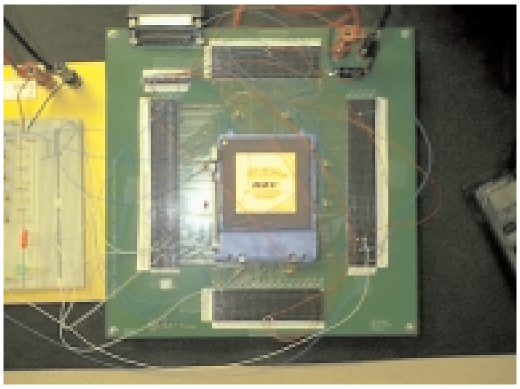


Figure 1. Custom circuit board with FPGA.

These simplifications were necessary in order to deal with the area restrictions in the FPGA. The final FPGA utilization was 79 percent of the 250K gates. The high utilization was caused by the lack of interconnect available on the FPGA, and the fact that barrel shifting in floating-point operations quickly consumes interconnect. It is well known that forward Euler integration suffers from stability problems. We chose this scheme only for its simplicity. For later generations, a more suitable (stable and simple) integration technique must be found. This scheme should operate on local data, since global dependencies significantly complicate the hardware design.

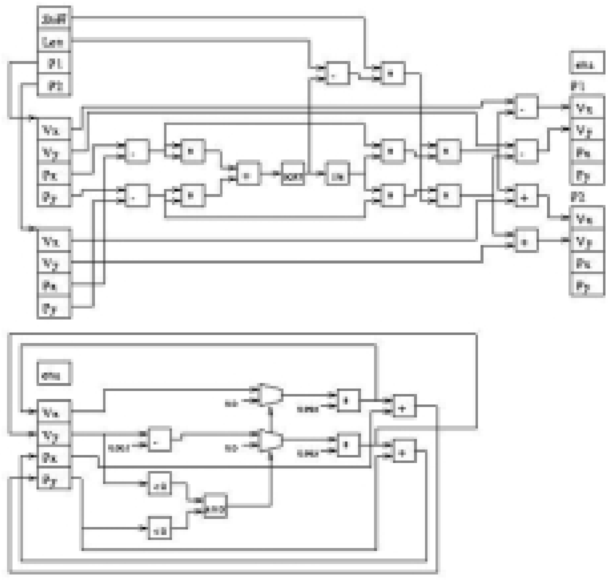


Figure 2. Architecture implemented in the FPGA. Spring solver at top, simple collision detection at bottom.

Results

Figure 3 shows an animation sequence that was generated on the board. It is a simple four-point, six-spring cube. In comparison with our host machine, we estimated the performance improvement (assuming pipelining and equal clock rates, which would be easily attainable in an ASIC) to be 92 times, so the potential gain from hardware acceleration is clear.

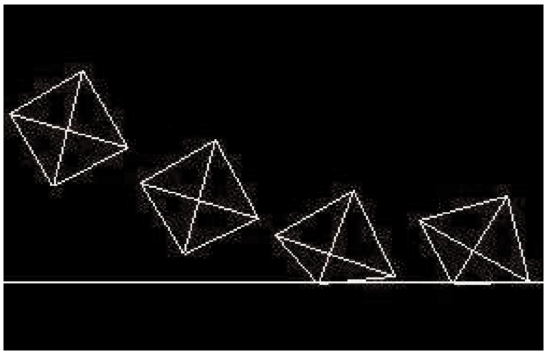


Figure 3. Animation generated in hardware.

Hardware-Accelerated High-Quality Filtering of Solid Textures

Contact
Markus Hadwiger
Helwig Hauser
VRVis Research Center, Vienna
msh@vrvis.at

Thomas Theußl
Eduard Gröller
Institute of Computer Graphics and Algorithms
Vienna University of Technology

Recent consumer graphics hardware is able to texture 2D surfaces via lookups into a 3D texture domain. This can be used for texturing a polygonal object with a solid texture,[2] for instance. However, the filtering performed by current hardware is constrained to tri-linear interpolation within the texture volume. In order to achieve high-quality reconstruction, filter kernels of higher order have to be used.[3,4]

We describe a hardware-accelerated approach for high-quality texturing of polygonal objects with a solid texture. Several passes with 3D texture mapping and multi-texturing are accumulated to yield the final result. Conceptually, arbitrary filter kernels are possible. On hardware supporting multi-texturing with 3D textures, interactive speeds can be achieved.

Hardware-Accelerated Higher-Order Filtering

The basic idea of our approach is to reorder evaluation of the filter convolution sum. Instead of calculating each output sample at a point x in its entirety, we instead distribute the contribution of a single input sample to all output sample points simultaneously. We do this for all contributing input samples, accumulating their contribution over multiple passes.

That is, instead of using

```
FOR ALL output samples $x_i$ DO
  FOR ALL deltas $r_j$ of contributing neighbors DO
    $g(x_i)$ += $f[\mathrm{trunc}(x_i) + r_j] * h(\mathrm{frac}(x_i) - r_j)$;
```

where f[] is the input signal, h() the filter kernel, g() the output signal, the x_i are the output samples at fractional locations, input samples are at the integers, and r_j is an integer in [-m+1, m], with m being half the filter width, we do

```
FOR ALL deltas $r_j$ of contributing neighbors DO
  PAR ALL output samples $x_i$ DO
    $g(x_i)$ += $\mathrm{shift}_j(f)[\mathrm{trunc}(x_i)] * h_j(\mathrm{frac}(x_i))$;
```

The inner loop is performed in parallel for all output samples (pixels) by the hardware in a single rendering pass. The outer loop is achieved through multiple passes. For the multiplication in the inner loop, we use texture mapping hardware that is capable of multi-texturing. One texture contains the current filter tile (an integer section of the filter kernel, e.g., [-1, 0], [0, 1], etc.), denoted as $h_j()$. The other contains the input texture using appropriately offset texture coordinates to provide the corresponding input values, the offset operation denoted by $\mathrm{shift}_j()$.

Figure 1 shows this method for the simplest example of a tent filter, using two passes in order to achieve linear interpolation. Note that the filter shape is not required to be linear at all.

We subdivide the filter kernel into its integer tiles and download each discretized tile as a separate texture. Exactly one tile will be used in a single pass, replicated over the entire output polygon.

This approach immediately generalizes to two and three dimensions. Conceptually, it is entirely independent of both the actual shape of the filter kernel used and its width. A greater width increases the number of passes necessary but does not change the algorithm.

There are as many rendering passes with two-texture multi-texturing as the number of input samples contributing to a single output sample. That is, a cubic kernel requires four passes in one dimension, 16 passes in two dimensions, and 64 passes in three dimensions, respectively.

Results

Figure 2 shows the vase mapped with a solid marble texture from the ATI RadeonVolumeTexture example,[1] where we have integrated our filtering approach. The cubic B-spline filter clearly exhibits significantly less interpolation artifacts than simple tri-linear interpolation.

For further information regarding hardware-accelerated filtering with arbitrary filter kernels, see:
www.vrvis.at/vis/research/hq-hw-reco/

References
1. ATI Radeon SDK. URL: www.ati.com/
2. Ebert, D., Musgrave, F., Peachey, D., Worley, S., & Perlin, K. (2000). *Texturing and modeling: A procedural approach*. Academic Press, 2000.
3. Möller, T., Machiraju, R., Müller, K., & Yagel,R. (1997). Evaluation and design of filters using a Taylor series expansion. *IEEE Transactions on Visualization and Computer Graphics, 3* (2), 184-199.
4. Theuβl, T., Hauser, H., & Gröller, E. (2000). Mastering Windows: Improving reconstruction. In *Proceedings of IEEE Symposium on Volume Visualization*, 101-108.

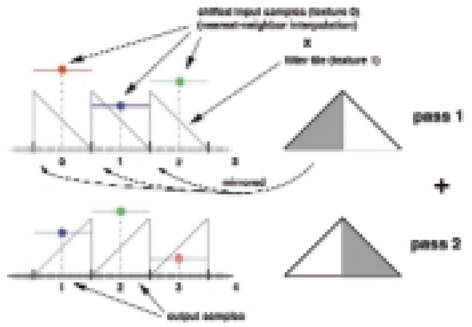


Figure 1. Both filter tiles of a tent filter replicated separately over the output sample grid for two passes. Adding up the two passes yields the final result.

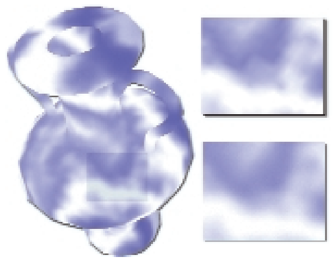


Figure 2. Vase textured with solid marble texture. The shaded area is shown magnified on the right; tri-linear interpolation on top, cubic B-spline at the bottom.

HDR Shop

Contact
Chris Tchou
USC Institute for
Creative Technologies
tchou@ict.usc.edu

Paul Debevec
USC Institute for
Creative Technologies

HDR Shop is a computer application (currently under development) designed to view and edit high-dynamic-range (HDR)[1] images: pictures that can capture a much greater range of light intensities than standard photographs or computer images. This approach is very useful for image-based lighting and post-render processing.

Photographs from traditional cameras do not record the amount of light over a certain level. All the bright points in a photo are white, which makes it impossible to detect any difference in intensity. The standard technique to acquire HDR images that capture this missing information is to take several photographs at different exposures (making each photo progressively darker, without moving the camera), until the bright lights no longer saturate. The sequence of photographs can then be analyzed to derive the light intensity of each point in the scene.

Whereas traditional image editors work with 8- or 16-bit images, HDR Shop is built from the ground up to work correctly with HDR images. All operations are done with linear floating-point numbers. In many cases, this simplifies the code, as well as providing more correct output.

For the purpose of real-time display, however, it is important to quickly convert linear floating-point images to 8-bit RGB with the appropriate gamma curve. The standard gamma formula involves an exponentiation, which is slow. In the interest of speed, we have found it useful to approximate this calculation by constructing a lookup table indexed by the most significant bits of the floating-point values. For common gamma values of 1.4 ~ 2.2, it suffices to use 16 bits (eight exponent bits and eight mantissa bits) to reduce the error below rounding error.

In addition to resampling, cropping, and mathematical operations, HDR Shop also supports transformations among most common panoramic formats, facilitating the use of HDR panoramas in image-based lighting[2]. HDR Shop can also automatically export a low-dynamic-range (LDR) copy of any image to an external image editor. Changes to the LDR image are then incorporated into the HDR image, so existing tools can be used to modify HDR images.

See also: www.debevec.org/HDRShop

References

1. Debevec, P. & Malik, J. (1997). Recovering high dynamic range radiance maps from photographs. *Proceedings of SIGGRAPH 97.*
2. Debevec, P. (1998). Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. *Proceedings of SIGGRAPH 98.*

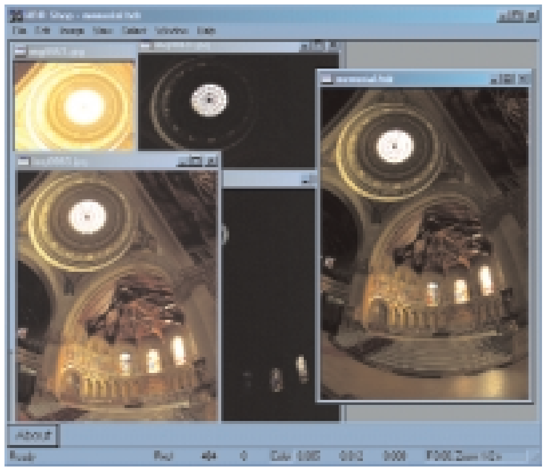


Figure 1. In HDR Shop, a sequence of low-dynamic-range images (left) can be compiled into a single high-dynamic-range image (right).

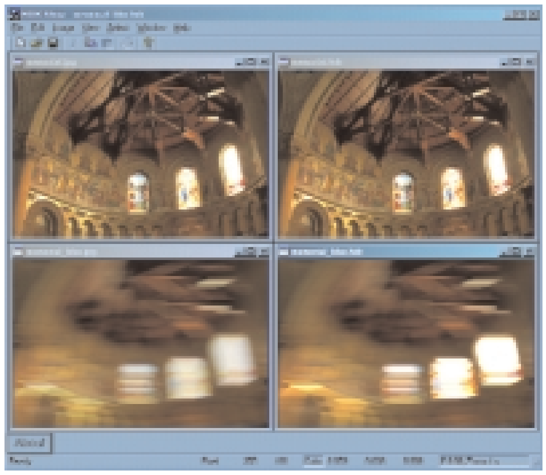


Figure 2. Comparison of HDR Shop's horizontal motion blur on a low-dynamic-range image (left) vs. a high-dynamic-range image (right).

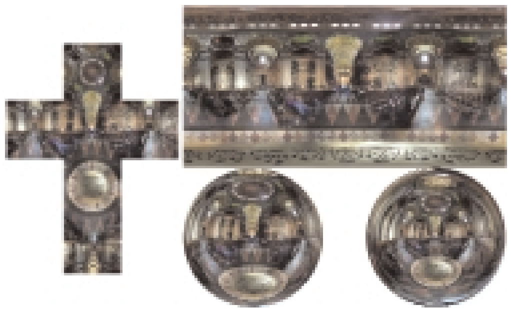


Figure 3. St. Paul's Cathedral panorama, originally in cube-map format (left), converted in HDR Shop to latitude-longitude (upper right), mirrored ball, and light probe formats (lower right).

A Head-Mounted Projective Display and its Applications in Interactive Augmented Environments

Contact
Hong Hua
Beckman Institute
University of Illinois
at Urbana-Champaign
Urbana, Illinois 61801 USA
Honghua@uiuc.edu

Contributors
Hong Hua
Leonard D. Brown
Chunyu Gao
Narendra Ahuja
Beckman Institute
University of Illinois
at Urbana-Champaign

Jannick P. Rolland
School of Optics/CREOL
University of Central Florida

Frank Biocca
Michigan State University

Technology

The concept of projective displays using retro-reflective material was initially patented by Fergason in 1997, and head-mounted projective displays (HMPDs) were proposed as an alternative to conventional eyepiece-type head-mounted displays and stereo projection systems for 3D visualization.[1] An HMPD consists of a pair of miniature projection lenses, beam splitters, and displays mounted on the head, and a supple and non-distorting retro-reflective sheeting material placed strategically in the environment.[1]

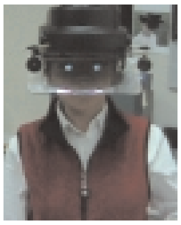


Figure 1. HMPD prototype.

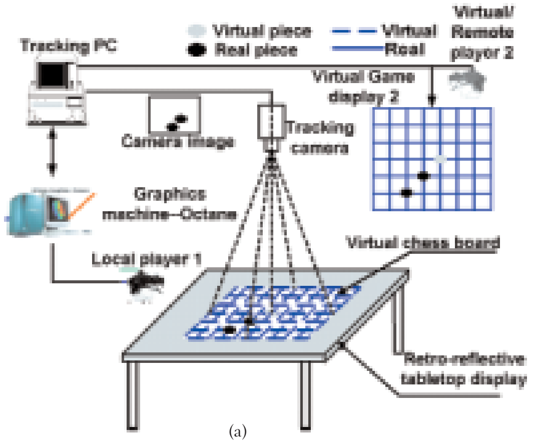
The use of projection lenses and replacement of a diffusing projection screen with a retro-reflective screen distinguish HMPDs from conventional head-mounted displays (HMDs) and stereoscopic projection-based displays. In addition to direct see-through capability, HMPD technology intrinsically provides correct occlusion of computer-generated objects by physical objects and creates ubiquitous display environments in which a retro-reflective material can be applied to any location in space and can be tailored to any shape without introducing additional distortion to the virtual images. Such a design also allows for larger field of view (FOV) and higher optical performance than eyepiece-based HMDs,[2] In multi-user collaborative environments, the retro-reflective screen makes it possible to generate unique perspectives for each user without introducing crosstalk from other participants.[2] We present our recent design of a compact head-mounted prototype implementing ultra-light custom lenses.[2] The prototype (Figure 1) achieved 50 degrees FOV and 3.96 arcmin/pixel visual resolution at a total weight of less than 700 grams.

Application in Interactive Augmented Environments

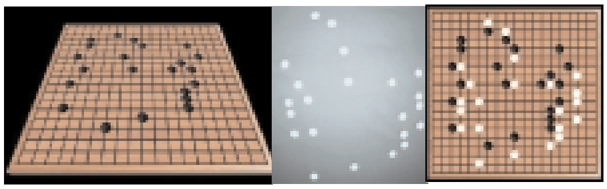
To demonstrate the various capabilities of this technology, we present two applications:

1. Play-augmented "GO" chess game with a remote opponent: In Figure 2(a), a computer-generated 3D "GO" chessboard is projected through an HMPD onto a tabletop retro-reflective screen. The local player (1), wearing the HMPD, perceives the virtual chessboard as if it were a real object on the tabletop and manipulates his real chess pieces on the virtual board. A vision-based tracking setup detects the locations of his pieces on the virtual board and transmits this information via network to the remote player. The remote player (2) uses a PC-based game interface in which all game components are visualized on a PC monitor and chess-piece manipulation is achieved via a standard mouse. When the remote player adds a piece to the board, a corresponding computer- generated piece is projected onto the HMPD user's virtual chessboard. Therefore, the HMPD player perceives the virtual chessboard, the real pieces, which correctly occlude the virtual chessboard, and the virtual pieces in a seamless augmented environment. A head-tracking system is used to maintain the correct registration of the real and virtual elements. The virtual and direct views of both players are shown in Figure 2 (b) through (d) respectively.

2. Interactive simulation of fluid flow over a simple terrain: We are also developing an augmented tool for simulation of fluid flow over simple terrain.[3] With this tool, virtual fluid is animated through physical-trough models coated in retro-reflective material. Users can dynamically alter or obstruct the flow with a variety of physical objects, which are tracked via a vision-based algorithm.



(a)



(b) (c) (d)

Figure 2. Playing a "GO" chess game with a remote opponent:
(a) setup illustration; (b) HMPD player's virtual view;
(c) HMPD player's direct real view; (d) remote player's PC-based interface.

References
1. Hua, H., Girardot, A., Gao, C., & Rolland, J. P. (2000). Engineering of head-mounted projective displays. *Applied Optics, 39* (22), 3814-3824.
2. Hua, H., Gao, C., Biocca, F., & Rolland, J.P. (2001). An ultra-light and compact design and implementation of head-mounted projective displays. *Proceedings of the IEEE Virtual Reality Annual International Symposium 2001* (In press).
3. Chen, J.X. & Da Vitoria Lobo, N. (1995). Toward interactive-rate simulation of fluids with moving obstacles using Navier-stokes equations. *Graphical Models and Image Processing, 57* (2), 107-116.

High-Dynamic-Range Photography, Global Illumination, and Mental Ray on a Commercial Budget and Timeline

Contact
Brian Goldberg
Digital Domain
300 Rose Avenue
Venice, California 90291 USA
bgold@d2.com

Dan Lemmon
Digital Domain

This sketch presents a detailed look into the application of high-dynamic-range (HDR) image-based lighting techniques and new features in rendering technology on a high-profile commercial. Two technical directors had two months to build a complete image-based lighting and global illumination pipeline into the current workflow and methodology at Digital Domain. The commercial featured detailed, synthetic characters rendered into live-action scenes using HDR lighting information and global illumination rendering. The final result is an excellent example of how these technologies came together to produce highly believable characters integrated seamlessly into a live-action environment.

Goals

The illumination pipeline was to use HDR imagery and HDR mapped, reconstructed set geometry as scene lighting information in a global illumination render. Custom software and shaders, controlled by a simple configuration methodology, enabled an easy (as seen by the user) transition from a Maya scene to a GI-rendered sequence through Mental Ray.

The Approach

The main focus of the project was to use real-world, HDR lighting information in our global illumination renders. To obtain accurate lighting data, we were required to photograph both incident lighting information and the more subtle inter-reflection effects from the live-action set. The complexity of the lighting and shadows on set dictated that we photograph each different lighting setup from multiple positions in order to capture all of the light sources directly.

Far from a point-and-shoot solution, the process of transitioning from camera image to HDR dataset involved shooting a precise range of exposures at calculated set locations. Instead of the traditional "chrome sphere" approach used in obtaining wide field-of-view environment images, we chose to use a 180-degree fish-eye lens attached to a Nikon D1 digital SLR. This enhanced the efficiency of the process, but a great deal of work was done to transform the component images into a "light probe" (a unified, HDR, 360-degree-by-360-degree view of the environment in angular map coordinate space. This work included coordinate space conversion, image stitching, and compilation of HDR images. In some cases, we altered reality by "painting" onto the HDR light probe images, which helped achieve visually appealing results.

Our CG characters were required to interact with the set and to run in and out of light and shadow – the field of view, so to speak, of different light probes. We were faced with the problems of CG characters interacting with a live action set in an image-based lighting environment. Special attention had to be paid to characters that moved from the "field of view" of one light probe to another. We used the HDR information we obtained on set not only to simulate the traditional light fixtures on set, but also to texture map our CG set, in high-dynamic range, in order to accurately capture complex inter-reflection effects from the set pieces onto the characters.

An efficient storage and retrieval mechanism for these data was required. Our HDR data had to be accessible from a multitude of custom and third-party packages, most of which were not designed to handle this sort of data. We eventually settled on a four-channel, eight-bit encoding of the floating-point HDR data, based on the radiance file format. We built the file format around the RGB file structure and called it "rgbe" to emphasize the inclusion of mantissa/exponent information. The resulting data representation allowed us to store HDR lighting data in a compact and highly versatile format.

Bringing Monte Carlo radiosity simulation to character animation work was not trivial. This technique has produced impressive results in situations like architectural lighting simulation. In these applications, surfaces are largely flat and lighting changes fairly smoothly; so sparse samples can be interpolated well. Our situation was quite different. In this project, our out-of-this-world characters were highly organic and had very detailed, bumpy, porous skin. This presented additional technical challenges, as results could not be interpolated smoothly across the surface.

In order to maintain the integrity of our floating-point HDR data through the color and lighting pipeline, we designed a Mental Ray output shader to render the characters in our custom "rgbe" file format. This enabled an enormous amount of freedom for our compositing team. To create a realistic "film look," our compositors matched the characteristic "toe-and-shoulder" response curve that is characteristic of film stock. Using a response-flitting system designed in Nuke, our in-house compositor, they selected a low-dynamic-range "slice" of the HDR image. By using a synthetic "photographic response curve," we were able to generate images that matched the plate photography with unprecedented accuracy.

How Changes in Technology Change the Use of Computer Animation at Walt Disney Feature Animation

Contact
Steve Goldberg
Walt Disney Feature Animation
steve.goldberg@disney.com

This sketch explores how advances in software and hardware have influenced changes in not only our films' aesthetics, but also our artists' attitudes towards the use of computer animation at Walt Disney Feature Animation over the last decade. As the tools get faster and more intuitive for traditionally trained artists, and the gap between "computer people" and "artists" closes, big changes are in store for the animated film medium, both in 2D and 3D. This discussion covers uses of computer animation in the departments of layout, character animation, effects animation, and background painting .

Video progression sequences from "Dinosaur," "Fantasia 2000," "Tarzan," "Hunchback of Notre Dame," "The Lion King," "Aladdin," "Beauty and the Beast," and "The Rescuers Down Under" illustrate how changes in technology have helped to change how our films are made and how they look.

"How the Grinch Stole Christmas:" The Who Construction Kit

Contact
Darren Hendler
Digital Domain
darren@d2.com

Dan Lemmon
Digital Domain

Background

Early on in our involvement with "How the Grinch Stole Christmas," we realized that we were going to be creating many, many characters. The scope of the work dictated both large-scale digital crowds and detailed foreground character animation. Traditionally at Digital Domain, all characters had been modeled, set up, enveloped, and textured on an individual basis, an approach that, while efficient for a few characters, becomes impossible for hundreds of characters. Our solution was to create a system that would enable us to manage a large volume of characters without sacrificing the unique variations of each Who.

The Who Construction Kit

The challenge was to create a system that would allow designers, animators, and art directors to create characters without any prior technical or even 3D experience. The turnaround time for generation of these characters had to be very short. The newly created characters had to be instantly animatable, fully rigged and weighted, and linked to valid shaders and textures. The system had to be scalable and allow for changes to rigging, weighting, character design, scale, and resolution throughout the production process. It also had to be robust enough to function on its own without constant support from technical directors. This system became known as the Who Construction Kit, and it was used in the creation of each and every Who, from the lowest-detail background characters to high-detail foreground characters.

Because the Who Construction Kit would be used to populate an entire village of Whos, it would have to provide an almost infinite number of character combinations created from a finite set of body sizes, face shapes, textures, and items of clothing. To accomplish this, each item of clothing had to be usable on any character irrespective of shape or size. So, in effect, as the user modified the body of the character, each item of clothing would have to be capable of automatically adjusting and deforming to the correct shape and size of the character underneath. Another difficult issue was the process of automatic character setup: rigging and weighting a character of arbitrary body shape and size with all attached clothing, collision models, and facial setup without any user input.As we began receiving footage, we realized that different sequences would require different clothing schemes. We catalogued our databases of Whos based on sequence, apparel, gender, and other factors.

Character replacement was another problem early on in the project. The more involved the director and art department became in the character-creation process, the more control they required. In the final stages of some shots, we were asked to change the clothing or weight of a particular character. The Who Construction Kit allowed us to import any given character and alter the character's size, weight, age, etc., all within the kit's interface. Once the character had been modified, it could be regenerated and substituted back into the scene within a matter of minutes.

The interface of the Who Construction Kit had to be user-friendly and intuitive so that non-technical users could generate characters quickly. As users created Whos, the construction kit updated the database Web pages with a preview-render and the name of each character. Animators could then peruse the Web pages and choose appropriate Whos for their shots from the catalogue of available characters. The kit interfaced directly with our kinematic and animation tools, allowing animators the ability to import any catalogued character at any point in the animation process. If animators could not find a character to their liking in the catalogue, they could easily create a new one. By keeping the system fast, flexible, and user-friendly, we were able to populate Whoville with hundreds of unique characters while minimizing our technical labor.

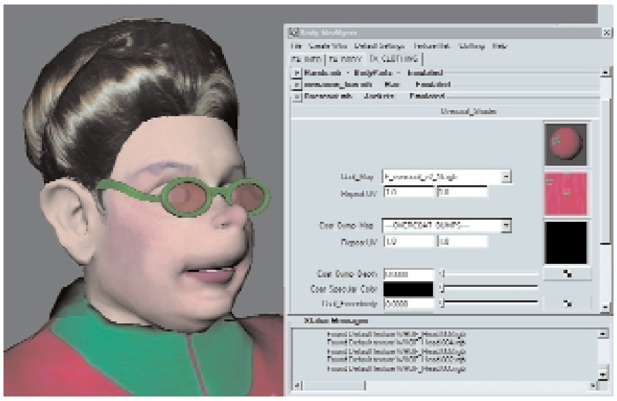


Figure 1. GUI of the Who Construction Kit.

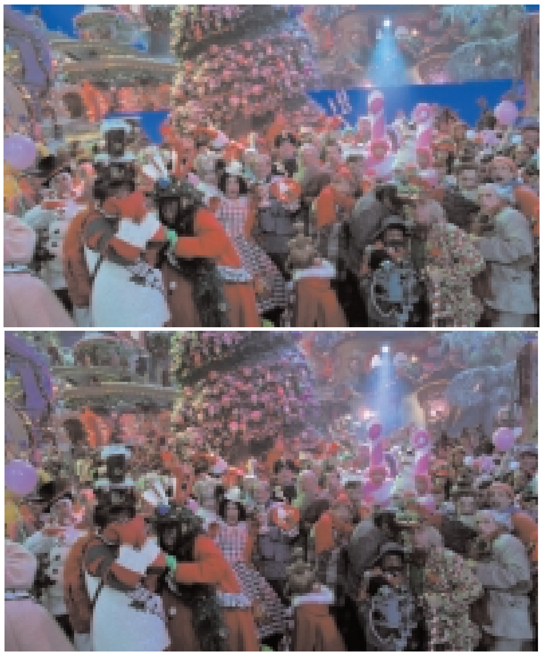


Figure 2. One of the scenes from the movie, in which the character creation kit was used to create the crowd scenes.

Human Motion Signatures for Character Animation

Contact
M. Alex O. Vasilescu
Department of Computer Science
University of Toronto
10 King's College Road
Toronto, Ontario M5S 3G4
Canada
alexv@cs.toronto.edu

Given motion-capture samples of Charlie Chaplin's walk, is it possible to synthesize other motions (say, ascending or descending stairs) in his distinctive style? More generally, in analogy with handwritten signatures, do people have characteristic motion signatures that individualize their movements? If so, can these signatures be extracted from example motions?

Human motion is the composite consequence of multiple elements – most importantly, the action performed and a motion signature. The *action* captures the person-invariant essence of an activity or movement. The *motion signature* captures the distinctive pattern of movement of any particular individual. In this sketch, we introduce an algorithm that separates these elemental effects and recombines them in novel ways for animation of graphical characters. For example, given a corpus of walking, stair-ascending, and stair-descending motion data collected over a group of subjects, plus a sample walking motion for a new subject, our algorithm can synthesize never-before-seen ascending and descending motions in the distinctive style of this new individual.

Our algorithm first decomposes a corpus of motion data into motion signatures and action components. Next, given an incomplete set of motion data for a new subject, the algorithm extracts a motion signature for this individual from the available data and the corresponding action components obtained previously. The remaining action components can then be recombined with this motion signature to synthesize a complete set of motions in the distinctive style of the new subject.

The mathematical basis of our algorithm is a statistical numerical technique known as *n-mode analysis*. The two-mode analysis algorithm that we adapt to our purposes was described for scalar observations by Magnus and Neudecker in their book *Matrix Differential Calculus* (Wiley, 1999).

Experiments

We begin by collecting a corpus of motion data spanning 10 different subjects using a Vicon motion capture system. Applying smoothing, interpolation, and IK motion-processing steps, the data are reduced to time-varying joint angles for complete cycles of three types of motions: walking, ascending stairs, and descending stairs. In a "leave-one-person-out" validation experiment, we verified that our algorithm is able to extract motion signatures and accurately synthesize all three types of motions.

Figure 1 shows a stair-ascending motion synthesized for one of the individuals. Our algorithm extracted the motion signature from a sample walk of this individual. The extracted motion signature was combined with general stair-ascending parameters to synthesize the stair-ascending motion that exhibits the characteristic signature.

Figure 2 shows frames from a short animation that was created with synthesized data. For the clown, the motion signature is that of a strutting male, and the action parameters are those for a walk. The other character was animated using the motion signature of a female and the action parameters of a walk.

Conclusion

We have introduced the concept of decomposing human motion data into motion-signature and action elements. These elements are useful in the synthesis of novel motions for animation of articulated characters.

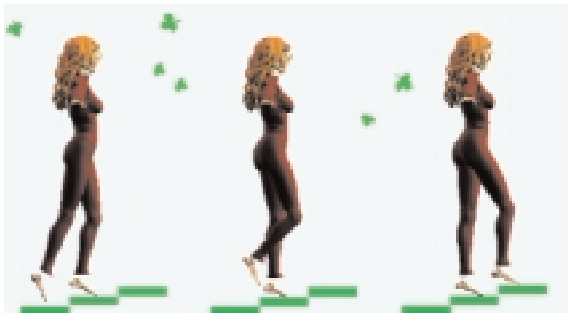


Figure 1. A synthesized stair-ascending motion.

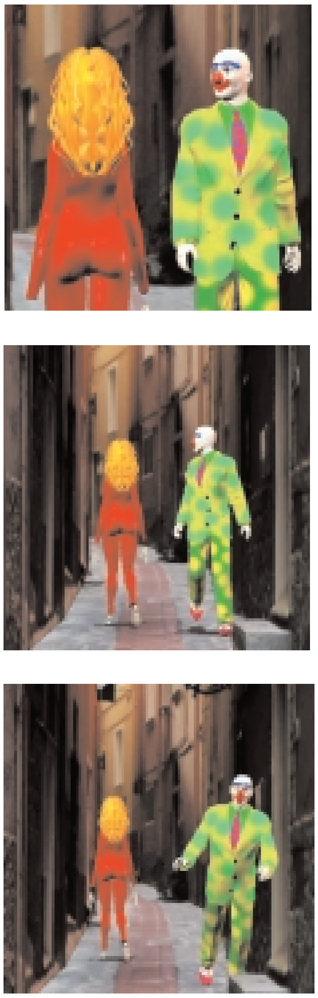


Figure 2. A short animation created using motion data synthesized by our algorithm.

Hybrid Ink-Line Rendering in a Production Environment

Contact
Rasmus Tamstorf
Walt Disney Feature Animation
500 South Buena Vista Street
Burbank, California 91521 USA
Rasmus.Tamstorf@disney.com

Ramón Montoya-Vozmediano
Daniel Teece
Patrick Dalton
Walt Disney Feature Animation

Walt Disney Feature Animation has used a number of in-house ink-line rendering solutions in the past, all of which have been image-based. Used in the production of sequences such as the Hydra in "Hercules," these post-processors were slow and had high memory demands. Inka was developed as a robust long-term alternative, and has been employed on "The Emperor's New Groove" as well as in several hundred shots on "Atlantis."

Inka takes a hybrid approach. While it is primarily geometry-based, with information from the 3D model driving placement of 2D ink lines, it also uses some image-processing methods and employs a z-buffer to perform hidden-line removal. Rendering speed is typically an order of magnitude faster than previous image-based techniques, and ink lines found during the geometry processing stage can be rendered without the artifacts inherent in pixel-based methods.

To ensure consistent, high-quality renders, the user can set attributes for specific NURBS patches or meshes in a text file. This allows the appearance of the ink-line image to be tailored to particular needs and settings to be tweaked for complex shots. These attributes may affect not only the appearance of a line (width, color, opacity) but also visibility determination and surface tessellation parameters. In this way, the user has a great deal of low-level control over each part of each individual line, which has proved invaluable in a production environment.

In addition, custom shaders allow users to control how a line is drawn based on the underlying geometry or screen position. They are currently being used extensively on upcoming features to create more stylized strokes. Another feature that has turned out to be valuable, despite its simplicity, is attenuation of opacity and line width as a function of depth. This allows a smooth transition of objects as they recede into the background.

Geometry-based ink-line rendering is very sensitive to its input geometry. In production use, models that work flawlessly with a shaded renderer can cause imperfect ink lines. Some problems can be fixed by setting attributes, whereas others require re-modeling. However, modeling for Inka also provides new opportunities. As an example, curves defined on the surface of a patch can be used to substitute for texture or to better convey the shape of an object, as if the modeler were drawing on the geometry.

As Inka has matured, its approach has proven to be scalable and flexible enough to accommodate increasing scene complexity as well as models that range from rigid and mechanical to animated and organic. It is now being used in production on "Treasure Planet" and "Lilo and Stitch."

Special thanks to Joe Lohmar, Yun-Chen Sung, and Mike King.

Image-Based Rendering and Illumination Using Spherical Mosaics

Contact
Chen Shen
EECS, Computer Science Division
University of California, Berkeley
Berkeley, California 94720-1776
USA
+1.510.642.3631
csh@cs.berkeley.edu

Heung-Yeung Shum
Microsoft Research, China

James F. O'Brien
University of California, Berkeley

Introduction

Rather than rendering using geometric primitives, image-based rendering techniques synthesize novel views directly from a collection of sample images. This approach has proven to be a powerful alternative to traditional geometry-based methods, making it possible to interactively render views of complex scenes.

The work described here extends the concentric mosaic representation developed by Shum and He[1] to spherical mosaics that allow the viewer greater freedom of movement. Additionally, by precomputing maps for diffuse and specular lighting terms, we use high-dynamic-range image data to compute realistic illumination for objects that can be interactively manipulated within the scene.

Spherical Mosaics

Concentric mosaics represent a scene using a series of sample images captured along a circular path looking outward. Spherical mosaics simply extend this approach by using a set of sample images that are taken from locations distributed over the surface of a sphere. Within an inner sphere, whose radius is determined by the sample camera's field of view, any exiting ray can be mapped to a location in one of the sample images.

The mapping is accomplished by intersecting the exiting ray with the capture sphere to determine the nearest sample cameras. A point along the ray at a constant depth is then projected back to the center of projection for each camera to determine the closest pixels within each image. Linear interpolation of the resulting values yields the value for the exiting ray. If depth estimates are available, they may be used to improve the accuracy of the projection into the sample images. Because the sample images form a 4D representation of the external light field, the virtual camera is afforded a full six degrees of freedom within the inner sphere.

Diffuse and Specular Maps

Once a scene has been sampled, adding additional objects into the environment requires realistically replicating environmental illumination when shading the new object. If the sample images have high dynamic range with pixel values that record incident radiance, then the illumination at any point within the inner sphere can be determined from the sampled data. Unfortunately, shading calculations at a point on the surface of the new object require expensive summations over all the incoming ray directions.

To achieve interactive rendering speeds, we move the summations to a preprocessing step and implement them by filtering the sampled data to form diffuse and specular maps. Both of these maps are stored as spherical mosaics. A mipmap-like structure holds multiple versions of the specular map computed with different-sized kernels that can be used for different specular falloff parameters. The diffuse map is indexed according to surface position and normal, while the specular map is indexed by surface position and the reflected viewing direction. Simple ray tracing provides a unified way to render both the spherical mosaic environment and the added objects.

Results and Future Work

We do not currently have a physical device for capturing spherical mosaics of real environments, so we have tested our methods using synthetic images generated using the RADIANCE rendering package.[2] The figures on this page (and the accompanying video) show images that were synthesized from a data set consisting of 9172 small (256 x 256) sample images acquired with a 90° field of view. The user is able to interactively change the view, add new objects, move the new objects, and modify their surface properties. A 256 x 256 anti-aliased image with exposure compensation can be re-synthesized in 1.05 seconds on an SGI 350MHz R12000. Without anti-aliasing, only .35 seconds are required.

The primary future extension of this work would be to build a physical capture device. Another area for further investigation is allowing the added objects to affect the environment lighting by casting shadows or creating reflections.

References
1. Shum, H.-Y. & He, H.-W. (1999). Rendering with concentric mosaics. *Proceedings of SIGGRAPH 99,* 299-306.
2. Ward, G. (1994). The RADIANCE lighting simulation and rendering system. *Proceedings of SIGGRAPH 94,* 459-472.

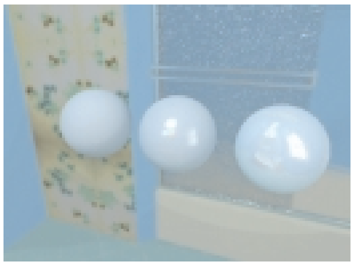


Image-based renderings showing inserted objects with different surface properties.

Image-Based Photometric Reconstruction for Mixed Reality

Contact
S. Gibson
Advanced Interfaces Group
Department of Computer Science
Manchester, United Kingdom
sg@cs.man.ac.uk
aig.cs.man.ac.uk

T.L.J. Howard
R.J. Hubbold
Advanced Interfaces Group

Image-based photometric reconstruction is the process of estimating the illumination and surface reflectance properties of an environment from a set of photographs. For mixed-reality applications, such a reconstruction is required if synthetic objects are to be correctly illuminated or if synthetic light sources are to be used to re-illuminate the scene.

Current approaches to photometric reconstruction[1,2] are limited in the situation they can be applied. The user must often provide a complete geometric model of the environment, and in some cases, the position and intensity of the light sources that are illuminating the scene. Additionally, current reconstruction algorithms are limited by the fact that they cannot be applied when a mixture of artificial and natural illumination lights the scene. This sketch shows results from applying a new reconstruction algorithm[3] to the problem of estimating the photometric properties of real scenes.

A New Approach

We use a combination of computer vision and photogrammetry algorithms to calibrate cameras and build a partial geometric model of a scene. A small number of high-dynamic-range images are then captured, and registered to the geometric model. Radiance values from these images are associated with each visible surface. A number of virtual light sources are then automatically positioned around the scene. These light sources are used to mimic the effects of unknown luminaries in the parts of the environment that have not been modelled, as well as the effect of light reflected indirectly off unknown geometry. An iterative refinement algorithm is used to estimate the intensity distributions of these virtual light sources, as well as the diffuse and specular properties of surfaces. At each stage of refinement, an optimisation process chooses light source intensities so that the illumination they cast on each surface matches the radiance values in the high-dynamic-range photographs. Further details are available in Gibson, Howard & Hubbold.[3]

Results

Figures 2 and 3 show synthetic renderings using materials and virtual light source intensities estimated for a scene containing artificial and natural light (Figure 1). These data were then used as input to a global illumination algorithm, allowing photo-realistic renderings to be obtained from novel viewpoints (Figure 4), and where the images have been augmented with synthetic light sources and artificial objects (Figures 4 and 5).

References
1. Yu, Debevec, P., Malik, J., & Hawkins, T. (1999). Inverse global illumination: Recovering reflectance models of real scenes from photographs. *Proceedings of SIGGRAPH 99.*
2. Loscos, C., Drettakis, G., & Robert, L. (2000). Interactive virtual relighting of real scenes. *IEEE Transactions on Visualization and Computer Graphics, 6* (4).
3. Gibson, S., Howard, T.L.J., & Hubbold, R.J. (2001). Flexible image-based photometric reconstruction using virtual light-sources. *Computer Graphics Forum (Proceedings of Eurographics 2001), 19* (3).

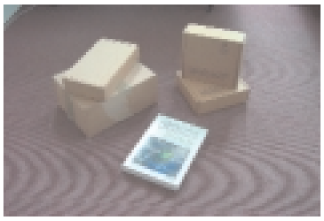


Figure 1. Original high-dynamic-range images for artificial and natural light.

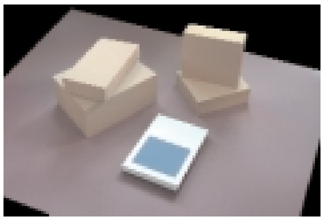


Figure 2; Synthetic renderings using the reconstructed illumination data (without texture).

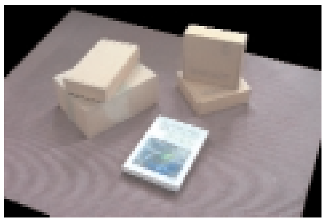


Figure 3. Synthetic renderings using the reconstructed illumination data (with texture).

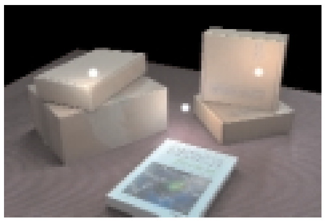


Figure 4. Renderings from a novel viewpoint and with synthetic light sources.

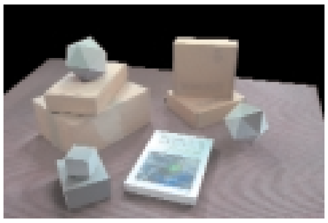


Figure 5. Rendering the scene with synthetic objects.

Image-Based Reconstruction of Shift-Variant Materials

Contact
Hendrik P. A. Lensch
Max-Planck-Institut für Informatik
lensch@mpi-sb.mpg.de

Jan Kautz
Michael Goesele
Hans-Peter Seidel
Max-Planck-Institut für Informatik

Wolfgang Heidrich
University of British Columbia

The use of realistic models for all components of image synthesis is a fundamental prerequisite for photo-realistic rendering. Manually generating these models often becomes infeasible as the demand for visual complexity steadily increases. In this sketch, we concentrate on acquisition of realistic materials. In particular, we describe an acquisition method for shift-variant BRDFs: acquiring a specific BRDF for each surface point.

Data Acquisition

We acquire the geometry of the object with a 3D scanner (for example, a structured light or computer tomography scanner) which yields a triangular mesh. In order to capture the reflection properties, we take a relatively small number (around 20) of high-dynamic-range (HDR) images of the object, lit by a point light source. We recover the camera position and orientation as well as the light-source position relative to the geometric model for all images.

For every point on the object's surface, we collect all available data from the different views in a data structure called lumitexel. It contains the position of the surface point, its normal, and a list of radiance samples together with their viewing and lighting directions.

Clustering of Materials

Since a single lumitexel does not carry enough information to reliably fit a BRDF model to the radiance samples, we first determine clusters of lumitexels belonging to similar materials. Starting with a single cluster containing all lumitexels, the parameters of an average BRDF are fitted using the Levenberg-Marquardt algorithm. From this, two new sets of parameters are generated by varying the fitted parameters along the direction of maximum variance, yielding two slightly separated BRDFs. The lumitexels of the original cluster are then assigned to the nearest of these BRDFs, forming two new clusters. A stable separation of the materials in the clusters is obtained by repeatedly fitting BRDFs to the two clusters and redistributing the original lumitexels. Further splitting isolates the different materials until the number of clusters matches the number of materials of the object as illustrated in Figure 1.

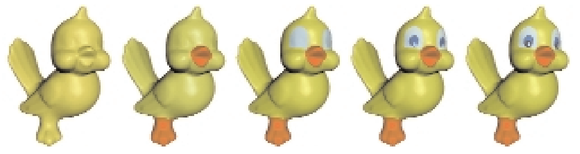


Figure 1. The clustering process at work. In every image, a new cluster was created.

Shift-Variant Behavior

After the clustering, we still have the same reflection behavior assigned to all lumitexels in one cluster. However, small features on the surface and smooth transition between materials can only be represented if every lumitexel is assigned its own BRDF. In our algorithm, this BRDF is a linear combination of the BRDFs recovered by the clustering procedure. This can be represented by a set of basis BRDFs for the entire model plus a set of weighting coefficients for each lumitexel. An optimal set of weighting coefficients minimizes the error between the measured radiance and the weighted radiance values obtained by evaluating the basis BRDFs for the specific viewing and lighting directions. To recover the coefficients we compute the least-square solution of the corresponding system of equations using singular-value decomposition. This method allows for accurately shaded, photo-realistic rendering of complex solid objects from new viewpoints under arbitrary lighting conditions with relatively small acquisition effort.

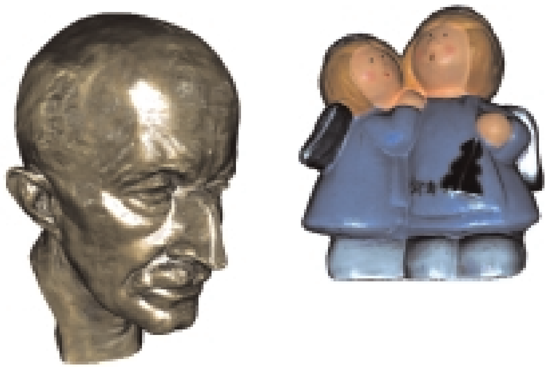


Figure 2. Two models rendered with shift-variant BRDFs acquired with our reconstruction method.

Related Work
1. Debevec, P., Hawkins, T., Tchou, C. Duiker, H.P., Sarokin, W., & Sagar, M. (2000). Acquiring the reflectance field of a human face. *Proceedings of SIGGRAPH 2000*, 145-156.
2. Wood, D., Azuma, D., Aldinger, K., Curless, B., Duchamp, T., Salesin, D., & Stuetzle, W. (2000). Surface light fields for 3D photography. *Proceedings of SIGGRAPH 2000*, 287-296.

Image-Based Rendering for Animated Deforming Objects

Contact
Hiroshi Kawasaki
Institute of Industrial Science
University of Tokyo
4--6--1 Komaba, Meguro-ku
Tokyo 153-8505 Japan
h-kawa@sak.iis.u-tokyo.ac.jp

Hiroyuki Aritaki
Takeshi Ooishi
Katsushi Ikeuchi
Masao Sakauchi
Institute of Industrial Science

This technical sketch presents a description of how an image-based rendering (IBR) technique can be used to produce photo-realistic animation of real-world objects that usually have non-rigid-surface effects (for example, animal fur and velvet).

In recent years, principles and various kinds of implementation and theoretical analyses of IBR have been proposed and published one after another. However, for practical use of IBR (for example, for animation production), little research has been done, and few actual applications have been developed. Although there may be many reasons for this, the following two reasons are significant:

1. Huge data size.
2. Lack of interactivity among objects and illumination.

IBR data volumes are very large. This is a crucial issue for actual implementation, and there have been many attempts to reduce the data volumes. On the other hand, little research has been devoted to realizing interactivity for IBR. Therefore, we leave the data-volume problem for future research and concentrate on the interactivity problem encountered in the process of making photo-realistic animations.

Our method is based on the surface-light-field technique, a term coined by Miller et al.[1] Our research is also inspired by the work done by Nishino et al.[2] and Daniel N.Wood et al.[3]

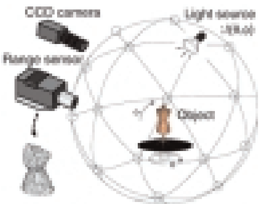
Interactivity and Animation

Basically, we assume that the interaction of the objects can be defined by three aspects: the arbitrary position of the object, including deformation of the object; the arbitrary illumination change that usually causes shadow changes; and real-time rendering.

The purpose of our system, to achieve interactivity for IBR, can be translated as rendering arbitrarily positioned objects with arbitrary deformation and illumination in real time.* However, real-time rendering is not necessary for animation, and we are not currently interested in real time.

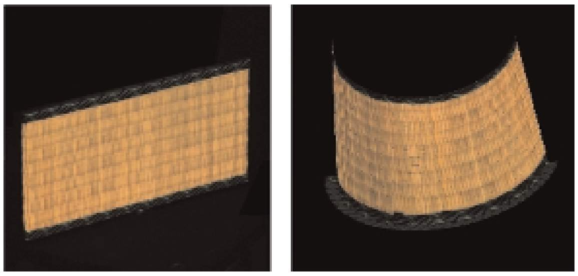
System

To synthesize an object whose position and pose change arbitrarily, we re-use the actual ray derived from the object's surface. To achieve photo-realistic rendering, we developed a mesh-based rendering algorithm that selects the appropriate ray from the whole ray based on the BRDF (bidirectional reflectance distribution function) for the individual mesh. Also, because the data acquisition process is very important for this system, we configured an original data-acquisition system: "light dome," shown in the following figure. The light dome can automatically acquire the 4D data that is necessary for image synthesis.

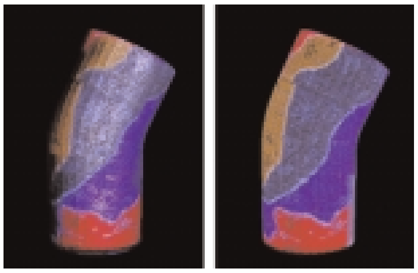


Results

We performed several experiments to show the effectiveness of our method. The following figure shows the result that was achieved by using a Tatami block.** The image on the left is the actual captured image, while the image on the right is the synthesized image after deformation.



The next figure shows another result using a paper-wrapped can; the left image is the image synthesized with our method; the right image is a texture-mapped image.



These results demonstrate the effectiveness of our proposed method to render photo-realistic images of the deformed objects with non-rigid surface effects. In the future, we will synthesize the object under arbitrary illumination changes.

References
1. Miller, G., Rubin, S., & Ponceleon, D. (1998). Lazy decompression of surface light fields for precomputed global illumination. *Rendering Techniques (Eurographics Proceedings)*, June 1998.
2. Nishino, K., Sato, Y., & Ikeuchi, K. (1999). Appearance compression and synthesis based on 3D model for mixed reality. *Proceedings of Seventh International Conference on Computer Vision*.
3. Wood, D., Azuma D., Aldinger, W., Curless, B., Duchamp, T., Salesin, D., & Steutzle, W. (2000) Surface light fields for 3D photography. *Proceedings of SIGGRAPH 2000*.

* Rendering such a deformed object with consistent illumination and geometry is always difficult.

** A 3D block made of tightly bound straw. This material also has non-rigid effects on the surface.

Imaging the Living Computer: Neuroscience Methods in Visualizing the Performance of Parallel Programs

Contact
Daniel L. Herman
DigitalFish Films
dh@digitalfish.com
www.digitalfish.com

Introduction

Neuros is a testbed system for graphical visualization of program executions on large parallel computers. It consists of an extensible, configurable toolkit that provides still-frame and animated views of parallel tracefiles. The system has several unusual strengths that set it apart from most other execution visualizers, including its ability to manipulate and display networks that have high dimensionality or that contain large numbers of processors. Neuros takes a scalable, summarizing approach to viewing trace data, which makes it suitable for work with both small-scale and large, highly complex executions.

Neuros is built on the OpenDX (formerly IBM Data Explorer) visualization system, so users familiar with this popular environment can extend the built-in views and add their own. Input is an MPICL trace file. Publicly available converters allow use of other data formats.

Neuroimaging Techniques

We focused on massively parallel (MP) programs because the problems in visualizing such programs tend to be more severe than those faced in small-scale parallelism. Numerous similarities exist between the obstacles in MP execution visualization and those faced in neurological diagnosis of brain disorders, including the large number of nodes (processors/neurons) involved, high aggregate bandwidth, difficulty in obtaining execution data without perturbing the system under study, difficulty in abstracting high-level information from single-node statistics, and difficulty in performing hardware monitoring on large numbers of individual nodes.

A number of elegant and powerful visualization solutions have been developed to aid in neurological diagnosis. Four that we focus on particularly are electroencephalography (EEG), evoked potential (EP), positron emission tomography (PET), and the magnetoencephalogram (MEG).

There are a number of ideas we can borrow from these techniques. The first is an emphasis on high-level characterizations that employ summary statistics. While it is possible for neurologists to make single-cell recordings, summary information tends to be more useful for practical diagnostics. Further, neurologists rely on visualization to study large sets of measured data. They use topographical mappings of data to some notion of the problem domain. In neuroimaging, the mappings usually reflect the physical layout of the system. In our application, such mappings could be to the logical topology of the processor network or to some abstract representation of the problem domain.

Neuros Visualization Techniques

Neuros provides tools to assist in visualizing topologies with high dimensionality and large numbers of nodes. It supports mapping data to topologies of arbitrary dimensionality, while providing various means of projecting an n-dimensional space onto a three-dimensional space for display. *Nesting* supports the representation of four-dimensional hypercubes, for example, as two nested three-dimensional cubes, one within the other. A similar projection is *staggering*, in which the second 3-cube is displaced laterally (in 3-space) from the first. A *shadow cast* computes the shadow that the n-dimensional object would make on a $k < n$ dimensional volume. When dealing with large numbers of nodes, rather than strictly representing individual processors and interconnections (for example, spheres and lines), Neuros can display summarizing abstractions for groups of nodes as clouds or isosurfaces through the machine topology. This gives an amorphous view of activity that mandates abstract reading and interpretation.

Time is represented in various ways. It may be spread out along some spatial dimension, as is done frequently for EP and EEG, or the entire visualization may be presented as distinct time slices or a time-dilated animation, typical of PET and MEG.

Any data values in the execution (including statistical observations such as average load at a node or messaging rate along an edge) can be encoded into the visualization. They may directly affect glyph properties (for example, color or scale), or they may cause more abstract effects such as general warpings of space. This allows, for instance, load to be represented by causing an n-ary mesh to bulge outward in high-load regions. Other visualizations include lighting effects and volumetric fog density.

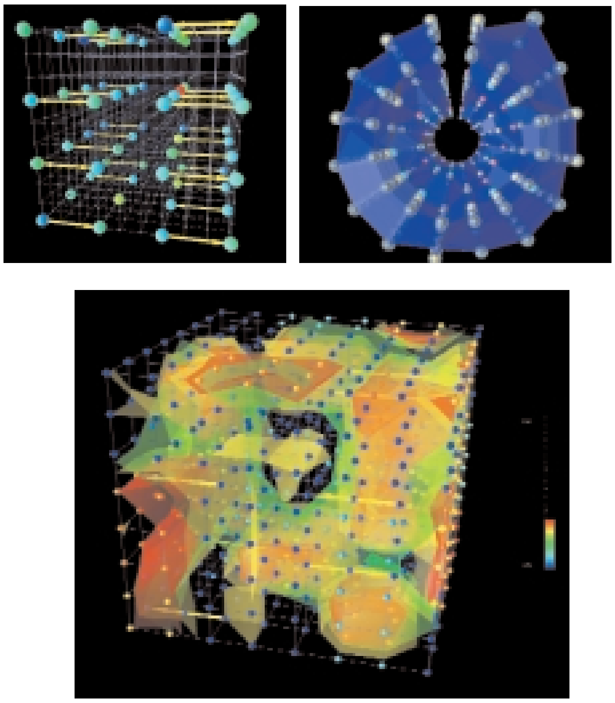


Figure 1. Top left: processor load (node color) and messaging (arrows) during an FFT on a 1,024-processor machine. The data have been downsampled so that only statistics at 64 "virtual" nodes are displayed. Top right: a 16-processor machine performing an FFT. The processor nodes at a given time-slice are arranged in a circle, and successive time-slices are layered to produce a tube. Event type at a given processor is shown by node color, and latency is shown by the diameter of the surrounding clouds. Distortions in the cylinder surface indicate load. Bottom: a complex visualization of a computation on a 512-processor machine. Isosurfaces provide a statistical overview of machine state and algorithm performance.

Immersive Visualization of a Very-Large-Scale Seismic Model

Contact
Prashant Chopra
Mississippi State University
NSF Engineering Research Center for Computational Systems
2 Research Boulevard
Starkville, Mississippi 39759 USA
prash@erc.msstate.edu

Joerg Meyer
Michael L. Stokes
Mississippi State University
NSF Engineering Research Center for Computational Systems

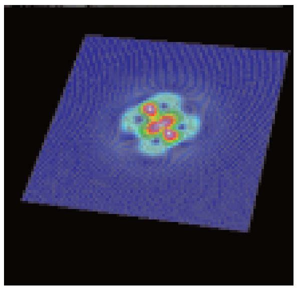
This application portrays our ongoing efforts to interactively and immersively visualize the results of very-large-scale earthquake simulations.

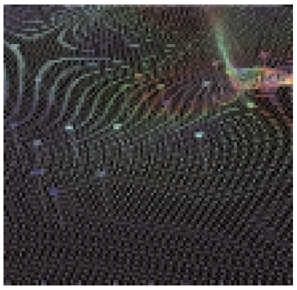
Origin of the Dataset

The original dataset is a result of a collaborative effort among the University of California, Berkley; Carnegie Mellon University; and Mississippi State University. The basic geometry consists of 11,800,639 nodes with their tetrahedral connectivity. In addition, each node has a velocity-vector attribute spanning over 120 time steps. Each simulation run generates structural responses for buildings with varying physical properties. The building locations are associated with selected nodes on the top surface of the structure, which represents a layered, block-shaped soil model.

Challenges

Our visualization team faced two immediate challenges: a large number of files in the raw dataset and numerous generated and derived attributes (for example, velocity, acceleration, ground motion, structural response for buildings, etc.).





Preprocessing

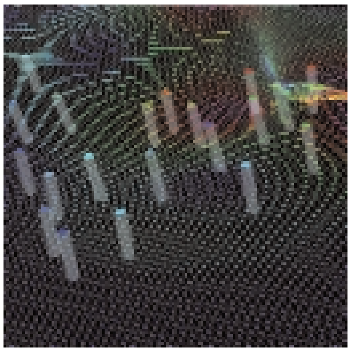
The first essential step to reduce this large-scale dataset was extraction of the geometry of the topmost layer as a triangulated surface, followed by an efficient loss-less encoding scheme for the whole dataset based on wavelet compression.

Visualization Paradigms

The first paradigm that was implemented was an interactive 3D time-varying visualization of the ground motion on a desktop. Though it was not very immersive, it gave us our first insight into the temporal behavior of the model. The next step led to an interactive simulation and visualization Web portal that generated the structural response data for selected parameters. The back-end rendering engine was made flexible enough to move the user to a prespecified location around the model with results presented in the form of an animated GIF or MPEG movie.

The Model

Initially, the buildings were modeled as simple sticktop structures, where the top of the buildings moved in response to the shaking ground. Now, fully textured translucent buildings have replaced this simple model. We decided on four visual cues for better insight: the color of the surface nodes (velocity), the color of the buildings' roofs (structural response), the shaky ground motion, and the shaking buildings in response to the ground displacement.



Immersion: Feel the Tremors

The above paradigms, though useful in their own ways, turned out to not fully exploit the immersive capabilities of a four walled CAVE. Currently, a pre-computed simulation can be viewed from different angles. Future versions will allow us to interactively manipulate the scene and control the simulations. Imagine moving around when the earthquake is "actually" happening around you. You can "see" the buildings shaking and the tremors spread out, while the ground below your feet is sliding and moving!

What's next in this project?

Adding an additional cue for sound, exploring below the surface as the tremor spreads out, modeling buildings with different structural properties in the same scene, and adding more photorealism.

Acknowledgements

Contributors: Gregory L. Fenves, Bozidar Stojadinovic (University of California, Berkeley), Jacobo Bielak et al. (Carnegie Mellon University), Tomasz Haupt, Purushotham Bangalore (Mississippi State University). Supported by the National Science Foundation.

Implementing Vector-Based Texturing In RenderMan

Contact
John Haddon
NCCA
Bournemouth University
United Kingdom
theboyhaddon@hotmail.com

Ian Stephenson
NCCA
Bournemouth University
United Kingdom

The predominant method of texturing for production is using bitmap image files in conjunction with procedural shading. However, the storage demands of bitmap images increase greatly with resolution, and procedural techniques are typically unsuitable for generation and fine control of complex figures.

Vector graphics provide resolution-independent, scaleable images, typically with low file sizes, and are easily designed using available software. This makes them ideal for use in some texturing situations, particularly where it is necessary to incorporate imagery in a graphic style. This sketch presents the implementation of vector-based texturing in a RenderMan renderer.

API

As implemented, vector graphics lookups appear to the shader writer much as the built-in texture() calls do. A family of new shadeops of the general form vtexture (uniform string filename, float u, float v, string filtertype) return texture color and alpha information. Beyond this, the shader writer requires no knowledge of the system's internals.

Implementation

vtexture() is implemented as two DSO shadeop calls. The first, called once per grid with uniform parameters, ensures that the required texture is loaded into a texture cache and marked as being current. At this point, the texture is in its idealised, resolution-free form.

A second shadeop call, executed once per micro-polygon, receives areas to be filtered as arbitrary quads in texture space and returns texture color and alpha information. This is achieved by generating and maintaining a cache of tiles (rasterised sections of texture) and filtering them appropriately. Tiles are rasterised at resolutions adapted to the lookups requested, and a new tile typically includes a reasonable area surrounding the current lookup area. This means that there is a fair chance of a tile that is suitable for the following lookups being already present in the cache. Rasterisation is a computationally significant process, so the effectiveness of this caching is essential to performance.

Example Usage

vtexture() was employed in rendering a sequence that shows a track into a globe, starting at a point where the whole earth is visible and ending on a small high-resolution section, specifically the Isle of Wight. For comparison, the same sequence was texured using an 8,000 x 8,000-pixel bitmap.

Both sequences were net-rendered with PRMan 3.9. The vector version required approximately three times the computing time of the bitmap. The vector texture was approximately 800K in size, whereas the bitmap was significantly larger (almost 250 MB of uncompressed data). However, most significantly, the vector texture provides resolution several orders of magnitude greater than that achievable with a bitmap of this size (Figure 1).

Considerations

vtexture() has been implemented and tested with PRMan. The Reyes algorithm typically generates successive texture lookups that are adjacent in texture space, as they are generated from adjacent grid points. Other rendering algorithms, particularly ray tracing, are unlikely to generate such adjacent lookups, resulting in much less effective caching of tiled data. Presumably, this would significantly limit performance.

Currently, rendering with the system is significantly slower than with substitute bitmaps. Although this is likely to always be the case, it is believed that optimisations, particularly at the rasterising and filtering stages, could significantly increase performance.

Vector texturing is by no means a panacea. Textures of a photographic nature are simply not representable in a vector form. Text and graphic shapes are among those most suitable for vector description, and these could be augmented with procedural techniques in situations that demand greater photo-realism.

Conclusion

It has been shown that vector-based texturing can be successfully implemented under RenderMan as an extension to the shading language. The system described operates with viable performance and over a significant range of resolution. This demonstrates the potential value of vector textures in production.

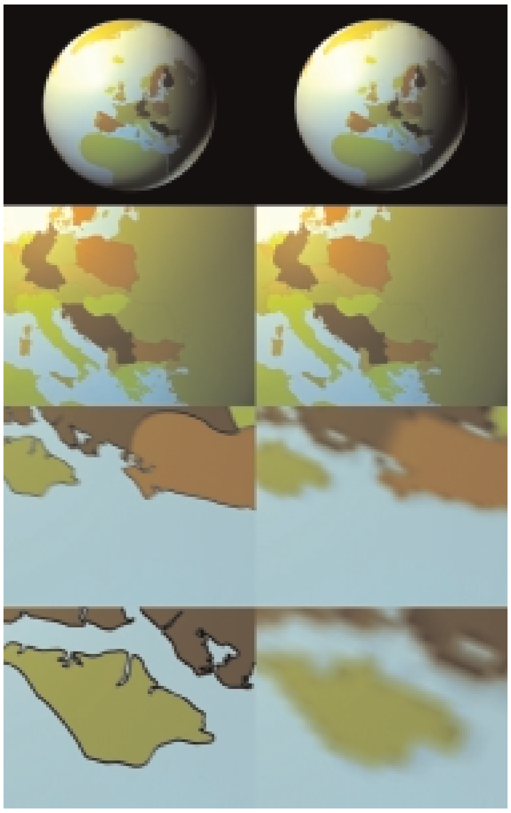


Figure 1: The vector texture (left) provides significantly increased resolution in comparison with the test bitmap texture (right).

An Integrated Solution to Camera Animation for Stage

Contact
Mathew Lamb
Digital Domain
lamb@d2.com

Over the last few years, pre-visualization has become an essential tool in development of visual effects work. Quite often at this stage, it is easy to have close directorial interaction and, as a result, produce work of fundamental significance to the project. When camera moves are being designed for stage, pre-visualization is doubly important, for not only is the director involved in an otherwise lengthy and tedious process, but also it is important that the integrity of the move that the director designs is not compromised by motion-control operators on stage, who may be trying to fix unforeseen problems, in a costly manner, on the day of the shoot.

To maintain the integrity of this previsualization work, Digital Domain developed a technique for transforming data from an animation package to a motion-control stage in an instantaneous and pixel-accurate manner. The system was first employed in production of "Supernova," and, over the following two years, it was refined to the extent that it has become the de facto solution for both designing and driving motion-control camera moves.

The solution was recently employed in development of a sequence of intricate camera moves for the notable commercial "Brobeck." The difficult challenge in this project was a requirement for a number of hook-ups among live, CG, and motion-control elements. Integral to this was the acute timing among all elements, so pre-visualization rapidly became a low-resolution proxy for development of the whole show. All the pre-visualization files included a kinematically accurate model of the motion-control rig and a complete model of the stage on which it was being shot.

Our current model of the rig contains a huge amount of flexibility. Essentially, it is a double-ended kinematic chain in which the track on which the rig rides, as well as the camera mounted at the other end of the chain, may be positioned independently of one another. In addition, since the rig has more degrees of freedom than are needed to solve for any given camera goal, redundant configurations exist. We capitalize on this and provide them as alternate solutions, which allows for greater flexibility in overcoming obstructions or space constraints on stage. The entire system is modeled in Houdini, our software of choice for integration and effects work, with the result that the director is, in real-time, able to repeat a proposed move until it works.

The range of achievable moves is significantly enhanced because the system includes interaction of the model being shot in the simulation. In this manner, degrees of freedom unobtainable by the camera can be handed off to the model mover (or vice-versa). A good example is the "boomerang," in which a camera move that pushes up to a model and then way past it is converted into a push toward a model, combined rotations of the model and camera, and then a pull away that is visually identical but requires half the length of the stage.

Accuracy is of utmost importance, so the solution includes a triangulation step in which the camera rig and model are precisely localized in space. This process, which only takes a few minutes, results in construction of a transform that allows the pre-visualization camera to align perfectly with stage. Throughout the process, the shoot is augmented by a video tap from the camera that is overlaid on the view of the model in Houdini. From this vantage point, it is trivial to verify that intended and actual moves remain aligned.

This system has proven hugely successful from the very first time it was rolled out, when it saved some 50 percent of our time on stage, to the "Brobeck" commercial, in which the director and animator barely noticed the underlying solution. Instead, they could concentrate on producing high-quality animations, secure in the knowledge that nothing would be compromised anywhere else in the process.

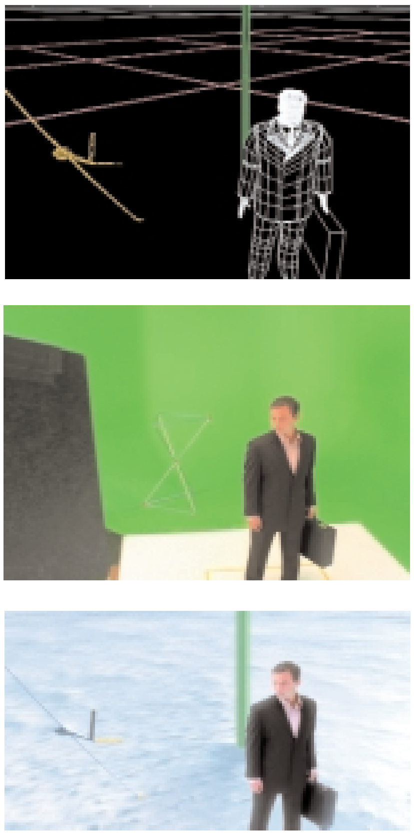


Figure 1. Three stages of the motion-control shoot: a wireframe camera view in Houdini; the green-screen plate on the day of the shoot; and the final composited image in the commercial.

An Interactive System for Robust Topological Modeling of Meshes

Contact
Ergun Akleman
Visualization Laboratory
216 Langford Center
Texas A&M University
College Station, Texas 77843
USA
+1.979.845.6599
+1.979.845.4491 fax
ergun@viz.tamu.edu

Jianer Chen
Department of Computer Science
Texas A&M University

Vinod Srinivasan
Visualization Sciences
Department of Architecture
Texas A&M University

Current computer graphics practice is almost exclusively based on polygonal meshes. To avoid artifacts such as wrongly oriented or missing polygons, and T-junctions, the polygonal mesh must satisfy a mathematical property called 2-manifold. 2-manifolds are essential for most computer graphics applications. For instance, initial control mesh for subdivision schemes must satisfy 2-manifold property. A polygonal mesh that has a missing polygon can ruin the radiosity computation. In ray tracing, a transparent shape with a wrongly oriented polygon can cause undesirable artifacts in the resulting image.

Topological modeling of 2-manifold polygonal meshes has always been a difficult challenge in computer graphics. Our Doubly Linked Face List (DLFL)[1,2] provides an effective solution to this challenge. It always corresponds to a valid, orientable 2-manifold polygonal mesh and provides a minimal set of operations to change the topology of 2-manifold meshes.

This sketch presents a prototype system to demonstrate the power of DLFL for development of interactive polygonal mesh modelers. Users of our system can easily change topology: create and delete holes and handles, connect and disconnect surfaces. Our system also provides subdivision schemes to create smooth surfaces. Moreover, the system provides automatic texture mapping during topology and smoothing operations. It is topologically robust in the sense that users will never create invalid 2-manifold mesh.

To demonstrate the effectiveness of the system, we have created various polygonal meshes that would be extremely difficult to model interactively without our system. The nested shapes shown here represent an example of models that can be interactively constructed using our system. The inspiration for this shape came from Chinese sculptures consisting of a set of nested, rotatable balls. The actual sculptures can have up to 16 nested balls. Our version consists of three surfaces with genera 31, 31, and 41, respectively.

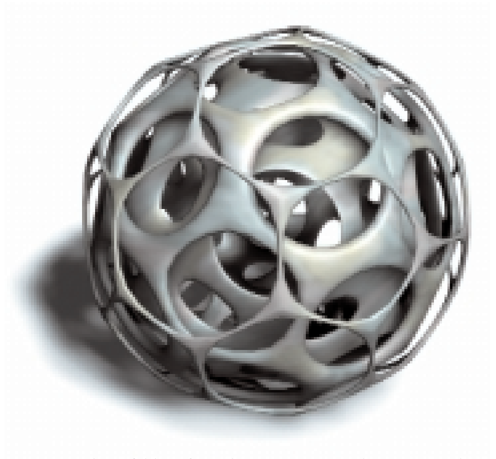
Creating holes and handles is not only useful for aesthetic purposes. In fact, holes and handles are essential to construction of functional models. The teapot shown here represents an example of a functional model. As can be seen from an X-ray image, this teapot has a real (not just a "look-like") hole to let the water pour from the spout. Because of the hole in the spout, this teapot can be used in physical simulations. The hole and the handle are designed in our system starting from a few rectangular prisms.

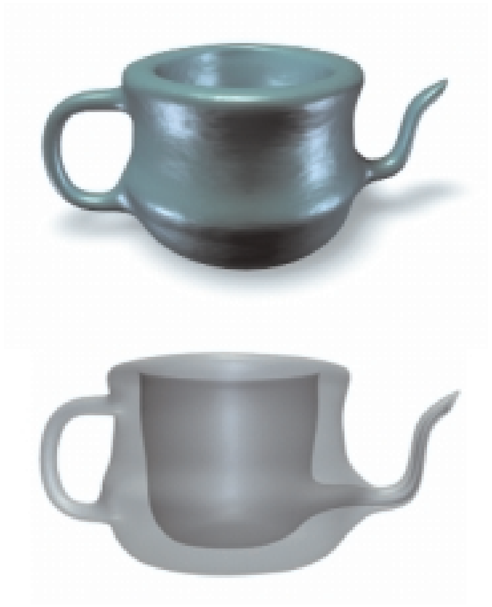
Acknowledgments

We are thankful to Sajan Skaria, P. David Romei, Glen Vigus, Christina Haaser, Mark Clayton, and Derya Guven Akleman for their help and suggestions.

References

1. Akleman, E. & Chen, J. Guaranteeing the 2-manifold property for meshes with Doubly Linked Face List. *International Journal of Shape Modeling, 5* (2), 149-177.
2. 1. Akleman, E., Chen, J., & Srinivasan, V. (2000). *A new paradigm for changing topology during subdivision modeling*, 192-201. Pacific Graphics, 2000.


Nested maifold surfaces that are interactively constructed using our system.


A teapot created by using our system (top) and its X-rayimage (bottom).

Interactive Virtual Clay Using Implicit Surfaces and Particle Systems

Contact
Masatoshi Matsumiya
Nara Institute of Science
and Technology
masato-m@is.aist-nara.ac.jp
yokoya.aist-nara.ac.jp/

Naokazu Yokoya
Haruo Takemura
Nara Institute of Science
and Technology

This sketch presents a virtual clay model developed for interactive virtual clay works that require real-time computation and rendering. Ordinary modeling software for 3D free-form objects has a lot of parameters to tune and a number of limitations on the object's topology and geometry due to underlying mathematical descriptions. Therefore, users must have enough mathematical knowledge and flexible spatial recognition to apply geometric and topological operations to free-form objects. Such problems in free-form modeling can be solved by regarding the objects as clay that can be deformed freely. When users can deal with objects in the same way they would deal with real clay, handling of free-form objects becomes very easy and user-friendly. To realize this modeling concept, virtual clay model must deform in real time.

Virtual Clay Model

Because it is a plastic fluid, clay has a yield point, a shear stress that has to be overcome so the fluid can start to flow. When a shear stress is below the yield point, clay has a solid structure that prevents plastic flow. Once the yield point is exceeded, the plastic flow allows clay to deform as its volume is preserved. We represent clay using particle systems and implicit surfaces.

In implementing particle systems, spatially interacting particles are used to approximate models for clay. In Figure 1, spatial interaction forces (attraction and repulsion) that act on any pair of particles, are defined depending on their positions. The motion of a particle is governed by:

$$\boldsymbol{F}_i = m_i \frac{d^2 x_i}{dt^2}$$

where Fi denotes a spatial interaction force applied to the i-th particle, which has mass mi and position xi. Particles that receive force below a threshold Fth (particle moving threshold) are not governed and stay where they are. We use Euler's method in calculating a numerial solution of this derivative function.

An implicit surface based on skeletons (skeletal implicit surfaces)[1] is employed to represent smooth surfaces. A skeletal implicit surface is defined by:

$$\{P \in R^3 \mid f(P) = c\},\ f(P) = \sum_{i=1}^{n} F_i(d(P, S_i)),$$

where P is a point in space, and f(P) is the value of a scalar field (implicit value) at the point P. An iso-surface surrounds a solid whose points satisfy f(P) = c. In skeletal implicit surfaces, the implicit value f(P) is generated by a set of skeletal elements Si(i = 1… n) with a set of associated field functions Fi as shown in Figure 1.

We combine particle systems and implicit surfaces into a virtual clay model. First, particles are evenly arranged with a stable distance as shown in Figure 2(a). In the interactive deformation process, each particle moves, preserving the stable distance as shown in Figure 2(b). Therefore, the deformation can preserve the volume like real clay. The particle-moving threshold Fth corresponds to the yield point of plastic fluid. The surface shape of clay is generated by regarding these particles as skeletal elements of skeletal implicit surfaces as shown in Figure 2(c).

To achieve interactive deformation of the virtual clay model, an image of the deformation must be renderd in real time. To reduce the time required for rendering for interactive modeling, we have developed an efficient algorithm for polygonizing skeletal implicit surfaces based on Bloomenthal's algorithm,[2] because polygons can be quickly rendered by using conventional graphics hardware. In our polygonization algorithm, real-time processing is realized by limiting the area of polygonization to the area around the moving particles.

Result

Figure 3 shows an image sequence of a deforming virtual clay model that consists of 343 particles. The model is deformed by pushing a part of the model. The computation time during the deformation is measured on a SGI Onyx2 (six MIPS R10000 195MHz CPUs). In the polygonization process, 0.65 seconds are required for one cycle at the maximum load. In particle systems, 0.044 seconds are required for one cycle of calculation of Euler's method. The frame rate of 30 fps is accomplished by carrying out these processes using multiple threads. These results show that a virtual clay model exhibits clay-like deformation and can be calculated and rendered in real time.

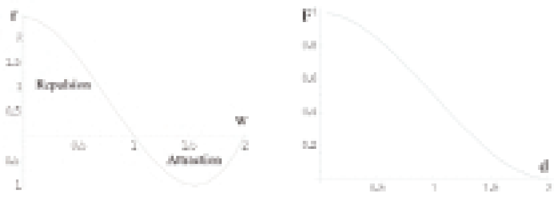


Figure 1. Spatial interaction forces (left) and field function (right).

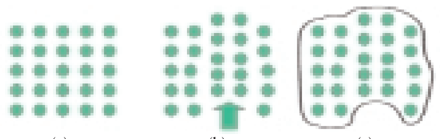


Figure 2. Virtual clay model using particle systems and implicit surfaces.

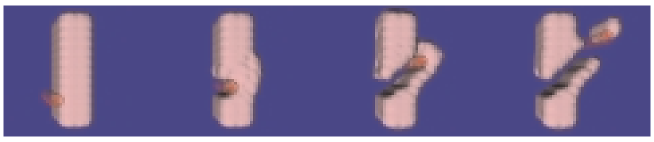


Figure 3. Image sequence of model deformation.

References
1. Cani-Gascuel, M.P. & Desbrun, M. (1997). Animation of deformable models using implicit surfaces. IEEE Trans. on Visualization and Computer Graphics, 3 (1), 39-50.
2. Bloomenthal, J. (1984). An implicit surface polygonizer. In P. Heckbert, Ed., *Graphics Gems IV*, 324-349. Academic Press, 1994.

Intuitive Multiple Viewpoints Control using Interlocked Motion of Coordinate Pairs

Contact
Shinji Fukatsu
Osaka University
fukatsu@eie.eng.osaka-u.ac.jp

Yoshifumi Kitamura
Toshihiro Masaki
Fumio Kishino
Osaka University

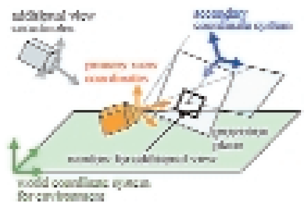
Adequate presentation of multiple views from different positions and directions, as well as in different scales, enables a user to acquire much information about an environment and recognize the environment in detail. In considering effective use of multiple views, the control method for multiple viewpoints (specifically, a primary viewpoint and additional viewpoints) must be clear to the user. A lot of existing viewpoint manipulation methods originally deal with a single viewpoint;[1] however, a method for effectively controlling multiple viewpoints has not yet been discussed. In this sketch, we propose the "interlocked motion of coordinate pairs" as a manipulation technique for intuitively controlling additional viewpoints and the primary viewpoint.

Interlocked Motion of Coordinate Pairs

Figure 1 shows the coordinate system used in the proposed technique. Three coordinate systems (world coordinates, primary-view coordinate, and additional-view coordinates) are used to present primary and additional view. In addition to these three fundamental coordinate systems, we introduce a secondary coordinate system to determine the additional viewpoints and then interlock the motion of the additional viewpoint with the relative motion of the primary viewpoint and the secondary coordinate system. The secondary coordinate system differs from a world coordinate system for the environment in the geometric relationship (scale, origin, and coordinate system axes). The generated additional view is displayed in the window on the projection plane of the primary view (Figure 2). Therefore, the original data size of the geometry of the environment does not increase even if the number of additional view images (viewpoints) does.

Intuitive Control of Multiple Viewpoints

There is some variation in the implementation of our proposed technique. Humans intuitively perceive the position and orientation of their own bodies by the sense called proprioception. Therefore, we couple the primary viewpoint and the secondary coordinate system with the user's natural movements. For example, the primary viewpoint is coupled with the user's head motion, and the secondary coordinate is coupled with the user's hand motion (Figure 3). Figure 4 shows the change of the user's view from initial condition (Figure 2) by the user's movements in this implementation example. The user's head rotation changes the orientation of both primary viewpoint and additional viewpoint – Figure 4(a) – and the user's hand rotation changes the orientation of the additional viewpoint – Fig. 4(b). This enables the user to intuitively understand the correspondence between the additional view and the primary view based on proprioception. Also, users can feel that they hold and manipulate a miniature of the environment in their hands and observe it through the window of their primary view.


additional view coordinates
secondary coordinate system
primary view coordinates
projection plane
window for additional view
world coordinate system for environment
Figure 1. Coordinate systems of proposed technique.

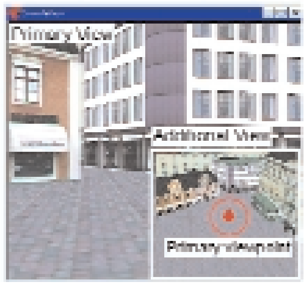


Figure 2. User's view.

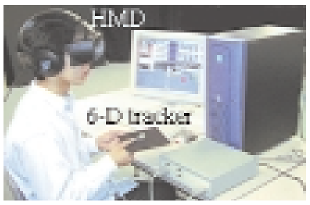


Figure 3. System overview.

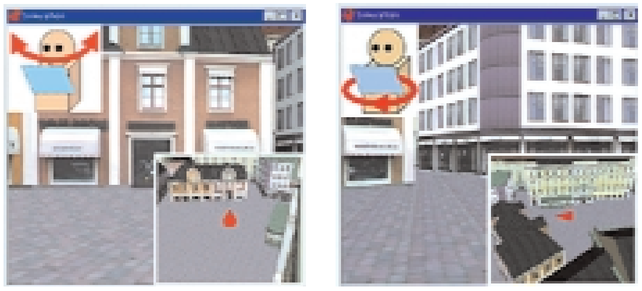


Figure 4. Change of user's view: (a) rotation of user's head; (b) rotation of user's hand.

Reference

1. Ware, C. & Osborne, S. (1990). Exploration and virtual camera control in virtual three-dimensional environment. *Computer Graphics of the ACM, 24* (4) , 175-183.

k-DOPs as Tighter Bounding Volumes for Better Occlusion Performance

Contact
Dirk Bartz
Dirk Staneker
WSI/GRIS
Universität Tübingen
bartz@gris.uni-tuebingen.de

James T. Klosowski
IBM T. J. Watson Research Center

Bounding volumes are used in computer graphics to approximate the actual geometric shape of an object in a scene. The main objective is to reduce the costs associated with interference tests such as intersection tests in ray tracing and collision detection, and to determine the visibility of an object. The bounding volumes most commonly used for these purposes are axis-aligned bounding boxes (AABB). However, in many cases this approximation fills a much larger volume in object space and a much larger screen area (once rasterized into screen space) than the actual geometry. This results in false-positive interference results that can increase the computational load significantly.

Alternatively, oriented bounding boxes (OBB) were proposed, where the spanning axes of the bounding box are oriented according to the shape of the object, thus generating a tighter approximation of the original shape than AABBs. While OBBs perform better for collision detection than AABBs, the benefits for occlusion culling are significantly smaller. This is mainly due to the fact that the rasterized screen area of an OBB is almost the same as for an AABB, and that the corners of an OBB still protrude through exterior hull elements, which occlude the actual geometry.

Another commonly used bounding volume primitive is spheres, which have also been used for ray tracing and collision detection, since intersection with a sphere is very easy to compute. However, a sufficiently tessellated sphere requires many polygons, which increase the costs for an image-space occlusion culling interference test. Furthermore, spheres tend only to approximate compact objects well.

Convex hulls are also good bounding primitives, but they are significantly more expensive to compute than other bounding volumes,[1] and they quickly become impractical in design tasks, where model objects are modified frequently.

In 1996, Klosowski et al.[1] proposed a collision-detection scheme using discrete orientation polytopes (k-dops), which enabled faster collision tests than OBBs. Essentially, k-dops are an approximation of an object by computing bounding planes of an object along k/2 directions.[2] An AABB is one example of a 6-dop, whose bounding planes correspond to the coordinate axes. Another common k-dop is the 26-dop, which is an AABB with the 12 edges and eight corners cut to the object's surface (6 + 12 + 8 = 26 bounding planes).

Experiments

In our experiments, we employed an image-space occlusion-culling test using the Hewlett-Packard occlusion-culling flag, implemented on the HP fx-series of graphics subsystems. This flag determines if geometry rendered during a special occlusion mode will modify the depth buffer, which indicates potential visibility. In other words, if a bounding volume is rendered, but the HP occlusion-culling flag indicates that the depth buffer would not have changed, then we need not render any of the geometry contained within that bounding volume. We use this flag on a depth-sorted list of objects of the tested models, which are located in the view frustum.

We tested a variety of "real-world" MCAD datasets using AABBs and k-dops. On average, we achieved a 50-percent improvement in the culling rate using k-dops instead of AABBs. The interior objects of MCAD datasets are frequently occluded by exterior hood or cover objects, like the hull of the servo screwdriver in Figure 1. However, AABBs do not provide a very tight approximation for rounded shapes. Hence, they frequently extend through the hull objects and generate false-positive visibility test results. In contrast, k-dops provide a much tighter approximation, where corners of the respective AABB have been cut off.

The polygonal complexity of a k-dop is naturally larger than the complexity of an AABB; if k=26, up to 26 polygons are used for a kdop, while only six polygons are needed for an AABB. However, an occlusion-culling query requires an update or synchronization of the visibility information, which is a pipeline flush for HP occlusion-culling flag-based approaches. The latency of the pipeline flush is equivalent to rendering approximately 190 triangles of an average size.[3] If the graphics subsystem does not provide hardware support for such queries, this latency is even larger. Experiments with specific polygonal models where k-dops do not facilitate a higher culling rate provide evidence for this statement, since the higher overhead of rendering three times more polygons for the occlusion test is not reflected in a lower frame rate. In fact, the frame rate did not change much beyond the limits of measurement noise.

Overall, k-dops provide tight bounding volumes for polygonal objects. Compared to AABBs, they significantly reduce false-positive visibility queries. The increased rendering costs due to the higher polygonal complexity of the k-dops are overshadowed by the latency of the required synchronization step of state-of-the-art graphics subsystems.

References
1. Held, M., Klosowski, J., & Mitchell, J. (1996). Real-time collision detection for motion simulation within complex environments. *Visual Proceedings of SIGGRAPH 96*, 151.
2. Kay, T. & Kajiya, J. Ray tracing complex scenes. *Proceedings of SIGGRAPH 86*, 269-278.
3. Severson, K. (1999). VISUALIZE fx graphics accelerator hardware. Hewlett-Packard Company Whitepaper, 1999.

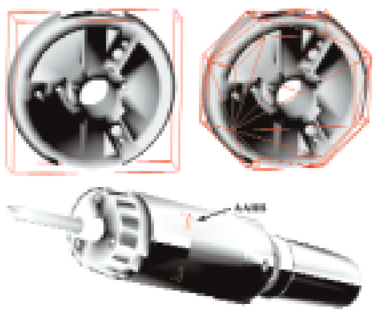


Figure 1. Servo screwdriver: (top left) motor part and AABB; (top right) motor part and 26-dop; (bottom) AABB of motor part is visible through hull, while 26-dop is completely occluded by hull.

Life-Sized Projector-Based Dioramas: Spatially Real and Visually Virtual

Contact
Kok-Lim Low
Department of Computer Science
University of North Carolina at Chapel Hill
lowk@cs.unc.edu

Greg Welch
Anselmo Lastra
Henry Fuchs
Department of Computer Science
University of North Carolina at Chapel Hill

In this sketch, we present work in progress on a new projector-based approach to visualizing re-creations of real or virtual places. The major difference between our approach and previous ones is that we use a set of life-sized display surfaces that closely approximate the scene geometry. The effect is a virtual environment that is both visually and spatially realistic, which gives the user a strong sense of immersion.

Our long-term goal is to re-create real remote places, providing users with a realistic sense of presence in the real places. We are interested in allowing people to experience re-creations of famous places such as Monticello, President Thomas Jefferson's home. For now we are working in a research laboratory, but we envision museum spaces with changing exhibits of far-away locations.

Raskar et al. have explored the use of projectors to illuminate real-world objects by projecting images of computer models that closely approximate the objects' geometry.[1] Here, we extend their ideas to life-sized virtual environments, using life-sized display surfaces that closely approximate the actual scene geometry.

Advantages and Limitations

Using physical display surfaces that closely match the actual scene geometry has many advantages over traditional approaches, such as HMD VR and CAVE.[2] Like a CAVE, our approach offers a wide-field-of-view experience that fills the user's peripheral vision. Such peripheral vision is necessary for maintaining spatial awareness during navigation in virtual environments[3] and provides a better sense of immersion.

The physical arrangement of display surfaces allows the user to really walk around in the virtual scene. Real walking gives a stronger sense of presence than walking in place and virtual flying[4] but at the expense of much larger physical space.

In general, different degrees of approximation of the scene geometry produce a spectrum of display surfaces that range from single flat screens to display surfaces that exactly match the scene geometry. When the display surfaces are significantly different from the scene geometry (which is typically the case in a CAVE), sensitivity to viewing errors (caused by system latency, errors in projector and tracker calibration, etc.) is significant. However, when the display surfaces closely match the scene geometry, sensitivity to such errors is minimized. In the best case, the approach effectively offers auto-stereovision that can support multiple untracked users who simultaneously explore the same virtual scene.

Unfortunately, building an exact detailed physical replica of the scene is usually infeasible. This has forced us to use a simplified set of display surfaces that approximate the scene geometry. For example, primary structures of building interiors and mid-sized architectural objects (walls, columns, cupboards, tables, etc.) can usually be approximated with simple shapes (boxes, cylinders, etc.). The trade-offs include re-introduction of some latency in visual feedback and partial loss of auto-stereovision. The user's eyes now need to be tracked.

Another significant limitation of our approach is that it requires a large physical space to faithfully re-create a large scene. Our approach is also not suitable for dynamic scenes that have large moving objects (cars, for example). Finally, while we have ideas for projector placement that would minimize viewer occlusions, we find it natural to want to approach and touch the display surfaces. In our envisioned museum setting, one might actually have to "rope off" areas near the display surfaces.

Status

We have performed some preliminary experiments in which the scene geometry is approximated by styrofoam blocks, and a two-pass rendering approach is used to generate the correct images as viewed from the user's tracked eye (see figures). The results are encouraging. While we currently only demonstrate our ideas with a synthetic scene, we have collected and are in the process of preparing very high-quality image-based models of the Monticello library. We acquired the models using a 3rdTech laser scanner during a multi-day trip to Monticello.

Challenges

The need to model and build non-trivial physical display surfaces is a challenge not seen in other projector-based approaches. The other problems of our approach are common to most multi-projector display systems. These problems include shadows, inter-reflections, overlapping projections, non-ideal display-surface properties, projector placement and calibration, multiple views, and rendering resource management. Moreover, to re-create real places, efficient and effective methods are needed to acquire the data, process them, store them, and make them suitable for interactive rendering.

References
1. Raskar, R., Welch, G., & Low, K. (2000). Shader lamps: Animating real objects with imaged-based illumination. Technical Report TR00-027, Department of Computer Science, UNC-Chapel Hill, January 2000.
2. Cruz-Neira, C. et al. (1993). Surround-screen projection-based virtual reality: The design and implementation of the CAVE. *Proceedings of SIGGRAPH 93.*
3. Leibowitz, H.W. (1986). Recent advances in our understanding of peripheral vision and implications. In the 30th Annual Meeting of the Human Factors Society, 1986.
4. Usoh, M. et al. (1999). Walking > walking-in-place > flying, in virtual environments. *Proceedings of SIGGRAPH 99.*

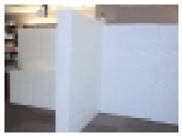


Figure 1. The display surfaces are built from styrofoam blocks.

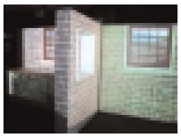


Figure 2. Perspectively correct imagery of the scene is generated in real time and projected onto the blocks.

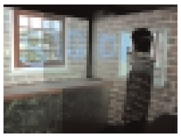


Figure 3. A user is virtually spray painting from a tracked "spray gun."

Lifting Detail from Darkness

Contact
J.P. Lewis
Disney TSL
3100 Thornton Avenue
Burbank, California 91505 USA
zilla@computer.org

This application describes a high-quality method for separating detail from overall image region intensity, an intensity-detail decomposition. The method can be used to automate some specialized image alteration tasks.

Our work was motivated by the movie "102 Dalmatians," which featured a dalmatian puppy without spots. Animal handlers determined that the obvious idea of applying makeup to the dog was not possible. There was no suitable makeup that was both safe to the dogs and that would stay in place during normal dog activity (including licking). This left Disney TSL with the task of painting out all the spots on the dog every few frames (the paintings were carried for a few frames with a simple scheme).

The spot-removal task was larger than anyone guessed, and ultimately required a large number of artists (up to 40) working for eight months. The problem also proved to be more difficult than expected from an algorithmic point of view. As the spots often had visible fur texture, we initially believed that there must be some simple compositing technique that could lighten the spots.

To get a feel for the problem, consider one representative approach inspired by unsharp masking: blur the dog and then subtract the blurred version from the original, giving a high-pass texture containing the fur. Then correct the intensity of the blurred dog to remove the spots (without saying how this is done). Lastly, add the high-pass texture to the lightened, blurred dog, resulting (we hoped) in a dalmatian without spots but with fur derived from the original texture. The problem with this approach is suggested in Figure 1: the scale of the blur must be exactly matched to the spot profile or there will be an overshoot/undershoot around the spot edge. This is not achievable, since the spot transition regions have markedly different widths even on opposite sides of a single spot. Some more adaptive technique was required.

Adaptive Filtering

The intensity-detail decomposition problem is reminiscent of the problem addressed by Wiener filtering: separating signal from noise. Making use of this observation, by casting detail in the role of "noise" and intensity in the role of "signal," we were able to apply a Wiener separation approach; a simple spatial-domain adaptive Wiener filtering algorithm described by Lim[1] works quite well.

Intensity Modification

Once the detail is successfully separated, we need a means of altering the image region intensity in a simple and controllable fashion. The membrane (Laplace) equation $\nabla 2u = 0$ produces an interpolation of specified boundary conditions that minimizes $\int (\nabla u)2dA$ (the integrated gradient); as such, it provides an adequate way of interpolating intensity specified on a boundary curve (for example, a rough spot boundary). The Laplace equation is a linear system $Au = b$ with A being a sparse square matrix whose dimension is the number of pixels in the region. The intensity-detail decomposition was initially prototyped in MATLAB, which took less than a day and used that package's sparse matrix routines. Our algorithm was later reimplemented in Java, which took about three months.

Approximately half of that time was devoted to the membrane solution. The first implementation was a direct (non-sparse) matrix solution, programmed by Yi-Xiong Zhou. This was adequate for very small regions but was too inefficient for larger regions. Areas of 150^2 pixels were requiring several minutes and 0.5G of memory! Fortunately, the Laplace equation can be solved with the multigrid technique[2] (and in fact is the model problem for this approach). A multigrid implementation of the membrane reduced the solution time to several seconds even for large regions.

Applications

In addition to the spot-removal application, intensity-detail decomposition has other specialized applications such as altering or removing shadows and reducing wrinkles (Figure 3). It should be emphasized that although the effects shown here are routine work for a photoshop artist:

- Each image alteration shown here was produced with no artistic skill from a crude outline for the desired region in a few seconds. Consider Figure 3: the altered regions have luminance gradients as well as recovered original texture, so the effects could not be produced with a simple cloning operation but would require careful airbrushing followed by detail painting.
- Unlike manual retouching, the detail decomposition can be keyframed and produces consistent effects across frames.

References
1. Lim, J.S. (1990). *Two-dimensional signal and image processing*. Prentice-Hall, 1990.
2. Press, W.H., Teukolsky, S.A., Vetterling, W.T., & Flannery, B.P. (1993). *Numerical recipes*.Cambridge, 1993.

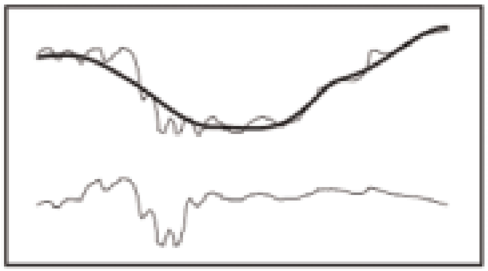


Figure 1. Hypothetical spot luminance profile, blurred luminance (heavy), and unsharp mask (bottom). When the blur size is too large, the texture overshoots; when it is too small, the blurred curve follows the texture, and texture amplitude is reduced.

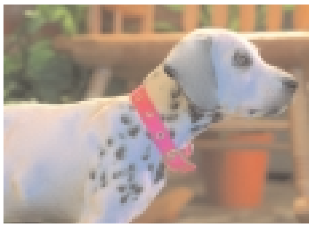


Figure 2. Demi-dalmatian.

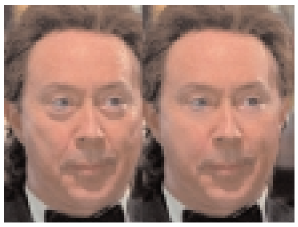


Figure 3. Altered regions include shadows under nose and eyes, and other changes.

Light Field Rendering with Triangular Viewpoint Mesh

Contact
Dongho Kim
Department of Computer Science
The George Washington University
dkim@seas.gwu.edu

James K. Hahn
Department of Computer Science
The George Washington University

Light field rendering[1] is a representative image-based rendering system using two-plane parameterization. The dataset in this scheme is a set of rays passing through regular grids sampled on two parallel planes. And this 4D dataset in (s,t,u,v) space can be thought of as a set of 2D images that have their viewpoints located at rectangular grids on a plane. We use (s,t) coordinates to parameterize the viewpoints and (u,v) coordinates to parameterize pixels in the images.

Rendering from this dataset is a process of reconstruction of rays. Due to the nature of digital sampling with limited resolution, we cannot avoid rendering error. In this work, we propose a new triangular sampling scheme of (s,t) plane. And we compare this scheme with rectangular sampling.

Figure 1(a) shows rectangular sampling in the original light field rendering. The viewpoints are sampled regularly at rectangular grids in (s,t) space. Figure 1(b) shows new parameterization of (s,t) based on triangular sampling. Here, the viewpoints are located at the vertices of a triangular mesh. This sampling can be obtained by moving the sample points in alternate rows of the original sampling by half the distance between the samples, and we can know that the sampling density is the same in both cases. In other words, the cost for the sampling does not change in new parameterization.

Since a light field is a set of discrete samples, we have to consider the aliasing problem. Figure 2(a) shows the frequency domain spectrum after the sampling in Figure 1(a). Due to the digital sampling, the spectrum of the original signal is repeated along two dimensions of the frequency domain. The spacing between the repetitions is determined by the sampling intervals. Here, we assume that the original signal is band-limited within a circle. This assumption makes sense if the original signal does not have any directional strength. Because of the overlapping spectrums, we get aliasing error from the overlapping spectrums. Figure 2(b) shows the frequency domain spectrum after the sampling in Figure 1(b). In this case, due to the characteristics of interlaced sampling, the amount of aliasing is reduced because the distances between spectrums are increased. In other words, for the same amount of aliasing, we need a smaller number of sampling points with new parameterization.

For reconstruction in original light field rendering, quadrilinear interpolation is used in (s,t,u,v) space. This is bilinear interpolation in (s,t) viewpoint sampling space. For new triangular parameterization, we interpolate with the barycentric coordinates of the reconstructed rays. So barycentric weights determine the interpolation in (s,t) space. Since we still use bilinear interpolation in (u,v), 12 samples are involved in the reconstruction, while original light field rendering uses 16 samples. Obtaining barycentric coordinates for reconstructed rays can be done quickly by calculating the distance to the parallel lines of the triangular mesh. Or hardware-based rendering presented in Isaksen et al.[3] can be used. Figure 3 is a rendering example from the DRAGON dataset in the Stanford light field archive.[4]

In order to compare two parameterizations, the images are reconstructed using the same viewpoints. From the original 32 x 32 images in the DRAGON dataset, we use 16 x 16 images for both parameterizations so that the samplings are performed as in Figure 1. And we reconstruct at the viewpoints along the blue lines in Figure 1, where the correct images are known from the DRAGON dataset but are not used in both samplings. From a rendering of 225 viewpoints, triangular sampling showed 2.1 percent less error on average, while using fewer samples for reconstruction. Moreover, the maximum error is smaller than original light field rendering. This is due to the enhanced sampling with less aliasing.

In future work, we need more concrete verification of the suggested parameterization. One method could be rendering a lot of images from randomly sampled viewpoints and performing statistical comparison. Or it is possible that spectral analysis like that suggested by Chai et al.[2] will be desirable.

References
1. Levoy, M. & Hanrahan, P. (1996). Light field rendering. *Proceedings of SIGGRAPH 96*, 31-42.
2. Chai, J., Chan, S., Shum, H., & Tong, X. (2000). Plenoptic sampling. *Proceedings of SIGGRAPH 2000*, 307-318.
3. Isaksen, A., McMillan, L., & Gortler, S.J. (2000). Dynamically reparameterized light fields. *Proceedings of SIGGRAPH 2000*, 297-306.
4. URL: www-graphics.stanford.edu/software/lightpack/lifs.html

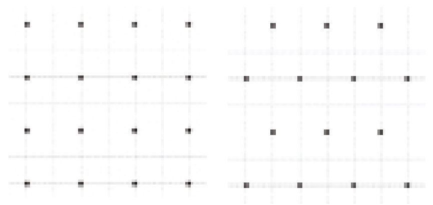


Figure 1. Sampling grids: (a) rectangular sampling, (b) triangular sampling.

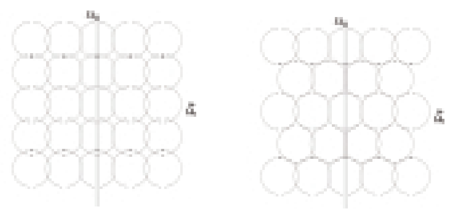


Figure 2. Frequency spectrum: (a) rectangular sampling, (b) triangular sampling.

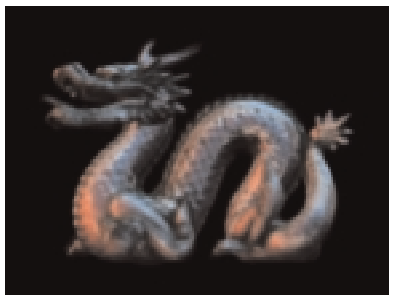


Figure 3. Reconstructed image.

Light Stage 2.0

Contact
Tim Hawkins
Institute for Creative Technologies
University of Southern California
timh@ict.usc.edu

Jonathan Cohen
Chris Tchou
Paul Debevec
Institute for Creative Technologies
University of Southern California

At SIGGRAPH 2000, we presented an apparatus for capturing the appearance of a person's face under all possible directions of illumination. The captured data can be directly used to render the person into any imaginable lighting environment, and can also be used to build photo-real computer graphics models that capture the unique texture and reflectance of the face. We have recently been developing the next generation of this lighting apparatus, which we call Light Stage 2.0.

Light Stage 2.0 is a much faster and more precise version of its predecessor[1]. The original device allowed a single light to be spun around on a spherical path so that a subject could be illuminated from all directions, and regular video cameras were used to record the subject's appearance as the light moved. This system had two major problems. First, since the light was moved around by pulling on various ropes, it was hard to be sure what the precise location of the light was at any given time. Second, because the device could not be spun very fast, and because of the limit of 30 frames per second imposed by the video cameras, it took over a minute to do a data capture. Since the subject must remain still during the data capture, this meant we could only capture people in very passive expressions, and even then multiple trials were often needed.

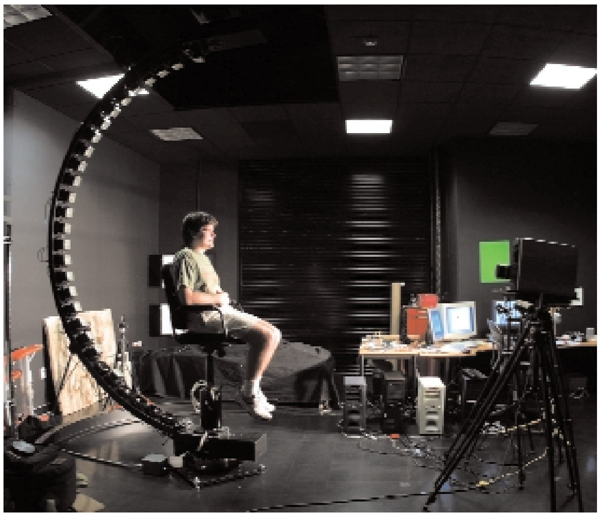
With Light Stage 2.0 (shown in Figure 1), we can capture all of the different lighting directions much more rapidly, with only a single rotation of a semicircular arm, and with greater accuracy. Thirty strobe lights arrayed along the length of the arm flash repeatedly in rapid sequence as the arm rotates. High-speed digital cameras capture the subject's appearance. This allows all directions of illumination to be provided in about four seconds, a period of time for which a person can easily remain still. It is also much easier to capture facial expressions that would be very difficult to maintain for an extended period of time (smiling, frowning, wincing, etc.).

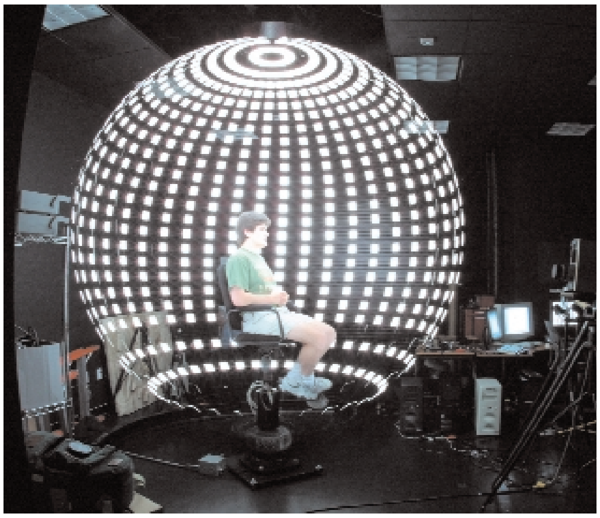
We are currently working on integrating geometry capture to provide a complete model of the subject. For this, we use digital LCD projectors to project different structured patterns onto the subject, quickly recording the appearance of the subject under each of the patterns with our high-speed cameras. From these structured-light data, the geometry of the subject is easily recovered. These data together with the reflectance data may provide more complete and photo-real models of faces than ever before.

In the next few months, we will be researching new ways of analyzing the large amount of reflectance field information captured in a Light Stage 2.0 scan and adapting the datasets for use in facial animation. We would also like to make our capture process even faster, with the goal of being able to capture both geometry and reflectance information in about five seconds. Our future plans include new prototype lighting devices that will allow similar datasets to be captured many times a second. This will allow an actor's performance to be recorded and then rendered photo-realistically into virtual environments with arbitrary lighting, where the performance can be viewed from arbitrary angles.

Another approach is to directly illuminate an actor with light sources aimed from all directions whose intensity and color is controlled by the computer. In this case, if the incident illumination necessary to realistically composite the actor into a particular scene is known in advance, the actor can be filmed directly under this illumination.

Reference
1. Debevec, P., Hawkins, T., Tchou, C., Duiker,H.-P., Sarokin, W., & Sagar, M. (2000). Acquiring the reflectance field of a human face. In *Proceedings of SIGGRAPH 2000.*


Light Stage 2.0, with seated subject.


A 10-second-exposure photograph of Light Stage 2.0 acquiring a 4D reflectance field dataset of the subject's face.

Lighting-Sensitive Displays

Contact
Shree K. Nayar
Computer Science
Columbia University
New York, New York 10027
USA
nayar@cs.columbia.edu

Peter N. Belhumeur
Electrical Engineering and
Computer Science
Yale University

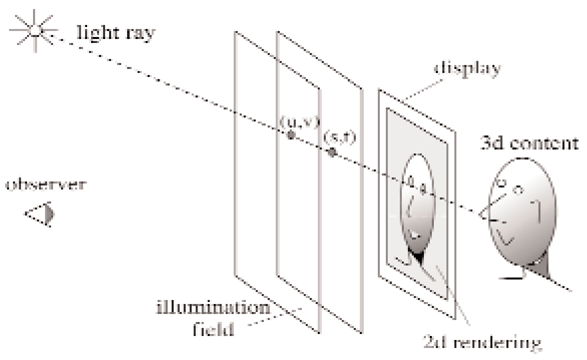
Terry Boult
Electrical Engineering and
Computer Science
Lehigh University

Displays have become a vital part of our everyday lives. They are used to convey information in a wide range of products including televisions, computers, PDAs, and cellular phones. Recently, high-quality digital displays have also emerged as possible replacements for physical media such as photographs, paintings, and sculptures.

Research on display technology has made great strides in improving the resolution, brightness, and color characteristics of displays. However, all display technologies suffer from a serious drawback: they are unable to respond to the wide spectrum of illumination conditions to which they are exposed. We introduce new technologies that enable a display to sense the illumination of its environment and render the appearance of its content accordingly.

Sensing a Display's Illumination Field

Today, the only lighting-related controls that the user has on a display are global brightness and color adjustments. These can be varied manually by the user, or automatically, using one or a few photodetectors.[1,2,3] Such adjustments are crude, as they only enable global brightness and color changes. The light field around a display (or for that matter, any object) is rather complex, and true compensation for environmental illumination requires a dense sampling and representation of the light field (see figure below) incident upon the display. To this end, we have developed a suite of sensing technologies that permit dense estimation of the illumination field in real time. These technologies include the arrangement of compact photosensitive arrays over and around the display device, the distribution of optical fibers around the device, and the placement of wide-angle imaging systems close to the device. In all cases, we obtain a dense spatial and/or directional sampling of the illumination field. In the system shown on the right, a very compact imaging system with a hemispherical field of view is embedded within the frame of the display device. The directional illumination distribution captured at the location of the imaging system is extrapolated over the surface of the display.

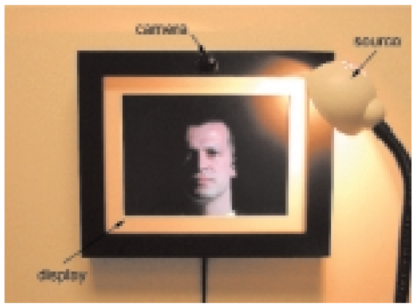


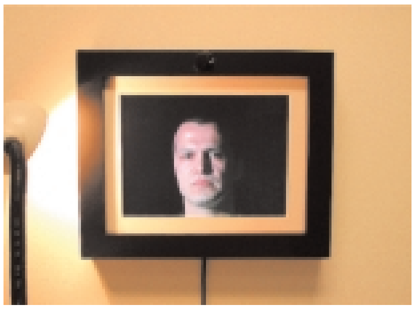
Rendering Illumination-Consistent Content

Once the illumination field as been measured, this information can be used to modify the visual content in several ways. The visual content can be either 2D images or 3D models of objects with arbitrary shapes and reflectance properties. These images and objects can be either purely synthetic, or data obtained by photographing/scanning real scenes, or some combination of the two. In the case of 2D images, the goal may be to spatially adapt the brightness of the image so that it appears to be lit by the environmental illumination. In the case of 3D scenes, the displayed image is rendered so that the scene is lit by the illumination of the environment. In both cases, a high degree of photo-realism is achieved. In the pictures shown below, the face is rendered based on the illumination field measured by the wide-angle camera attached to the display. As can be seen, the shadings and shadows in the image are consistent with the location of the light source. Since the illumination field is measured by the camera at video rate, the displayed content adapts to changing illumination conditions in real time.

Acknowledgements

This work was supported in part by an NSF ITR Award under Grant No. IIS-0085864. The authors thank Rahul Swaminathan at Columbia University and Melissa Koudelka at Yale University for their help in acquiring data related to this project.





References
1. Biggs, A.J. (1995). Compensator for color television receivers for chromacity variations in ambient light. United States Patent 3,200,193, August 1965.
2. Constable, D.W. (1978). Ambient light contrast and color control circuit. United States Patent 4,090,216, May 1978.
3. Heijligers, H. (1962). Circuit arrangement for automatically adjusting the brightness and the contrast in a television receiver. United States Patent 3,027,421, March 1962.

Linux for the Production Pipeline

Contact
Jeffrey Wike
Software Development Manager
DreamWorks Feature Animation
jwike@anim.dreamworks.com

Overview

The technology group at DreamWorks was given the task of lowering its capital budget, while at the same time providing increased computing power to address the ever-increasing creative appetite of our filmmakers. The decision was made to commit all new productions, beginning with a production that started in March 2001, to the all-Intel/Linux pipeline. The next question was: When would Linux be ready for prime time at the scale required to produce a full feature-length animated film? This technical sketch examines the challenges that we faced as the first major studio to successfully deploy an all-Intel/Linux production pipeline.

Most animation and visual effects production facilities have experimented with Intel/Linux solutions on some scale. Some have even successfully deployed departmental workstations and/or render-farm solutions. DreamWorks has, however, successfully deployed a complete Intel/Linux production pipeline consisting of hundreds of desktop and render-farm machines for all aspects of its traditional and CG pipelines. The challenges of early adoption on this scale included making the right hardware and software choices, effectively utilizing the open-source community, porting a large in-house code base, aligning third-party application support, and addressing support and training for our production users.

The Challenges

A significant challenge was aligning our hardware and software choices to get the OpenGL performance necessary for our demanding applications and users. It would not be good enough to simply meet the current performance expectations (most artists were working on high-end graphics workstations). We had to raise the bar. Selecting a graphics card that met our 3D rendering benchmarks provided several options. However, finding a card that could also support our image-playback requirement and provide support for hardware overlay planes (required for our internal code base) proved to be a challenge. Most vendors claimed to have Linux support but we quickly found that most of this support was shallow, and we were really in uncharted territory. We ultimately found support in a handful of key vendors who partnered with us to find our way through the maze of hardware, Linux kernels, OpenGL, X-Server, window managers, widget sets, desktops, and applications. Testing included rendering tens of thousands of frames on both our current and Linux platforms. The resultant images were compared, pixel-by-pixel, to access the impact of numerical-precision and compiler-optimization issues when differences occurred.

Our software and systems group also had various issues to deal with in making the transition. Configuration management, automated-machine building, and component installation became critical in order to keep pace with machine-deployment schedules. Core components of our infrastructure also had to be addressed to support Intel/Linux. Finding an adequate set of software development tools and solutions for source-code management, debugging, and optimization for our internal development staff also presented significant challenges.

The Results

We benchmarked our Linux against current platforms for interactive hardware renders, exercising moderate-to-heavy processor and graphics loads. We found that, on average, we could achieve nearly twice the performance at a fraction of the hardware cost. Based upon our success and results in driving toward the goal of our March 2001 production start, we were able to deploy into earlier productions to realize the benefit of the Intel/Linux solution almost a year ahead of our anticipated dates.

Intel/Linux is real for entertainment and will have a dramatic impact on studios small and large. We hope that this technical sketch can provide some insight into our journey and encourage others to follow and realize the potential of Intel/Linux for entertainment.

The Living Forest: The Story of a 3d Feature Film in Europe

Contact
Juan Nouche
DYGRA Films
Linares Riva S9
La Coruña 15005 Spain
cristobal@filloa.com

In this animation sketch, we explain the production process of "The Living Forest," a 3D animated feature film based on Wenceslao Fernández Florez' novel about life in the forests of Galicia. The movie was made in Spain by DYGRA Films, a 3D animation studio based in La Coruña. Buena Vista International Spain released the film in Spain in the summer of 2001. Other participants included MEGATRIX, Antena 3TV, Vía Digital, and TVG, and the production was supported by the European Union Media Programme, the Spanish Ministry of Culture, and the regional government of Galicia.

The Work Process

DYGRA Films created this film using Maya 2.5 and 3.0 from Alias|Wavefront. It also devoted considerable effort to research and development. After the character designer drew the characters, instead of immediately modelling them in a computer, we sculpted them in paste, so that each one had a distinct personality. Only then did the computer work start.

After the character was modelled, we added a skeleton and movement controls. The next stage was character texturization, in which the materials and colours were chosen. This task is all the more complicated because each character had to be handled separately. Insects cannot be texturized as trees are, and trees cannot be texturized as humans are. We had to make the characters credible and, at the same time, harmonize them with their surroundings.

We set two goals for this film:

1. To recreate the atmosphere of the Atlantic Galician forest.
2. To make the characters as expressive as possible.

The animation phase began when the programming engineer prepared the character interface to support the animator's work. This required building the controls that the animator used to move arms, hands, and other body parts without having to learn computer programming.

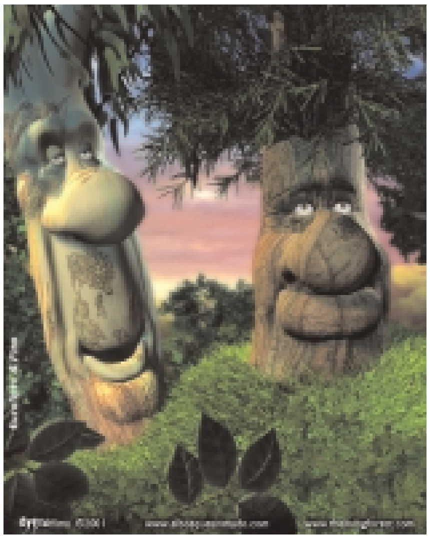
All the animation work was handled by a team of traditional animators who were able to use computers thanks to an exclusive animation interface, designed by the DYGRA Films R&D department, that lets the characters move like classic cartoon characters.

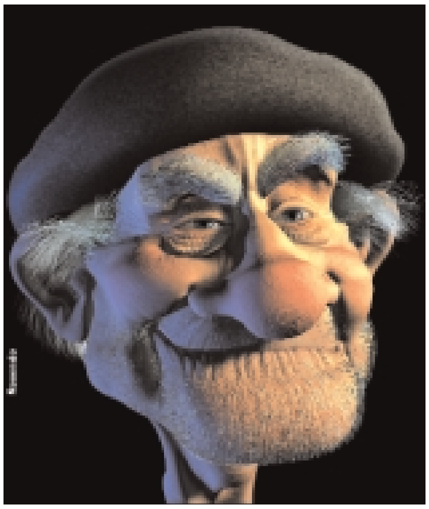
DYGRA has a multidisciplinary team of animators, writers, computer and telematics experts, physicists, sculptors, painters, and software engineers. Teamwork, in which each member's contribution is required to make the feature movie project a success, and computing combined to join the worlds of art and technology.

The Technology

The standard computer used for this production had a 550-MHz dual processor with double monitor and Tornado 3000 Open GL accelerators from Evans & Sutherland. The software in addtion to, Maya 2.5 and 3.0 from Alias|Wavefront, included Maya Fusion, Jaleo for post-production, and Real Flow and Real Wave from Next Limit for particles. Additional support was obtained from Videalab at Universidade da Coruña for R+D and CESGA for the render process.

See also: www.thelivingforest.com





Locally Reparameterized Light Fields

Xin Tong
Heung-Yeung Shum
Tao Feng
Microsoft Research China

Sing Bing Kang
Richard Szeliski
Microsoft Research

IBR techniques that have the most potential for photo-realism tend to be those that make use of densely sampled images. While using geometry helps reduce the database size, there is the problem of nonrigid effects such as reflection, transparency, and highlights. Without explicitly accounting for these effects, the light field sampling rate requirement would be even higher.

In this sketch, we propose a novel IBR representation to handle non-diffuse effects compactly, which we call locally reparameterized light field (LRLF). LRLF is based on the use of local and separate diffuse and non-diffuse geometries. The diffuse geometry is associated with true depth while the non-diffuse geometry has virtual depth that provides local photo-consistency with respect to its neighbors.

The Concept

The concept of LRLF can be explained using the Epipolar Plane Image (EPI).[1] An EPI is a 3D representation *(u,v,t)* of a stacked sequence of camera images taken along a path, with *(u,v)* being the image coordinates and *t* being the frame index. For a diffuse point on a scene with a linear camera path – Figure 1(a) – its trail within the *(u,t)* cross-section of the EPI is straight. The slope *k* of this trail is proportional to the depth *z* of this point. A non-diffuse point will also generate a trail: Figure 1(b). This non-diffuse point has an associated local virtual depth, which is often different from the actual object surface depth. In order to account for both diffuse and non-diffuse components, it is then necessary to provide separate depth compensations. The main idea of LRLF is to provide such local depth compensations in the form of local geometries.

Rendering

Our renderer is similar to the one described for the Lumigraph.[2] It uses the two-slab, 4D parameterization of light rays. As with the Lumigraph, each rendering ray is computed based on quadrilinear interpolation of rays from the nearest four sampling cameras using the local geometry for depth compensation. After each layer is separately rendered, the results are directly added to produce the output view.

Results and Future Work

Figures 2 and 3 show rendering results for a real scene involving strong reflective components. This scene consists of a picture frame with a toy dog placed at an angle to it on the same side as the camera. The input images are acquired using a camera attached to a vertical precision X-Y table that can accurately translate the camera to programmed positions. A grid of 9 x 9 images, each of resolution 384 x 288, was captured. The rendering resolution is also 384 x 288. The reflective component is extracted using the dominant motion-estimation technique.[3] As can be seen, the rendering results using the LRLF representation look markedly better than those obtained using just a single geometry. Each local geometry is approximated using a single plane.

Future extensions of this work include automatically computing all the non-diffuse effects and estimating separate geometries for different types of non-diffuse effects in the same scene.

References
1. Bolles, R.C., Baker, H.H., & Marimont, D.H. (1987). Epipolar-plane image analysis: An approach to determining structure from motion. *International Journal of Computer Vision, 1*, 7-55.
2. Gortler, S.J., Grzeszczuk, R., Szeliski, R., & Cohen, M.F. (1996). The lumigraph. *Proceedings of SIGGRAPH 96*, 43-54.
3. Szeliski, R., Avidan, S., & Anandan, R. (2000). Layer extraction from multiple images containing reflections and transparency. In *Conference on Computer Vision and Pattern Recognition,* I, 246-253.

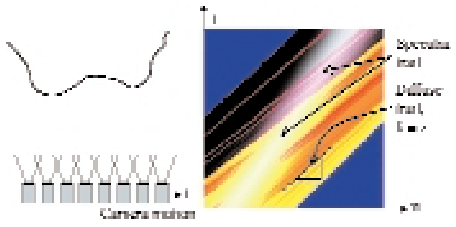


Figure 1. (left) Camera path, (right) EPI slice with highlight one-layer-middle.

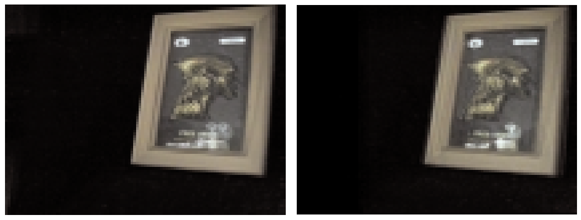


Figure 2. Two views rendered with a single local depth.

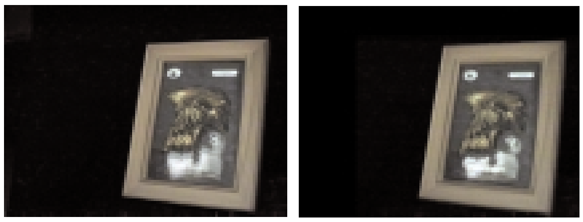


Figure 3. The same two views rendered using LRLF.

The Magic Behind the Magic: "Pearl Harbor"

Marshall Krasser
Compositing Supervisor
Industrial Light + Magic

For every spectacular shot that appears in a movie, there are untold scores of dedicated people who helped achieve its (hopefully) flawless execution.

At times, people have made the assumption that the computer has automatically created these visual wonders. I would like to take a moment and look at all of the phases of shot creation, not just the technical aspects.

Disciplines: model building, viewpainting, animating, matchmoving, lighting and rendering, plate photography, practical element photography, paint and rotoscoping, digital matte painting, and compositing.

This sketch presents video versions of "Pearl Harbor" shots that highlight how each discipline contributed to its creation. My goal is to step beyond the purely technical aspects of VFX creation and reveal more of the artistic and creative aspects. For example, in the case of compositing, we show a version of the shot that includes all of the elements prior to their integration into the plate. This reveals how artistic and visual talent is critical to the success of a shot.

Manipulate the Unreachable: Through-The-Lens Remote Object Manipulation in Virtual Environments

Contact
Stanislav L. Stoev
Universität Tübingen
sstoev@gris.uni-tuebingen.de

Dieter Schmalstieg
Technische Universität Wien

Wolfgang Strasser
Universität Tübingen

We have found that even though is does not have a counterpart in real life, remote object manipulation is both useful and intuitive. Our approach provides a solution to the problem of changing and examining the scene from the current viewpoint, while manipulating objects in distant locations of the virtual world. We achieve this with the aid of though-the-lens tools.

Through-the-Lens Concept

Through-the-lens (TTL) tools provide an additional viewpoint and display the scene in a dedicated viewing window. In this way, a preview window to a remote location is provided. In our semi-immersive setup, this window can be mapped onto a hand-held pad tracked with six degrees of freedom for convenient placement.[7] The pad becomes a magic lens[1,10] into a remote location. Previous remote object manipulation technologies such as Voodoo Dolls,[5] scaled-world grab,[2] or go-go,[6] allow viewing and manipulation either in the remote or the local environment, but not spontaneous combination of both. TTL remote object manipulation improves upon that by allowing both modes to be arbitrarily combined.

We have discussed elsewhere how the lens can be adjusted to show the desired remote location.[9] Once this is done, the user can manipulate remote objects that are visible through the lens, usually with the lens frozen in space rather than coupled to the hand-held pad. A tracked stylus is used for interaction with the remote objects. Similar to image-plane interaction methods,[4] the user can manipulate remote objects by reaching with the stylus into the frustum volume defined by the lens and the current viewpoint (see figure 1). If the stylus is outside this volume, it acts on the local environment in the normal way. Moving the stylus from the remote volume to the local volume and vice versa instantly changes the context of interaction, similar to the point-to-focus approach popular in some 2D windowing systems.

Moreover, this change of context can be exploited to teleport objects between locations by drag-and-drop operations between volumes. In a slightly more complex scenario, objects can be transferred between multiple remote locations with drag-and-drop operations. This application resembles some aspects of ToolSpaces.[3]

We have found TTL manipulations to be intuitive and efficient. The user is not required to navigate to the remote location in order to manipulate objects, but can stay at the current location and examine the result of the remotely performed actions. The proposed scenario is useful even if the “remote” location is within reach of the user, since scale and position of the remote view can be arbitrarily chosen. For example, a magnifying lens allows precise manipulation of details, while a minifying lens allows manipulation similar to using a world-in-miniature approach.[8]

References
1. Bier, E.A., Stone, M.C., Pier, K., Buxton, W., & DeRose, T.D. (1993). Toolglass and magic lenses: The see-through interface. In *Proceedings of SIGGRAPH 93*, 73-80.
2. Mine, M.R., Brooks, F.P., Jr., & Carlo H. Séquin. (1997). Moving objects in space: Exploiting proprioception in virtual-environment interaction. In *Proceedings of SIGGRAPH 97*, 19-26.
3. Pierce, J.S. (1999). Toolspaces and glances: Storing, accessing and retrieving objects in 3D desktop applications. In *Proceedings of the 1999 Symposium on Interactive 3D Graphics*.
4. Pierce, J.S., Forsberg, A.S., Conway, M.J., Hong, S., Zeleznik, R.S., & Mine, M.R. (1997). Image plane interaction techniques in 3D immersive environments. In *1997 Symposium on Interactive 3D Graphics*.
5. Pierce, J.S., Stearns, B.C., & Pausch, R. (1999). Voodoo dolls: Seamless interaction at multiple scales in virtual environments. In *Proceedings of the 1999 Symposium on Interactive 3D Graphics*, 141-146.
6. Poupyrev, I., Billinghurst, M. Weghorst, S., & Ichikawa, T. (1996). The go-go interaction technique: Non-linear mapping for direct manipulation in VR. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, 79-80.
7. Schmalstieg, D., Encarnação, L.M., & Szalavári, Z. (1999). Using transparent props for interaction with the virtual table (color plate p. 232). In *Proceedings of the 1999 Symposium on Interactive 3D Graphics*, 147-154.
8. Stoakley, R., Conway, M.J., & Pausch, R. (1995). Virtual reality on a WIM: Interactive worlds in miniature. In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems,* 265-272.
9. Stoev, S.L., Schmalstieg, D., & Strasser, W. (2001). Two-handed through-the-lens-techniques for navigation in virtual environments. In *Proceedings of the Eurographics Workshop on Virtual Environments*, 16-18 May 2001.
10. Viega, J., Conway, M.J., Williams, G., & Pausch, R. (1996). 3D magic lenses. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, 51-58.

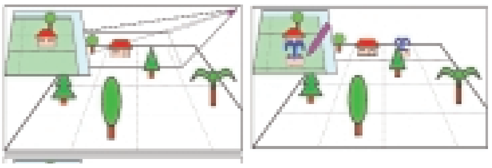
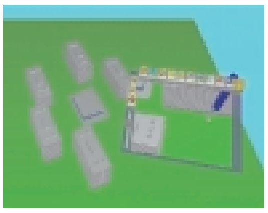
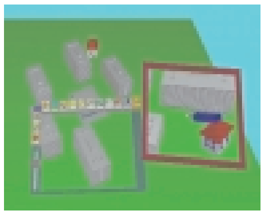


Figure 1: Schamatic drawing of the remote manipulation process. After defining the additional viewing window (left picture), one can manipulate the remote location through the window shown in the right picture.



First, a window (mapped on the pad) is defined through which the remote location is viewed.



After freezing the viewing window in space, two-handed manipulation is performed at the remote location.

Mapping-Independent Triangulation of Parametric Surfaces

Contact
Marco Attene
IMA-CNR
Via De Marini 6
Genova, Italy
attene@ima.ge.cnr.it

Geoff Wyvill
University of Otago

Triangulation of parametric surfaces is important for visualization and finite element analysis. The wrong choice of parameterization can spoil a triangulation or even cause the algorithm to fail. We present a new method that uses a local inverse mapping for almost uniformly sampling and triangulating a surface, so that its parameterization becomes irrelevant; differential geometry provides almost all we need.[1]

The Inverse Metric Tensor

The metric tensor, G, is a symmetrical 2 x 2 matrix whose elements are the coefficients of the first fundamental form. G transforms vectors, at a given point in parameter space, to the tangent plane at the corresponding point on the surface. The inverse, G^{-1} reverses this transformation. Given that G^{-1} exists, how can we sample the parameter domain so that the surface triangulation is as uniform as possible?

The "Normal Umbrella"

Given a point p_p in the parameter domain whose image in the surface is p, let $H = \{p_1, p_2 ..., p_6\}$ be a regular hexagon centered on p and lying in the tangent plane at p, Tp(p). For each p_i, draw a curve of arc length r on the surface from p to a point q_i so that its projection on Tp(p) is a straight line parallel to $v_i = p_i - p$. Connect each end-point, q_i, with its neighbor, q_{i+1}. This process defines the normal umbrella of radius r and center p which will be triangulated by the star $\{(p,q_1,q_2), (p,q_2,q_3), ...,(p,q_6,q_1)\}$.

To construct these curves we start at the point p_p and crawl in the direction $G^{-1}J^T\mathbf{v}_i$, where $J = [\mathbf{x}_u\ \mathbf{x}_v]$ is the Jacobian matrix of the surface's first derivatives. Notice that G and J vary as we proceed. Effectively, we are solving a differential equation defined by G and J at each point.

By definition, given a center point and a radius, the normal umbrella is independent of the parameterization, provided that the surface is smooth and regularly parameterized.

The Algorithm

1. Choose a random starting point, $p_p \in$ parameter domain.
2. The user chooses a radius, r, the expected average edge length in the final mesh.
3. Build the normal umbrella at p with radius r.
4. While there exists a vertex on the boundary of the mesh whose image in parameter space belongs to the required domain, complete its approximate normal umbrella.

To complete an approximate normal umbrella, centered on q_i, treat the triangles meeting at q_i as if they were already part of a normal umbrella about q_i. Project this part onto the tangent plane, compute the remaining angle, α, to be triangulated and divide it by an integer, n, so that α/n is as close as possible to 60 degrees. Finally, trace n-1 curves as described above to complete a set of triangles meeting at q_i with angle α/n.

E. Hartmann[2] proposed a similar marching method for implicit surfaces and it can also be used for parametric surfaces. However, it does not guarantee the desired edge length and requires two more numerical approximation steps (implicitization and point location) which can increase the error. Our method of computing lengths on the surface, rather than on the tangent plane, improves the accuracy of the mesh in regions of high curvature.

Work in Progress

Even if a uniform triangulation is suitable for some applications, it is not the best choice for interactive visualization. A mesh in which triangle size and shape varies depending on local surface behavior is preferable. We are working on the definition of a "distorted tensor" by which we can achieve a stretching and twisting of the local coordinate system; in particular we want the triangles to be stretched in proportion to the two principal curvature radii.[3]

References
1. Lipshutz, S. (1983). *Shaum's outline of differential geometry.* McGraw Hill, 1983.
2. Hartman, E. (1998). A marching method for triangulation of surfaces. *The Visual Computer* 14: 95-108.
3. Seibold, W. & Wyvill, G. (1999). Near-optimal adaptive polygonization. In *Proceedings CGI'99 IEEE Computer Society*, 206-213.

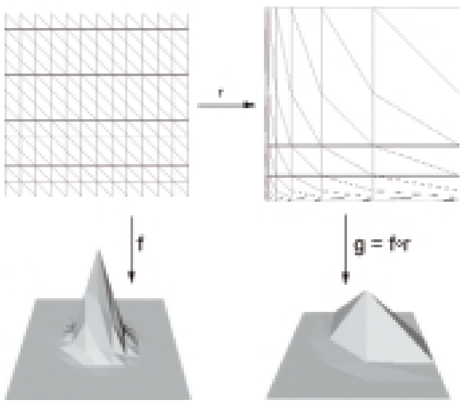


Figure 1. Example of "bad" parameterization. Uniform sampling of the parameter space does not imply uniform sampling in real space.

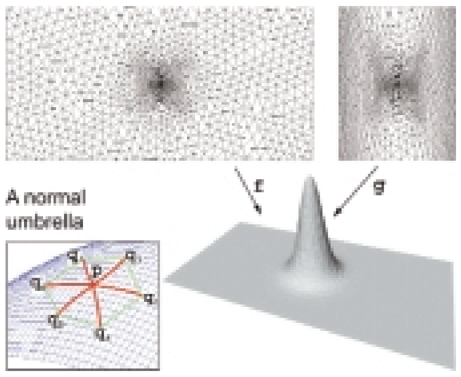


Figure 2. As the mapping changes, so does the triangulation in parameter space. The real-space mesh does not vary.

Mayhem: A New Trend for Virtual Environments (and Online Games)

Contact
Manuel Oliveira
University College London
Department Computer Science
Gower Street WC1E 6BT
London, United Kingdom
m.oliveira@cs.ucl.ac.uk

Jon Crowcroft
Mel Slater
University College London
Department Computer Science

Motivation
The ultimate aim of virtual environment (VE) research is to build a global networked infrastructure with high-resolution 3D graphics. However, current VE are confined to laboratory experiments, military simulations or are customized to carry out a specific task. In every case, the VEs are session based so user identity is short-lived. Online games and multi-user dungeons (MUDs) include social interaction as part of the core system design. This promotes the emergence of cybercommunities.

However, all the systems experience serious problems that prevent the achievement of the aforementioned goal. Two examples are: monolithic implementations -- this makes upgrades, adaptation, and evolution difficult; and scalability is limited which results in rudimentary mechanisms that compromise the maximum number of simultaneous users.

Description
We believe that most of the problems are related to how the problem domain has been addressed so far. Therefore we created the Mayhem project to test and develop new concepts and approaches in building VE and online games. At present, the project consists of a massive online role playing game (MORPG) that has adopted similar aims as the Internet:

- Heterogeneity. It should be possible to support different applications (either client or server, or both), each with their own particular set of quality of service (QoS) demands and capabilities.
- Scalability. The system should be able to scale independently of the content size, the number of players or the network infrastructure.
- Evolutionary. Any application should be connected to the infrastructure without requiring a restart. An existing application should not require to shutdown every time an upgrade is available or modification is required.

System Overview
The current system built for the Mayhem project uses hybrid architecture based on client/server, which is used to support persistency and non-realtime services to users, such as character creation, locating worlds, and configuring their client application.

The location of worlds resembles a naming service of downloadable components associated to each VE, so when a user chooses a particular world then their client downloads the specific code and necessary resources. This means the client application is built at runtime.

After the customized configuration has finalised, the client is ready for real-time interaction with the system. As a result, the system architecture shifts to being distributed, with occasional remote connections to a remote server whenever resources (content or code) are necessary.

As the client plays the game, various mechanisms are in place to assure scalability in terms of computational and network resources. At the heart of it all is the AOIModule, which is responsible for coping with one or more area of interest management (AOIM) policies.

Component Design
The cornerstone of our approach is a strong component oriented design, along with an application framework that is flexible enough to be partially or fully adopted.

We have built the Java Adaptive Dynamic Environment (JADE),[1] which is itself a component framework. The module represents the main building block of the framework, which defines the necessary interface for JADE to manage a component. A microkernel is responsible for the management of modules enforcing the chosen security policies.

With JADE, it is possible for a system to load a module at runtime, replace an existing one, or just use one of the operations available: activate, deactivate, reload, initialise, and shutdown. Thus an application may evolve over time. To aid the system development, a set of extension modules are provided, such as TreacleWell (TW),[2] which is responsible for providing any network connectivity and protocol composition.

Conclusions
VE and online games seem to bear little resemblance to each other when in fact the underlying objectives are similar. Unfortunately, the knowledge and experience garnered is hard to reuse in existing or new systems. The Mayhem project provides a new approach to VE, and consequently, games. Its goal is to provide a system with heterogeneity, scalability, and evolutionary properties to demonstrate our new trend. Our approach addresses the existing problems of current VE and online games.

References
1. Oliveira, M., Crowcroft, J. & Slater, M.. (2000). Component framework infrastructure for distributed virtual environments. In *Proceedings ACM CVE'00*.
2. Oliveira, M., Crowcroft, J. & Slater, M.(2000). Treaclewell: unraveling the magic "black box" of the network. Submitted for publication.

A New Method for Numerical Constrained Optimization

Contact
Ronald N. Perry
Mitsubishi Electric
Research Laboratories
perry@merl.com

Introduction

Numerical constrained optimization is an important tool with extensive applications in computer graphics[3] and many other diverse fields.[1] In computer animation, for example, constrained optimization has been used for finding initial conditions, smoothing or finishing motion paths, interpolating orientations, animating flexible bodies, self-assembly of objects, and finding boundaries.[3]

An ideal problem for constrained optimization is one that has a single measure defining the quality of a solution plus some requirements upon that solution that must not be violated. The quality measure is called the *objective function* (denoted as F below); the requirements are called *constraints* (denoted as C_i below). A constrained optimization solver maximizes (or minimizes) the objective function while satisfying the constraints. In this sketch, we propose a new method for constraint handling that can be applied to established optimization algorithms and which significantly improves their ability to traverse through constrained space. To make the presentation concrete, we apply the new constraint method to the Nelder and Mead polytope algorithm.[2] The resulting technique, called SPIDER, has shown great initial promise for solving difficult (e.g., nonlinear, nondifferentiable, noisy) constrained problems.

The Idea

Many constrained problems have optima that lie near constraint boundaries. Consequently, avoidance of constraints can hinder an algorithm's path to the answer. By allowing (and even encouraging) an optimization algorithm to move its vertices into constrained space, a more efficient and robust algorithm emerges. In the new method, constraints are partitioned into multiple levels. A constrained performance, independent of the objective function, is defined for each level. A set of rules, based on these partitioned performances, specify the ordering and movement of vertices as they straddle constraint boundaries; these rules (employing the insight stated above) have been shown to significantly aid motion along constraints toward an optimum. Note that the new approach uses no penalty function and thus does not warp the performance surface, thereby avoiding the possible ill-conditioning of the objective function typical in penalty methods.

Problem Statement

Maximize $F(\mathbf{x})$ for all $\mathbf{x} \in R^N$ such that $C_i(\mathbf{x}) \geq 0$ where $F{:}R^N \rightarrow R$, $C_i{:}R^N \rightarrow R$, and $i = [1...M]$. F or any C_i may be highly nonlinear, nondifferentiable, and/or *noisy*. A function (e.g., F) is considered noisy when repeated evaluation of that function with identical input yields different values. Figure 1 contains a detailed description of the SPIDER algorithm.

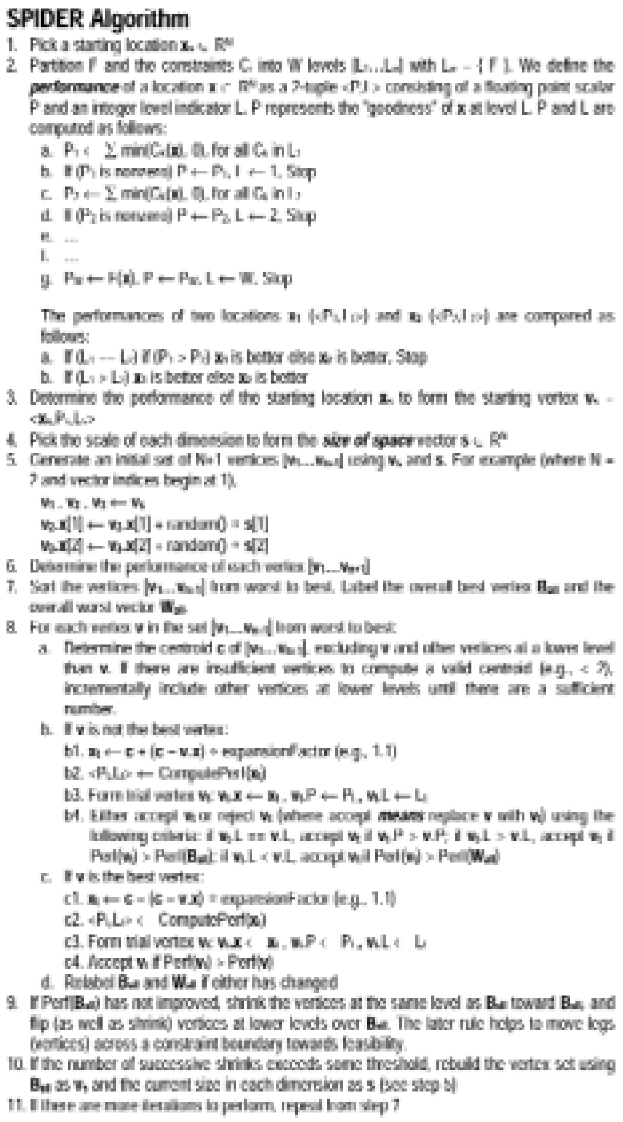
Notes and Extensions

1. One effective classification of constraints is to place simple limits on **x** that are independent of F into level 1, all other constraints into level 2, and the objective function F into level 3. Many different strategies for classification are being explored. *2.* SPIDER permits dynamic classification of constraints at anytime. This classification can be specified by a user (through observation of how SPIDER is moving) or by a classification algorithm which, for example, categorizes constraints into "active" and "inactive" levels. *3.* Other non-polytope optimization methods[2] can be modified to use the dynamic partitioned performances and corresponding rules introduced above, thereby improving their ability to traverse constrained space.

Summary

Partitioned performances permit many optimization algorithms, including SPIDER, to better traverse constrained space. The dynamic classification of constraints, either by a user or a classification algorithm, further enhances navigation through difficult terrain. Finally, the SPIDER algorithm, including the centroid computation, leg flipping when shrinking, and cycling through all legs before resorting, is a robust method for solving difficult constrained problems.

References
See www.merl.com/reports/TR2001-14



SPIDER Algorithm

```
1. Pick a starting location $\mathbf{x}_0 \in R^N$
2. Partition F and the constraints $C_i$ into W levels $[L_1 \ldots L_W]$ with $L_W = \{ F \}$. We define the *performance* of a location $\mathbf{x} \in R^N$ as a 2-tuple $<P,L>$ consisting of a floating point scalar P and an integer level indicator L. P represents the "goodness" of $\mathbf{x}$ at level L. P and L are computed as follows:
   a. $P_1 \leftarrow \sum \min(C_k(\mathbf{x}), 0)$, for all $C_k$ in $L_1$
   b. If ($P_1$ is nonzero) $P \leftarrow P_1$, $L \leftarrow 1$, Stop
   c. $P_2 \leftarrow \sum \min(C_k(\mathbf{x}), 0)$, for all $C_k$ in $L_2$
   d. If ($P_2$ is nonzero) $P \leftarrow P_2$, $L \leftarrow 2$, Stop
   e. ...
   f. ...
   g. $P_W \leftarrow F(\mathbf{x})$, $P \leftarrow P_W$, $L \leftarrow W$, Stop

   The performances of two locations $\mathbf{x}_1$ ($<P_1,L_1>$) and $\mathbf{x}_2$ ($<P_2,L_2>$) are compared as follows:
   a. If ($L_1 == L_2$) if ($P_1 > P_2$) $\mathbf{x}_1$ is better else $\mathbf{x}_2$ is better, Stop
   b. If ($L_1 > L_2$) $\mathbf{x}_1$ is better else $\mathbf{x}_2$ is better
3. Determine the performance of the starting location $\mathbf{x}_0$ to form the starting vertex $\mathbf{v}_0 = <\mathbf{x}_0,P_0,L_0>$
4. Pick the scale of each dimension to form the *size of space* vector $\mathbf{s} \in R^N$
5. Generate an initial set of N+1 vertices $[\mathbf{v}_1 \ldots \mathbf{v}_{N+1}]$ using $\mathbf{v}_0$ and $\mathbf{s}$. For example (where N = 2 and vector indices begin at 1),
   $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \leftarrow \mathbf{v}_0$
   $\mathbf{v}_2.\mathbf{x}[1] \leftarrow \mathbf{v}_2.\mathbf{x}[1] + random() * \mathbf{s}[1]$
   $\mathbf{v}_3.\mathbf{x}[2] \leftarrow \mathbf{v}_3.\mathbf{x}[2] + random() * \mathbf{s}[2]$
6. Determine the performance of each vertex $[\mathbf{v}_1 \ldots \mathbf{v}_{N+1}]$
7. Sort the vertices $[\mathbf{v}_1 \ldots \mathbf{v}_{N+1}]$ from worst to best. Label the overall best vertex $\mathbf{B}_{all}$ and the overall worst vector $\mathbf{W}_{all}$.
8. For each vertex $\mathbf{v}$ in the set $[\mathbf{v}_1 \ldots \mathbf{v}_{N+1}]$ from worst to best:
   a. Determine the centroid $\mathbf{c}$ of $[\mathbf{v}_1 \ldots \mathbf{v}_{N+1}]$, excluding $\mathbf{v}$ and other vertices at a lower level than $\mathbf{v}$. If there are insufficient vertices to compute a valid centroid (e.g., < 2), incrementally include other vertices at lower levels until there are a sufficient number.
   b. If $\mathbf{v}$ is not the best vertex:
      b1. $\mathbf{x}_t \leftarrow \mathbf{c} + (\mathbf{c} - \mathbf{v}.\mathbf{x}) * expansionFactor$ (e.g., 1.1)
      b2. $<P_t,L_t> \leftarrow ComputePerf(\mathbf{x}_t)$
      b3. Form trial vertex $\mathbf{v}_t$: $\mathbf{v}_t.\mathbf{x} \leftarrow \mathbf{x}_t$, $\mathbf{v}_t.P \leftarrow P_t$, $\mathbf{v}_t.L \leftarrow L_t$
      b4. Either accept $\mathbf{v}_t$ or reject $\mathbf{v}_t$ (where accept *means* replace $\mathbf{v}$ with $\mathbf{v}_t$) using the following criteria: if $\mathbf{v}_t.L == \mathbf{v}.L$, accept $\mathbf{v}_t$ if $\mathbf{v}_t.P > \mathbf{v}.P$; if $\mathbf{v}_t.L > \mathbf{v}.L$, accept $\mathbf{v}_t$ if $Perf(\mathbf{x}_t) > Perf(\mathbf{B}_{all})$; if $\mathbf{v}_t.L < \mathbf{v}.L$ accept $\mathbf{v}_t$ if $Perf(\mathbf{x}_t) > Perf(\mathbf{W}_{all})$
   c. If $\mathbf{v}$ is the best vertex:
      c1. $\mathbf{x}_t \leftarrow \mathbf{c} - (\mathbf{c} - \mathbf{v}.\mathbf{x}) * expansionFactor$ (e.g., 1.1)
      c2. $<P_t,L_t> \leftarrow ComputePerf(\mathbf{x}_t)$
      c3. Form trial vertex $\mathbf{v}_t$: $\mathbf{v}_t.\mathbf{x} \leftarrow \mathbf{x}_t$, $\mathbf{v}_t.P \leftarrow P_t$, $\mathbf{v}_t.L \leftarrow L_t$
      c4. Accept $\mathbf{v}_t$ if $Perf(\mathbf{v}_t) > Perf(\mathbf{v})$
   d. Relabel $\mathbf{B}_{all}$ and $\mathbf{W}_{all}$ if either has changed
9. If $Perf(\mathbf{B}_{all})$ has not improved, shrink the vertices at the same level as $\mathbf{B}_{all}$ toward $\mathbf{B}_{all}$, and flip (as well as shrink) vertices at lower levels over $\mathbf{B}_{all}$. The later rule helps to move legs (vertices) across a constraint boundary towards feasibility.
10. If the number of successive shrinks exceeds some threshold, rebuild the vertex set using $\mathbf{B}_{all}$ as $\mathbf{v}_0$ and the current size in each dimension as $\mathbf{s}$ (see step 5)
11. If there are more iterations to perform, repeat from step 7
```

Figure 1. The SPIDER algorithm.

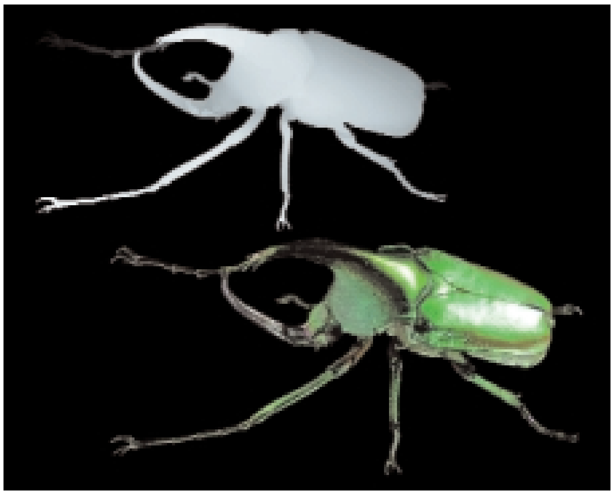
Micro Archiving

Contact
Tatsuya Saito
Keio University
tatsu@wem.sfc.keio.ac.jp

The experience provided by this technology will allow a visitor to visualize and interact with microscopic structures that cannot be seen with the naked eye, but that commonly exist in our everyday surroundings. Through the combination of Micro Archiving technology and Virtual Reality technology, we present immersive virtual environments in which people will be able to observe these microscopic structures in a private or collaborative workspace.

Micro-Presence is our term to describe the ability to experience these hidden realities. There are many terms that describe experiences of presence other than the real world in which we feel something through our sense organs directly. For example, Tele-Presence is a term to describe the technology that enables people to feel as if they are actually present in a different place or time. We define Micro-Presence as the environment in which people can feel as if they became minute and can observe and interact naturally with things in the Microcosmic world.

With the Micro Archiving technology, it is possible to create the high definition virtual 3D models. And these models can be fit for the academic research in such fields as biology and zoology that requires the actual things to observe. In addition, for educational use, this technology will create the high definition multimedia space in which visitors can freely participate and interact with the exhibit.



Model-Based Analysis and Synthesis of Human Body Images

Contact
Junichi Hoshino
University of Tsukuba
PRESTO, JST Science and Technology Corporation
jhoshino@esys.tsukuba.ac.jp

Generating digital actors is important for many applications such as special effects in movies, games, and virtual reality. In this sketch, we present a mode-based analysis and synthesis of human body images. First, we estimate the current 3D pose of the human figure by using the spatio-temporal analysis with kinematic constraints. Then, we store the intensity images of the human body as textures of the 3D body model. Such a human model can be used to generate multiple views, merge virtual objects, and change motion characteristics of human figures in video.

Generation of the Texture-Mapped Body Model

First, we need to register the 3D body model onto input images to collect the human body texture. We represent a human body by an articulated structure consisting of 10 rigid parts corresponding to body, head, upper arms, under arms, upper legs, under legs. The motion parameters between the image frames are estimated using the spatial and temporal gradient method.[1] One of the drawbacks in the previous method[1] is that the operator needs to give the initial position of the human body model. In our new method, the system estimates the initial position from the center line of each body part extracted from the silhouette. Personal variations in the 3D body model can be also adjusted by searching the size parameters that minimize the difference between the silhouette boundary and the body model. Such new extensions are also useful for video motion capture applications.

Interpolation of Missing Texture

The 3D body model supplements the whole-body textures that cannot be collected from a single camera view:

- Texture collection from multiple frames. The missing texture at one frame may be visible in other frames. When new intensity regions appear in the input image, we update the stored texture of the 3D body model.

- Supplementing missing textures. The missing textures cannot be identified until they become visible. However, convincing results may be obtained by using the texture of the other body parts. One simple method is to use the symmetry of the body parts and mirror the texure of the opposite side. Another alternative is to generate missing textures by CG objects. Body parts such as hair can be generated by using CG hairstyles. The color and the texture of the hairstyle can be extracted from the visible regions.

Model-Based Editing of Human Body Images

The texture-mapped 3D human model can be used for virtual view generation and model-based video editing:

- Merging virtual objects. Virtual objects (for example, CG cloth and hairstyle) are added to the texture-mapped 3D model of the human body. We render the texture-mapped body model and CG objects together to obtain a synthesized image.

- Converting motion characteristics. Because the body pose can be obtained at each frame, we can change the behavior by replacing the pose values. For example, we can change the mass and inertia of the body model and simulate the motion by using standard computer graphics techniques. We can also replace the pose parameters with motioncapture data of a different person.

Example

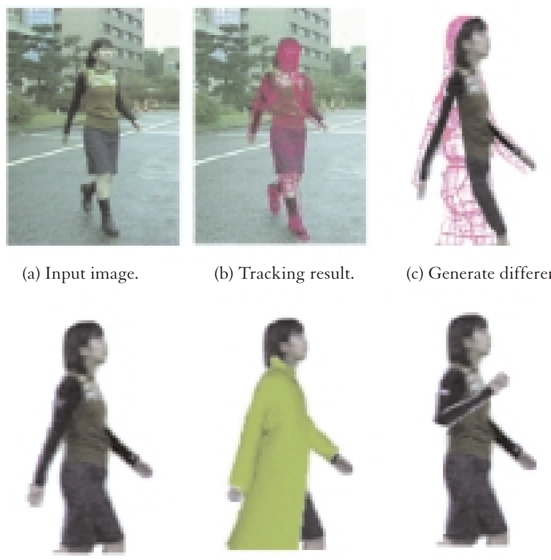
Figure1 illustrates model-based analysis and synthesis of human body images: (a) is the typical input image; (b) is the result of the automatic registration; (c) is the side view with the human body texture from the input image.

Because we can obtain partial texture in (a), we cannot recover the whole view. We estimate the missing texture by assuming that the intensity distribution is symmetric around the body axis.

Hairstyle was generated by CG; (d) shows interpolation of missing texture (when the textures within body parts are uniform, these simple techniques can generate a realistic view); (e) is the result of merging CG cloth from the side view; (f) is an example of changing motion characteristics by replacing the pose parameters of the body model.

Reference

1. Hoshino, J., Saito, H., & Yamamoto, M. (2000). Match moving technique for human body images. *SIGGRAPH 2000 Conference Abstracts and Applications*.



(a) Input image. (b) Tracking result. (c) Generate different view.

(d) After estimating missing texture. (e) Merging CG cloth. (f) Changing pose.

Figure 1. Result of model-based synthesis of human body images.

Modeling and Dynamics of Clouds Using a Coupled Map Lattice

Contact
Yoshinori Dobashi
Hokkaido University
doba@nis-ei.eng.hokudai.ac.jp

Tomoyuki Nishtia
Ryo Miyazaki
Satoru Yoshida
The University of Tokyo

Introduction

Clouds play an important role in creating images of outdoor scenes. This abstract proposes a new method for modeling various kinds of clouds. The shape of clouds is determined by atmospheric fluid dynamics. This implies that various types of clouds can be modeled by a physically based simulation of cloud formation. Our method simulates cloud formation by using a method called the coupled map lattice (CML). The proposed method can create various types of clouds and can also realize the animation of these clouds.

Classification of Clouds

Clouds can be classified roughly into two types based on their formation processes. Clouds in the first type are known as cumuliform (cumulus and cumulonimbus), and they are formed by ascending air currents. These air currents are generated by various mechanisms such as temperature rising on the ground. Clouds in the second category are generated by the cooling down of vapor that flows upward from the ground. In particular, the patterns observed in the cell-like shapes of stratocumulus, altocumulus, and cirrocumulus are generated by the Bénard convection.[1]

Overview of Our Method

Our method simulates cloud formation using a simplified numerical model, called CML. The advantages of using CML are that it is easy to implement and the computational cost is small. Figure 1 shows our system for cloud modeling-rendering. Firstly, the user specifies the types of clouds desired and the conditions for the simulation, i.e. the parameters for the atmospheric properties such as the viscosity and diffusion coefficient. The cloud formation is simulated based on the specified conditions. During the simulation, the distribution of the state values such as temperature, velocity and density of the clouds are visualized at every time step. The user can change the simulation parameters interactively to control the shape of the clouds.

Simulation of Cloud Formation Using CML

CML is an extension of cellular automaton, and the simulation space is subdivided into lattices. Each lattice has several state variables, and their status is updated depending on the variables on the adjacent lattices.The cloud formation processes are summarized as follows: clouds are formed as a bubble of air is heated by the underlying terrain, causing the bubble to rise into regions of lower temperature; the phenomenon known as phase transition then occurs, when water vapor in the bubble becomes water droplets, or clouds. Therefore, our method takes into account the following factors: viscosity and pressure effects, the advection by the fluid flow, diffusion of water vapor, thermal diffusion, thermal buoyancy, and the phase transition. For generating cumulus or cumulonimbus clouds, the distribution of the ascending current is specified, and then the velocity distribution is calculated by using CML method. We also input the humidity distribution and the humid air moves upward due to the ascending air currents. Then, clouds are formed by the phase transition. If we want to generate the cell-like cloud formations we simulate the Bénard convection by inputting the difference in the temperature in the vertical direction and, clouds are generated by using the calculated velocity field.

Result

Figure 2 shows images of clouds generated by our method. Figures (a) and (b) are the cumulonimbus and the altocumulus, respectively. In Figure (a), the tower-like cloud is generated by the strong ascendant current. In Figure (b), the sky is dotted with relatively small clouds. The numbers of lattices for Figures (a) and (b) are 178 x 178 x 98 and 256 x 256 x 10, respectively. The computation times per time step of the simulation are five and two seconds, respectively, for Figs. (a) and (b). Rendering times are less than one minute by using the method developed by Dobashi et al.[2]

References

1. Houze, R.A. (1993). Cloud dynamics. *International Geophysics Series* Vol. 53, Academic Press, New York, 1993.
2. Dobashi, Y., Kaneda, K., Yamashita, H., Okita, T., Nishita, T. et al.(2000). A simple, efficient method for realistic animation of clouds. In *Proceedings SIGGRAPH 2000,* 19-28.

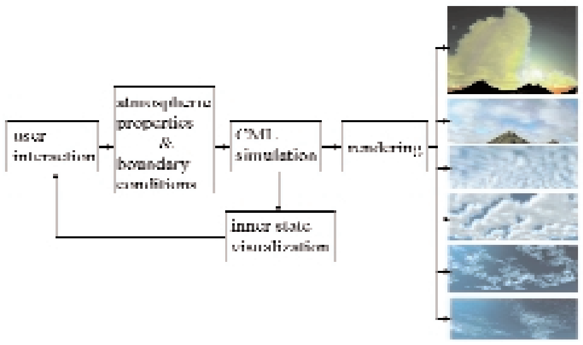

user interaction
atmospheric properties & boundary conditions
CML simulation
rendering
inner state visualization

Figure 1. Our system for modeling and rendering clouds.

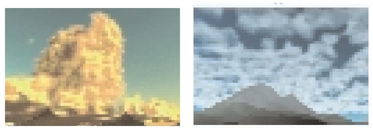


Figure 2. Simulation of clouds: (a) cumulonimbus, (b) altocumulus.

The Mouthesizer: A Facial Gesture Musical Interface

Contact
Michael J. Lyons
ATR Media Integration &
Communication Research Labs
2-2-2 Hikaridai, Seika-cho
Kyoto 619-0288 Japan
http://www.mic.atr.co.jp/~mlyons

Contributors
Michael Haehnel
NobujiTetsutani

Introduction

In this technical sketch we introduce a new interaction paradigm: using facial gestures and expressions to control musical sound. The mouthesizer, a prototype of this general concept, consists of a miniature head mounted camera which acquires video input from the region of the mouth, extracts the mouth shape using a simple computer vision algorithm, and converts shape parameters to MIDI commands to control a synthesizer or musical effects device.

Face and Gesture Musical Interfaces

Leading the edge of musical technology innovation are the efforts to interface expressive gestures to sound synthesis and modulation.[1,2] The face is a salient source of non-verbal communication conveying information about attention, intention, and emotion through facial expressions and gestures. The lower face is also critical to speech production through the action of the lips, tongue, and jaw. It therefore seems natural to us to link expressive action of the face to the control of musical sound.

Technology of the Mouthesizer

Our system consists of a wearable, lightweight, head mounted camera (Figure 1). The open area of the mouth is segmented by intensity and color thresholding and the largest blob is selected. The mouth opening is not a surface so the segmentation algorithm is robust over a wide range of illumination conditions. Statistical shape analysis is used to extract parameters proportional to the width and height of the mouth opening, which are normalized and output in MIDI format to a synthesizer-effects unit. The system runs at 30 fps on a notebook computer. Experimentation has revealed several intuitive shapesound mappings. The video demo at the site below implemented mouth-controlled guitar effects of wahwah and distortion (Figure 2). Performers report that the mouth controller is more intuitive and versatile than a foot pedal and that it feels good to use. Another example uses mouth aspect ratio to audio morph between three formant filters corresponding to vowel sounds [i], [a], and [o]. These audio effects can be applied to synthesizer patches or analog signals. Future work will consider other regions of the face and further explore the rich space of gesture-sound mappings. The research is targeted both at musical performers, who have their hands busy playing an instrument, and the handicapped, who may lack control of their limbs due to spinal damage, but who retain control of their facial muscles.

A short video is available at:
http://www.mic.atr.co.jp/~mlyons/mouthesizer.html

References
1. Wanderley, M. & Baffier, M., eds. (2000). Trends in gestural control of music, CD-ROM. IRCAM, Paris, 2000.
2. Cutler, M., Robair, G., & Bean. (2000). The outer limits: a survey of unconventional musical input devices. *Electronic Musician*, August 2000: 50 - 72.

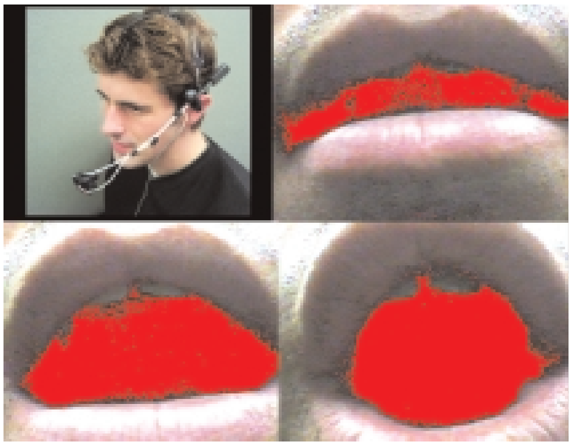


Figure 1. The head mounted video camera with views of the vowels [i], [a], and [o].The segmented area is shown in red.

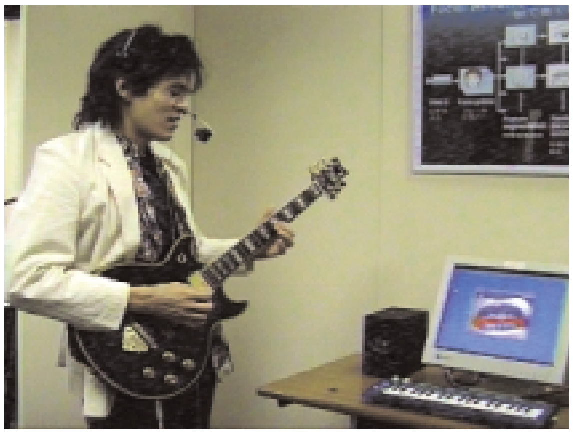


Figure 2. Guitarist Ichiro Umata using the Mouthesizer.

Multi-Modal Translation System by Using Automatic Facial Image Tracking and Model-Based Lip Synchronization

Shin Ogata
Takafumi Misawa
Faculty of Engineering
Seikei University

Kazumasa Murai
Satoshi Nakamura
Shigeo Morishima
ATR Spoken Language
Translation Research Labs

This sketch introduces a multi-modal English-to-Japanese and Japanese-to-English translation system that also translates the speaker's physical speech motion by synchronizing it to the translated speech.

Current speech translation systems transmit verbal information, but they cannot translate and transmit articulation and intonation. Although verbal information is a key element of human communication, facial expression also plays an important role in face-to-face communication. If we could develop a technology that is able to translate facial motions synchronized to translated speech, we could construct a multi-lingual tool that would be much more natural than existing systems.

Overview of System

Figure 1 shows an overview of the system developed in this research. The system is divided broadly into two parts: speech translation and image translation. The speech-translation part is composed of ATR-MATRIX,[1] which was developed at ATR-ITL. In the speech-translation process, the two parameters of phoneme notation and duration information are applied to facial image translation. These two parameters are outputs from CHATR[2] that generate synthesized speech.

The first step of the image-translation process is to make a 3D model of the mouth region for each speaker by fitting a standard facial wireframe model to an input image. The second step is face-tracking processing by template matching. Tracking motions of the face from video images is one of the key technologies required for translating images. The third step in image translation is to generate lip movements for the corresponding utterance. The 3D model is transformed by controlling the acquired lip-shape parameters so that they correspond to the phoneme notations from the database used in the speech-synthesis stage. Duration information is also applied and interpolated by linear interpolation for smooth lip movement. In the final step of image translation, the translated synthetic mouth region's 3D model is embedded in input images. In this step, the 3D model's color is adjusted to the input images. Even if an input video image is moving during an utterance, we can acquire natural synthetic images because the 3D model has geometry information. Consequently, the system outputs a face movie lip-synchronized to the translated synthetic speech and image sequence at 30 frames/second.

Automatic Face Tracking

Tracking by template matching can be divided into three steps. First, texture mapping of one of the video-frame images is carried out on the individual 3D face-shape model. Here, a frontal face image is chosen from the video-frame images for the texture mapping, since the video images used in this experiment are mainly frontal in movement and rotation. Next, we make a 2D template image for each movement and rotation by using a 3D model. Here, in order to reduce matching errors, the mouth region is excluded from the template images. Consequently, even while the person in a video image is speaking, tracking can be carried out more stably. Finally, we carry out template matching between the template images and an input video-frame image, and estimate movement and rotation values so that the matching error becomes minimum.

Generation of Lip Shape and Movement

To prepare a lip-shape database, we used reference lip-shape images from the front and side. Then, we transformed the wireframe model to approximate the reference images. In this process, we acquired momentum vectors of lattice points on the wireframe model. Then, we stored these momentum vectors in the lip-shape database. This database is normalized by the mouth region's size, so we do not need speaker adaptation. We classified English and Japanese phonemes into 28 types based on visemes (a viseme is the smallest linguistic visual sound unit). Lip shapes composed of visemes are decided by the phonemic notation outputs from CHATR. And lip movement is generated by linear interpolation of momentum vectors located on keyframes according to phoneme duration time, also from CHATR.

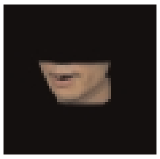
Summary

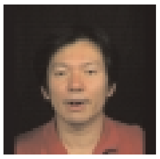
As a result of this research, the proposed system can create any lip shape with an extremely small database, and it is also speaker-independent. Furthermore, the system retains the speaker's original facial expression by using input images other than those of the mouth region.

For further improvement of this system, we need to design a model that expresses images more precisely. At the same time, we must improve the method for linear interpolation of the li shape. Furthermore, higher speed is needed to operate the system online.

References
1. Takezawa, T., Morimoto, T., Sagisaka, Y., Campbell, N., Iida, H., Sugaya, F., Yokoo, A., & Yamamoto, S. A. (1998). Japanese-to-English speech translation system ATR-MATRIX. *Proceedings International Conference of Spoken Language Processing, ICSLP*, 957-960.
2. Campbell, N. & Black, A.W. (1995). CHATR : a multi-lingual speech re-sequencing synthesis system. *IEICE Technical Report, sp96-7*, 45.
3. Ogata, S., Nakamura, S., & Morishima, S. (2001). Multi-modal translation system - model based lip synchronization with automatically translated synthetic voice. *IPSJ Interaction 2001*, 203.

(a) Input image.


(b) Mouth-region model.


(c) Translated image.

A Multipurpose Array of Tactile Rods for Interactive eXpression

Contact
Dan Overholt
Media Laboratory
Massachusetts Institute of Technology
dano@media.mit.edu

Egon Pasztor
Ali Mazalek
Massachusetts Institute of Technology

Introduction

The MATRIX (Multipurpose Array of Tactile Rods for Interactive eXpression) is a device that offers real-time control of a deformable surface, enabling the manipulation of a wide range of audiovisual effects. The interface functions as a versatile controller that can be adapted to many different tasks in a variety of application domains. It was first used as a new type of musical instrument in the Emonator project.[1] New domains are now being explored in areas such as real-time graphics animation, sculptural design and rendering, still and moving image modification, and the control of physical simulations.

The Interface and Technology

The human hand has shaped and produced many amazing things over the centuries. It is one of the most nimble and dexterous tools at our disposal. One of the goals of the MATRIX project was to create an interface that would make the most of our skill with our hands. The result was a human-computer interface that can extract useful information from a gesture, exploiting the senses of touch and kinesthesia. It allows a computer to respond in real time to the form of a 3D surface, and captures the shape and movement of the hand at high resolution, taking advantage of the skills we have developed through a lifetime of interaction with the physical world.

The MATRIX interface consists of 144 rods which move vertically in a 12 by 12 grid, and are held up by springs at rest. The device uses this bed of rods to provide a large number of continuous control points, thereby improving the communication bandwidth with the computer. The deformation of the surface is determined using opto-electronics, and a FPGA (Field Programmable Gate Array). A technique called quadrature encoding is used to derive the current rod positions. All resulting positions are sent to the computer at a frame rate of 30 Hz, letting the system respond quickly and smoothly to a users input.

Visual Applications

The surface of the MATRIX can be used as a controller for any real-time graphics application where dynamic user input is required. Its responsive model of the hand provides a much richer method of control than a mouse, joystick, or keyboard. For example, the animation of ocean waves is traditionally done using a combination of physical simulation and careful positioning of isolated control points. Using the MATRIX, the motion of a user's hand can be mapped directly to the underlying forces in a wave simulation. The user would then have the experience of interactively creating the simulated ocean swells. This technique can also be applied to many other physical simulations for things that behave as particle systems, such as wind flow or the rippling of cloth.

Currently the MATRIX is being used as an input device for real-time graphics animation, image manipulation, and physical simulation. Figure 2 shows several examples of different tasks using the MATRIX. The upper left shows the rods of the device as perceived by the computer. These rods are interpreted as a shaded surface, which is shown on the upper right. This same surface can be used to deform virtual objects or images, such as the picture of the dog on the lower left. The surface can also be interpreted as an array of wind forces in a simulation of a grassy plain, as shown in the final image. As the user moves their hand, the blades of grass bend back and forth in response to the virtual winds.

In addition to these applications, the MATRIX is being used for the dynamic control of colored lights. The transparent acrylic rods serve as light guides for four computer controlled ColorKinetics C-30 lights[2] placed beneath the device. Colors and patterns of light are synchronized to the movement of the rods based on different mappings. For example, the average value and rate information of each quadrant of rods can be used to determine the color and intensity of each light.

Future Extensions

We are beginning to use the MATRIX for real-time processing of an incoming video stream using effects such as stretching, morphing, and other visual deformations. In the future, we plan to investigate using the interface to sculpt complete 3D objects by allowing users to change the orientation of the object in space and shape each of its sides individually.

Acknowledgements

We would like to thank Professors Tod Machover, Glorianna Davenport, and Joe Paradiso, as well as our colleagues in the Hyperinstruments and Interactive Cinema groups.

See also: www.media.mit.edu/~dano/matrix/

References
1. Overholt, D. (2000). The Emonator: a novel musical instrument. MIT Media Laboratory master's thesis, 2000.
2. ColorKinetics lights: URL: www.colorkinetics.com/

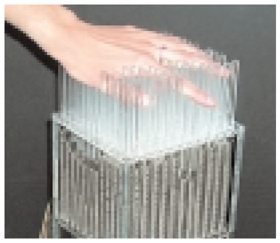


Figure 1. User interacting with the MATRIX interface.

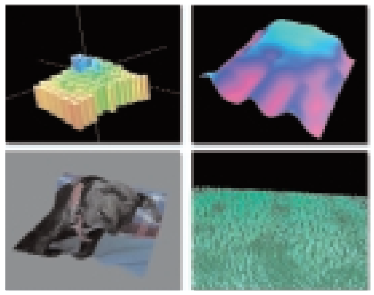


Figure 2. Example surface renderings generated by the MATRIX interface.

Multi-resolution depth buffer

Contact
Eugene Lapidous
ViewSpace Technologies Inc.
eugene@viewspace.com

Jianbo Zhang
Timothy Wilson
Trident Microsystems Inc.

One of the important factors affecting design of 3D applications is what depth precision is available across the view volume. Currently, the common denominator is a 24-bit screen Z buffer, which loses roughly log2(Zf/Zn) bits close to the far plane (Zf/Zn is a ratio of the distances to the far and near planes). More precise depth buffer algorithms[1,3], even when available, remain unused by the applications optimized for screen Z. To make high-precision depth buffers more popular, and therefore more widely supported, they have to benefit the majority of 3D applications.

We present new type of the multi-resolution depth buffer that allows applications optimized for screen Z to take advantage of the hardware support for high-precision depth buffer algorithms. It provides a flexible trade-off between precision and performance, easily adjustable in software.

Our observation is that in the open environments, or when camera is close to the ground, precision of the 24-bit screen Z-buffer at the far plane is often lower than precision of 16-bit depth buffers that remain linear or quasi-linear across the view volume. One such example is W-buffer, where stored depth is proportional to the distance from the camera. Figure 1 compares renderings of the human face using 24-bit screen Z buffer (Figure 1a) and 16-bit W- buffer (Figure 1b). The model is positioned close to the far plane at Zf/Zn = 4000; 24-bit screen Z-buffer fails to resolve objects behind the face mask (eyes, teeth), while 16-bit W-buffer renders them correctly.

Obviously, the 16-bit W-buffer cannot be used instead of 24-bit screen Z: when the object is close to the near plane, screen Z precision can be equal or larger than 24-bit; 3D applications can use it by keeping ratios Zf/Zn relatively low or by rendering complex objects close to the near plane. On the other hand, 3D applications cannot count on the depth precision higher than 24 bits, because the 24-bit W-buffer is currently a quality standard for 3D accelerators with 24-bit depth buffers.

We propose to leverage advantages of linear and quasi-linear depth buffers by dynamically switching resolution from 24 to 16 bits, so that the resulting precision is always kept at or above screen Z level (as long as it does not exceed 24 bits). As shown in Figure 2 for the dynamic ratio Zf/Zn = 1000, a switch from 24-bit to 16-bit W-buffer at half of the distance between far and near planes will be enough to support any application optimized for 24-bit Z. The share of the view volume where a 16-bit W-buffer can be used grows roughly as a cube of the distance between the switching point and the far plane, providing significant bandwidth savings.

It is important to note that a decision to use 16 or 24 bits can be based solely on the newly computed depth value and before the old one is read from the depth buffer. If the new distance from the camera is farther than the threshold, we'll read and write 16 bits of the corresponding depth value per pixel; otherwise we'll read and write 24 bits/pixel. This allows for a true multi-resolution depth buffer: no flags are necessary to define per-pixel resolution, because it is defined by the data itself.

The resolution switch threshold depends on the ratio Zf/Zn and can be set at the same time as a view matrix. Figure 3 shows how this threshold changes for three different types of depth buffers: fixed-point W (24->16); floating-point complementary Z with 4-bit exponent (4.20 -> 4.12); and floating-point 1/W with 4-bit exponent (4.20 -> 4.12). While multi-resolution 1/W is useful only if Zf/Zn > 5000, the resolution switch for complementary Z and fixed-point W may provide bandwidth savings for Zf/Zn > 500..1000. Complementary Z is significantly easier to implement than fixed-point W: it is a (1 - screen Z) value stored in the floating-point format.[1]

In conclusion, the proposed multi-resolution depth buffer allows 3D applications to improve performance by relaxing depth precision requirements with an increase of the distance from the camera. This optimization, already useful for legacy 3D hardware, does not have to be abandoned for the next generation of depth buffers. A tradeoff between precision and performance can be finely tuned by changing the resolution switch threshold.

References
1. Lapidous, E. & Jiao, G. (1999). Optimal depth buffer for low-cost graphics hardware. In *Proceedings. of SIGGRAPH 99/Eurographics Workshop on Graphics Hardware,* 67-73.
2. Deering, M.F.. US Patent 6046746: Method and apparatus implementing high resolution rendition of Z-buffered primitives.
3. Microsoft Corp. (2000). What are depth buffers? DirectX8 SDK, 2000. URL: msdn.microsoft.com/library/psdk/directx/DX8_C/hh/directx8_c/_dx_what_are_depth_buffers_graphics.htm

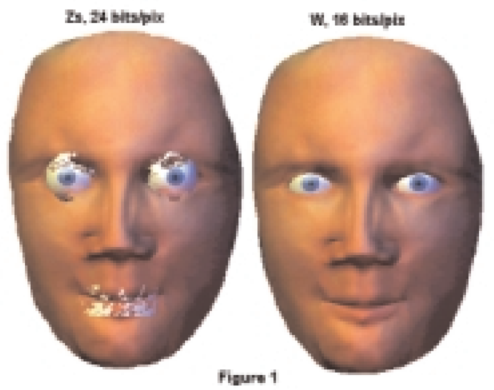


Figure 1

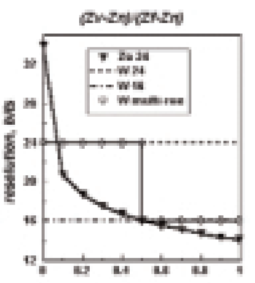


Figure 2

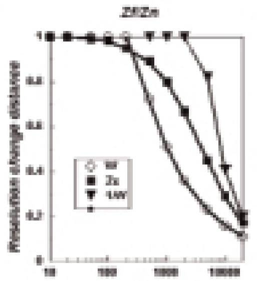


Figure 3

Multi-Resolution Light Field Rendering with the Wavelet Stream

Contact
Ingmar Peter
WSI/GRIS
Universität Tübingen
peter@gris.uni-tuebingen.de

Wolfgang Strasser
WSI/GRIS
UniversitätTübingen

Introduction

In this sketch we describe the wavelet stream, a new data structure for progressive transmission, storage, and rendering of compressed light field data. It employs non-standard 4D wavelet decomposition to make use of the coherence in all of the four dimensions of a light field data set. Compression ratios of up to 1:200 were obtained still providing visually pleasing results. Furthermore our data structure allows for an adaptive multi-resolution representation of the data. In contrast to similar approaches, the wavelet stream can be decompressed in real time for rendering and thus allows for use in interactive graphic applications.

Light Field Rendering

Image based rendering (IBR) tries to overcome the limitations of traditional geometry based computer graphics by replacing geometrical with pictorial information. One of the most general image based scene representations is the light field,[1,2] which can be used to represent arbitrary real or synthetic objects. It is independent as well from the objects geometry as from its material properties.

A light field is formed by a 2D array of images of an arbitrary object where all images share the same image plane. Using a parameterization as proposed in,[1,2] all light field samples can be addressed by a 4D coordinate (s,t,u,v). Unfortunately a large number of samples is required to create a light field which allows for a visually pleasing reconstruction of new views of arbitrary objects, overlaid with cross hairs after program.

Wavelet-Based Light field Compression

During compression the light field data is decomposed into one scaling coefficient and a number of detail coefficients employing non-standard 4D wavelet decomposition with the Haar-basis. Detail coefficients with very low values are discarded. The positions of the surviving coefficients are encoded using significance maps.

To support progressive storage, transmission, and rendering of light field data, the detail coefficients are stored together with the significance maps and some meta information for navigation in a byte stream. The data is ordered in a way, that the most important (i.e. with a high impact on the result) coefficients are encoded first. In this way an approximation of the light field can be rendered even if only a small fraction of the wavelet stream is transmitted.

Silhouette Encoding

Usually solitary objects are encoded in light fields. Unfortunately, due to the border's high frequency it causes the generation of a large number of detail coefficients during wavelet decomposition. This can be avoided by encoding the silhouette of the object. In this way the number of coefficients sufficient to encode the objects appearance is bounded by the number of samples with values belonging to the object.

Our silhouette encoding is fully embedded into the wavelet stream. By exploiting coherence, for typical data sets about 10 percent of the entire data have to be used to encode the silhouette information. On the other hand, the utilization of silhouette information reduces the number of coefficients about 20-30 percent while obtaining the same compression error compared to data sets compressed without using silhouette information.

Rendering

During light field rendering a texture map showing the new light field view is created and displayed using Open GL. The reconstruction algorithm is completed by supplementary image based caching schemes. Three different cache levels allow for reconstruction and rendering of 10-15 frames/second on standard PC-hardware.

Results

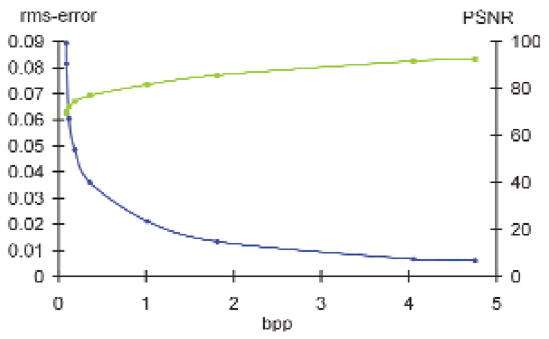
Our approach improves on existing wavelet-based compression schemes since it allows for:

- interactive reconstruction times;
- high compression ratios while preserving image quality;
- additional silhouette encoding to reduce the number of coefficients significantly.

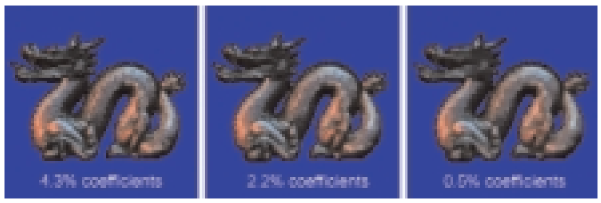
Some results are given showing the Dragon light field and respective error measurements.

References

1. Gortler, S., Grzeszczuk, R., Szeliski, R., & Cohen, M. (1996). The lumigraph. In *Proceedings SIGGRAPH 96.*
2. Levoy, M. & Hanrahan, P. (1996). Light field rendering. In *Proceedings SIGGRAPH 96.*
3. Peter, I. & Strasser, W. (1999). The wavelet stream: progressive transmission of compressed light field data. In *IEEE Visualization 1999 Late Breaking Hot Topics.*



PSNR and rms-error measured for different compression ratios of Dragon light- field.



Comparison of Dragon Light Field from The Stanfords Light Field Archive for different compression ratios.

My Finger's Getting Tired: Interactive Installations for the Mind and Body

Contact
Don Ritter
Pratt Institute
Computer Graphics and
Interactive Media
Brooklyn, New York 11205 USA
ritter@aesthetic-machinery.com
http://aesthetic-machinery.com

Introduction

This sketch presents an overview of Ritter's large scale interactive video and sound installations which use interactivity as content, and require viewers to use their entire bodies to control unencumbered interactive experiences for multiple users simultaneously.

Unencumbered Interactivity for the Entire Body

When experiencing interactive art as a cultural medium, the goal is to create an overall aesthetic experience. The use of any encumbered device, such as a mouse, can detract from this experience because the physical motions required by an encumbered input device rarely provide a pleasant physical experience. In the interactive video-sound installation "Fit,"[1] viewers use their entire bodies to interact with a video projection of an aerobics instructor. In response to a viewer's own exercising, the aerobics instructor begins exercising in synchronization with music. Within "Fit," viewers interact with their entire body without the use of an encumbered input device.

Interactivity as Content

Many interactive computer based works incorporate designs that are functionally similar to light switches: a button is pressed and a light turns on, a mouse is clicked and an image is displayed. Although this capability provides a convenience when coordinating events over time, the physical gestures expressed by viewers are not conceptually related to the responses. The experience of interactive media may be more satisfying aesthetically if a conceptual relationship exists between the human gesture and the interactive response: interactivity can be used as content. In the interactive video-sound installation "TV Guides,"[2] viewers encounter a living room environment containing a television which plays live programs overlaid with cross hairs. In response to any form of movement by the viewers, the television sound fades out and the cross hairs recede into a small circle, followed by text on the screen which requests viewers to remain still. The television imagery and sound will resume only when viewers remain motionless. The inactivity required by the viewers reflects the intended content: TV as medium of control.

Multiple Users Simultaneously

Because most people enjoy and apparently prefer group experiences -- especially during cultural events -- it seems appropriate that exhibitions of interactive art provide simultaneous interactive experiences to groups rather than single users. In the interactive sound installation "Intersection"[3] viewers encounter the sounds of cars speeding across a dark 40 x 50 foot exhibition space. In response to a viewer's presence in a lane, the cars screech, idle, accelerate and crash. This installation exists as a four or eight lane version and can accommodate more than 100 users simultaneously.

Multiple Users as Content

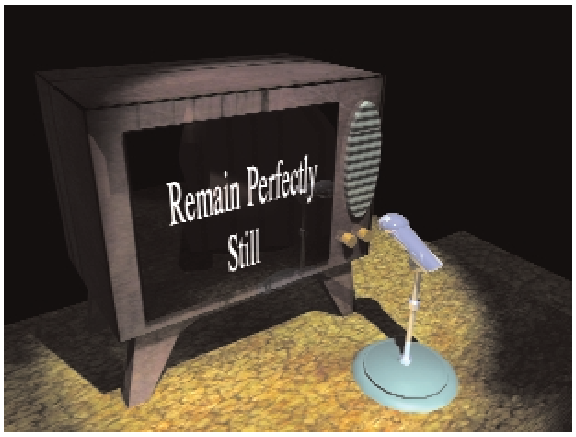
The accommodation of many viewers simultaneously by an interactive work is an efficient use of technology and provides a social environment for viewers. This form of design also has the potential to use the multiple-user design as an element within the content. The 50 x 50 foot interactive video-sound installation "Skies"[4] presents people with the experience of cooperation between themselves and cooperation with nature. As viewers walk onto video projected imagery, they discover black paths under their feet. According to the combination of paths discovered, different video sequences of nature are projected onto the wall and floor. The installation can accommodate and be controlled by an unlimited number of viewers simultaneously, although at least five must cooperate with each other to experience the work completely.

Conclusion

Although the experience of screen based interactive art may be satisfying visually and audibly, the experience of mouse clicking is not a satisfying physical experience. If interactive art is going to become an influential and cultural medium, the entire body and mind should be involved in the interactive and aesthetic experience. This can be accomplished by using human interfaces which are conceptually related to a project's content and require participation by the entire body.

References
1. aesthetic-machinery.com/fit.html
2. aesthetic-machinery.com/tvguides.html
3. aesthetic-machinery.com/intersection.html
4. aesthetic-machinery.com/skies.html

Don Ritter, "Skies" interactive video-sound installation, 50 x 50 feet.



Don Ritter, "TV Guides" detail of interactive video-sound installation, 15 x 25 feet.

NATO.0+55+3d.Modular – A New real-Time and Non-Real-Time Multimedia Authoring and DSP Environment

Contact
Netochka Nezvanova
0f0003 Maschinenkunst
Stichting STEIM
Achtergracht 19
1017 WL Amsterdam
The Netherlands
ecdysone@eusocial.com

The vectors are meant to converge through morphogenesis. They chart a fundamental shift away from the millennia, from old, worn, and static rational discourse toward the pleasure of the transient and momentary, toward the allure of a brevity of meaning, the spectral wall of impossibilities, toward the thermophilic envelope of desire that most elegantly unveils the dynamic core of a lifeform's functional cluster. Suspended in the undecided and wavering viscosity of reality, this sketch discusses a new authoring and DSP evironment distributed by 0f0003 Maschinenkunst and Ircam.

The NATO.0+55/DEAF2000 festival literature reads: "NATO.0+55 is a new software that is set to revolutionize the realtime manipulation of video and sound." Arie van Schutterhoef, artistic director of the Schreck Ensemble writes: "I think that Nato Modular is the most interesting software for Max, since Max... consider the contradiction... and offers possibilities for live manipulation of images, that can only be rivaled in the audio domain by SuperCollider."

NATO.0+55+3D. Modular is an advanced real-time and non-real-time, component driven visual authoring environment for creative construction of audio-visual networks in Ircam Max. It is a non-linear, vastly expressive and syntactic, modular infrastructure for live video and graphics, specifically 2D and 3D graphics, VR (virtual reality), DV (digital video), live video, Firewire, QuickTime, Flash, MP3, OpenGL, and real-time Internet streaming. It may be used for research, image analysis and editing, video conferencing, real-time 3D/VR viewing and modeling, real-time word processing, installations, and stand-alone multimedia applications.

NATO.0+55+3D. Modular has been deployed at Prix-Ars Electronica, DEAF, Interferences, ISEA, ICMC, the Guggenheim Museum, SIGGRAPH 2000, the US. Open, FCMM, VIPER Festival, Momentum, Expo 2000, Sonar, Not Still Art Festival, Mouse on Mars, Junge Hunde Festival, etc. It is used globally by hundreds of institutes, organizations, and individuals. NATO.0+55 workshops and seminars have been hosted by BEK, IAMAS, SALON/JAPAN, Ircam, V2/DEAF, etc.

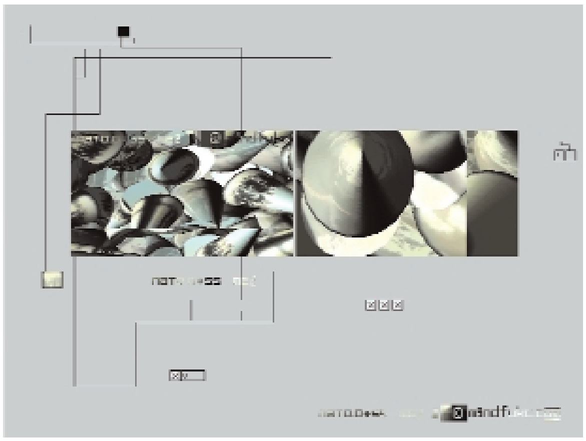


Figure 1. NATO+0+55+3d real-time 3D texture mapping.

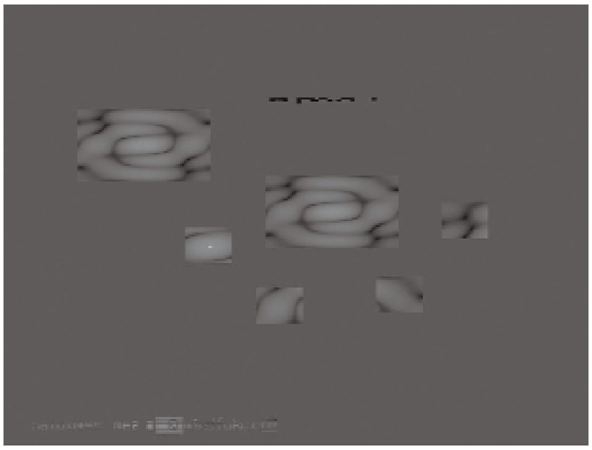


Figure 2. NATO+0+55+3d real-time 2D alphabet.

Non-Graphical Application of Hardware Accelerated Voxelization

Contact
Steffi Beckhaus
GMD - Media Communication Institute
Sankt Augustin, Germany
steffi.beckhaus@gmd.de

Jürgen Wind
GMD - Media
Communication Institute

Introduction

Voxelization is used in a number of applications, mainly in the field of volume graphics. We require it for a non-graphical application, the dynamic generation of animations for automatic presentation of models or scenes according to a user specified interest.[1] In this sketch we present the utilization of hardware accelerated voxelization to generate voxelized object data keeping a reference to the unique, single objects. With this voxelization approach we are now able to employ our animation system in highly dynamic environments and with increased resolutions.

Application

With our camera data generating approach based on potential fields, we can deal with rapidly changing user interest in objects of a scene, as well as with dynamic models and changes of the camera position introduced interactively by the user.[1] For this purpose the space is discretized into a regular 3D grid, or voxel space (see Figure 3), with object IDs assigned to each occupied voxel. According to queries to an underlying information space, user requests are mapped to objects, which in turn are set attractively in the voxel space. As a result, the camera will be attracted by these voxels and will visit all attractive objects consecutively.

In static environments the voxelization process has to be performed only once. In dynamic interactive environments the voxelization speed becomes crucial, as every dynamic object has to be mapped immediately to the voxel space when moved. Software based voxelization techniques soon become too slow to provide interactive response. With hardware based voxelization we are able to work with increased resolutions of the voxel space and with larger scenes.

Implementation

The system is implemented in AVANGO, our Performer-based framework for virtual environment development.[3] The voxelization process uses the Performer scene graph, which is at the same time used to render the scene. A custom node that handles switching between the normal rendering mode and the voxelization mode replaces the top node of each object of interest. An ID is assigned to this custom node to identify the object. In the voxelization mode this ID is mapped to an RGB color, which is then applied as an emissive material to the object's geometry. All other materials and textures for this object are disabled in this mode to ensure that the object is rendered only in the color of the ID. Objects that are not assigned a custom node are rendered in a single color. For the hardware accelerated voxelization we take advantage of the clipping capabilities of the graphics engine.[2] To voxelize the scene in a given resolution x, y, z of the voxel data set, we render the scene slice by slice in voxelization mode, thereby producing z slices of resolution x*y. The near and far clipping planes of the viewport are set to bound each slice to be as wide as the current z voxel. The hardware clips the geometry, and an image of the geometry in this slice is rendered (see Figures 1 and 2). The result is written into memory. It could now be directly rendered as a texture. For our purpose we analyze the colors in memory, obtaining for each position of the voxel space the information about whether, and by which object, it is occupied.

The voxelization time increases linearly with the resolution, but the influence of the number of slices to be rendered – the z resolution – is in our case four to five times larger than the influence of enlarging the image size by x or y (we used an Onyx2 IR2). Therefore, for efficient voxelization, the scene has to be transformed in a way that the smallest resolution of the voxel space maps to the z direction. Analyzing only sub-regions of the voxel space when only parts of the scene have changed also increases efficiency.

References
1. Beckhaus, S., Ritter, F., & Strothotte, T. (2000). Cubical path – dynamic potential fields for guided exploration in virtual environments. In *Proceedings of Pacific Graphics*, 387-395, October 2000.
2. Fang, S. & Chen, H. (2000). Hardware accelerated voxelization. In *Computers and Graphics*, 24(3): 433 - 442, June 2000.
3. Tramberend, H. (1999). Avocado: A distributed virtual reality framework. In *Proceedings of IEEE Virtual Reality*, L. Rosenblum & P. A. & Detlef Teichmann, editors, 14 - 21, March 1999.

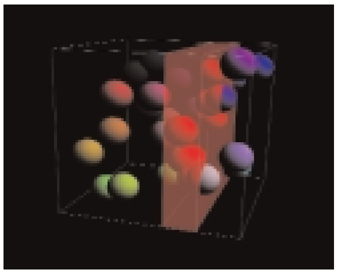


Figure 1. One slice is rendered for each z.

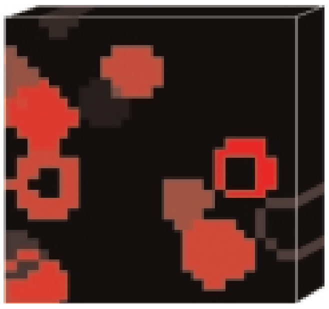


Figure 2. Slice with color-coded objects rendered in a resolution of 20 x 20 x 1.

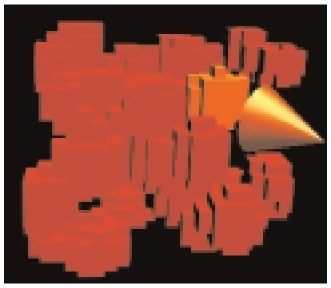


Figure 3. Bounding box in raster space.

Occlusion Culling and Summed-Area Tables

Contact
Andreas Genz
Universität Bremen
genzo@informatik.uni-bremen.de

We present a method integrating occlusion-culling[3] into the Catmull recursive-subdivision algorithm.[2] The Catmull recursive-subdivision algorithm is the ancestor of the REYES architecture, implemented in Pixar's renderer PRMan. It is capable of producing pictures from geometrically complex scenes. PRMan performs occlusion culling through sorting objects in a bucket (a rectangular pixel region) from front to back.[1]

In contrast, our method is optimized for scenes of very high depth complexity containing procedurally described geometry. The raster image is not split into buckets, instead the viewing volume is split into layers along the z-axis. Any sorting of objects is avoided. In a case study[6] the method is used to render scenes containing 100,000 trees with geometrically different features (Figure 1).

Here is an outline of the Catmull recursive-subdivision algorithm:
Given: camera, geometric objects
Requested: raster image of the scene

For every geometric object:
1. Compute camera-space axis-aligned bounding box.
2. If bounding box is entirely outside viewing volume, drop object.
3. Else: project bounding box into raster space.
4. If bounding box covers more than one pixel, subdivide object.
5. Else: rasterize object, z-buffer visibility test.

Our method

- Split scene into layers along the z-axis (Figure 2).
- Render the front most layer with the Catmull recursive subdivision algorithm.
- Build a summed-area table $T(x,y)$ of the function $f(x,y)$, where $f(x,y) = 1$, if element (x,y) is covered by some geometric object, otherwise $f(x,y) = 0$.
- Render the next layer with a modified Catmull recursive-subdivision algorithm:

4. If bounding box covers more than one pixel:
4.1. If bounding box is occluded: drop object.
4.2. Else: subdivide object.

Return to step c, until the image has totally been covered or the last layer is reached.

The occlusion test in detail (Figure 3)

Number of raster elements covered by a bounding box $b = (x_1-x_0)(y_1-y_0)$
Number of raster elements covered by some object in this bounding box $a = T(x_1,y_1) + T(x_0,y_0) - T(x_1,y_0) - T(x_0,y_1)$
Error tolerance $t \geq 0$
Drop object, if $a \geq b - t$

Further work

For which kind of scenes and visible features is this method advantageous? Does it make sense to integrate it into, or combine with, the REYES architecture?

References
1. Apodaca, A.A. & Gritz, L. (2000). Advanced renderman. Creating CGI for Motion Pictures, Morgan Kaufmann.
2. Catmull, E.E. (1974). A subdivision algorithm for computer display of curved surfaces. Ph.D. thesis, Department of Computer Science, University of Utah.
3. Cohen-Or, D. et al. (2000). Visibility, problems, techniques and applications, *SIGGRAPH 2000 Course Notes*.
4. Crow, F.C. (1984). Summed-area tables for texture mapping. *Proceedings of SIGGRAPH '84* , 207-212.
5. Foley, J. et al. (1990). *Computer Graphics, Principles and Practice*, 2nd ed. Reading, MA: Addison-Wesley.
6. Genz, A. (1998). Baum & Wald – Modellierung & bildsynthese. Dimplomarbeit, Informatik, Universität Bremen.

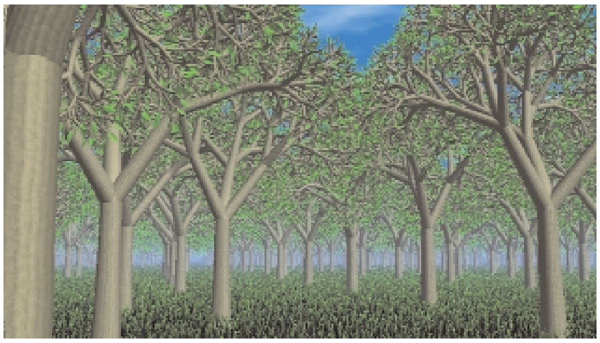


Figure 1: Trees.

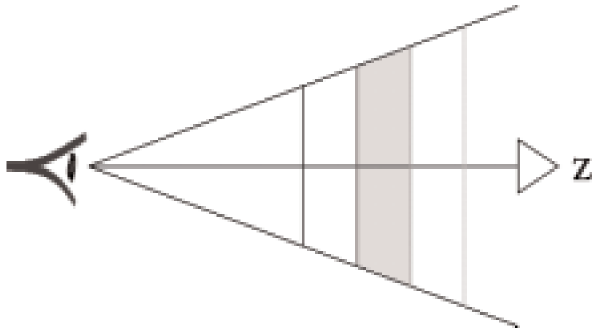


Figure 2: Split viewing volume.

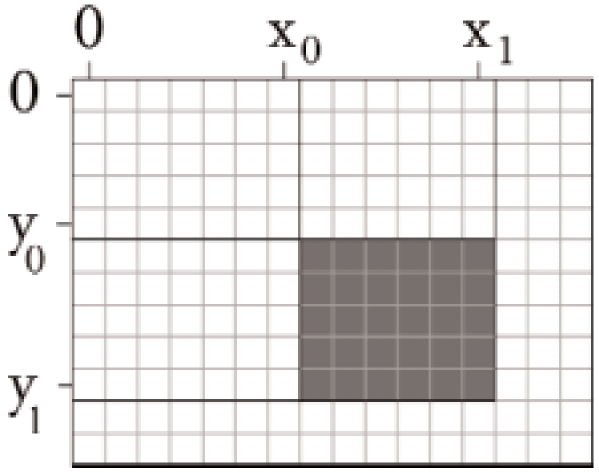


Figure 3. Bounding box in raster space.

Open Processes Create Open Products: "Interface Ecology" as a Metadisciplinary Base for CollageMachine

Contact
Andruid Kerne
Creating Media
380 Lafayette Street, Suite 202
New York, New York 10003
USA
andruid@creatingmedia.com

Introduction
Most interactive artifacts are very finite state machines. They represent determinate solutions to closed form problems. However, life is an open-ended series of situations which are not very well formed at all. The more that computational artifacts are integrated with everyday life, the more culturally significant this gap becomes. Equal value in interface ecology traverses this gap – in the process of the developer – by opening the range of methodologies that are invoked. Collage dynamically creates new meanings by combining found objects. Indeterminacy opens the set of possible states in an interactive automata. The decisions to use collage and indeterminacy in CollageMachine followed directly from an equal value, interface ecology process.

Equal Value in Interface Ecology
Interface ecology brings the perspectives of diverse disciplines to bear on the form and function of interfaces. It does this in a non-hierarchical way, according to the principle of equal value. Gertrude Stein developed equal value in consideration of the paintings of Cézanne, as a way of writing in which all words and aspects of what is represented are assigned the same rights, responsibilities, and weights. This conception of value comes from painting, where it refers to the magnitude of brightness.

Likewise, with interface ecology, no discipline dominates; none are considered subordinate. Rather, they are interdependent components, connected by flows of interaction. This "meshwork" ecosystem form supports open-ended inquiry. It enables the development of artifacts, like CollageMachine, that support open-ended processes.

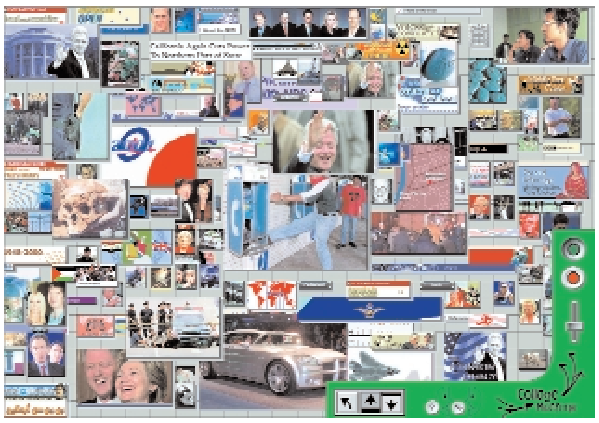
Collage
Collage is work created by combining found objects. In collage, pasted objects function semiotically to introduce new meaning to the work. As with Duchamp's readymades – such as *Fountain* – the new presentation environment of the collage creates a new context for the interpretation of its elements. Additionally, the juxtaposition of elements within a collage further alters their context, and thus their meaning. It forms semiotic relationships between them. Disciplines are a form of information age readymade; a discipline is a urinal. In interface ecology, disciplines are among the found objects which are recombined.

Indeterminacy
Indeterminacy is a means for structuring decision-making during collage-making. It has a long history as a cultural method, predating collage by many millennia. Indeterminacy refers to the utilization of chance procedures, such as random selection operations and random factors that influence the values of parameters. That is, certain creative decisions are expressed in an algorithmic form that relies partially on randomness.

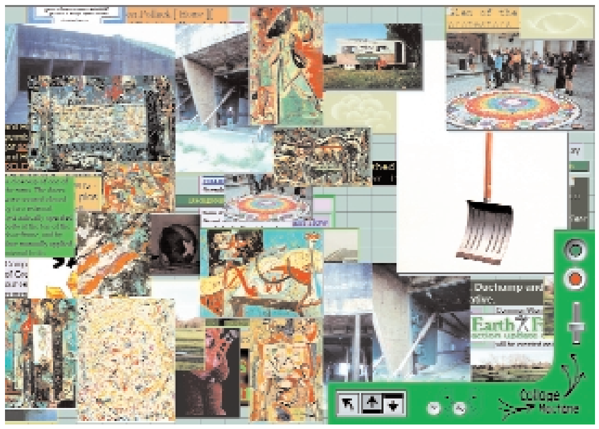
CollageMachine
CollageMachine is a creative Web visualization tool that affords browsing through streaming collage. The program deconstructs Web sites into media elements – images and chunks of text. These media elements continuously stream into a collage. A point and click, drag and drop interface enables the user to rearrange the elements. From this interaction, an agent learns about users' interests. It acts to shape the ongoing development of the collage. The agent model structures the use of indeterminacy in collage making decisions.

Open Ended Browsing
CollageMachine supports an open-ended process of Web browsing, in which the user may start only with a sense of direction. Clear advance goals are not required. On-going feedback with actual media elements enables direction to emerge. Some of the burden for clicking through hypermedia to find interesting material is relieved. This open process is an essential part of browsing. Open-ended browsing is an essential part of what makes up life. Inasmuch as interactive artifacts are integrated with everyday life, they must be conceived in terms of the full range of activities that life encompasses. Through interface ecology, the methods of collage, creative cognition, machine learning, usability, and software engineering are blended in CollageMachine with equal value.

www.mrl.nyu.edu/andruid



One state of a CollageMachine (mrl.nyu.edu/collagemachine): news collage session.



One state of a search's collage session with Greenham Common, Marcel Duchamp.

Organic Textures with Controlled Anisotropy and Directionality

Contact
Kazunori Miyata
Department of Media Art
Tokyo Institute of Polytechnics
1583 Iiyama, Atsugi-shi
Kanagawa 243-0297 Japan
+81.46.242.9634
+81.46.242.9634 fax
miyatak@acm.org

Takayuki Itoh
IBM Research
Tokyo Research Lab

Kenji Shimada
Mechanical Engineering
Carnegie Mellon University

Introduction

We propose a computational method for generating organic textures with controlled anisotropy and directionality. The method first tessellates a region into a set of pseudo Voronoi polygons (cells) using a particle model, and then generates the detailed geometry of each of the polygons using a subdivision surface method with fractal noises.

A user can create various types of realistic looking organic textures with ease by simply choosing a cell arrangement type, anisotropy, and directionality, along with the geometry control parameters.

Pseudo Voronoi Tessellation via Anisotropic Meshing

Our tessellation technique to represent organic texture consists of three steps: (1) elliptic or rectangular particle packing;[1] (2) anisotropic meshing by connecting the centers of the particles; and (3) Voronoi tessellation as a dual of the anisotropic mesh. Given preferred cell sizes and directionality within a geometric domain, the technique outputs Voronoi polygons compatible with the specified cell sizes, directionality, and anisotropy.

Organic Texture Generation

An organic texture is obtained by generating a skin texture for each of the pseudo Voronoi cells created by the method described in the previous section. Each skin texture is generated in the following three steps:

- generation of an initial skin mesh for each polygon.
- smoothing of the initial skin mesh by a subdivision surface method.
- addition of small bump features with fractal noises.

First, an initial skin mesh is generated according to the geometry of each cell. A 3D prism shape is obtained by sweeping each cell by a given skin height in the skin's normal vector direction. The initial skin mesh is then generated by deforming this prism by displacing each of the top corners randomly, skewing each of the top corners in a specified flow vector, and tapering the prism.

After initial skin meshes have been generated, they are subdivided and refined by using Loop's subdivision method.[2] Finally, small bump features are added to the skin meshes with fractal noises.

Results

The processing time depends on the number of the cells, but it is much faster than previous methods. On an Intel Pentium III 650MHz processor it takes only 10 seconds on average for the packing process and only 10 seconds on average for the skin texture generation for a texture size of 512 x 512. Figures 2 and 3 show organic textures with controlled anisotropy and directionality created by the proposed method.

References
1. Shimada, K., Liao, L., & Itoh, T. (1998). Quadrilateral meshing with directionality control through the packing of square cells. Seventh International Meshing Roundtable, 61-76, 1998.
2. Loop, C. (1994). Smooth spline surfaces over irregular meshes. In *Proceedings of SIGGRAPH '94*, 303-310, 1994.

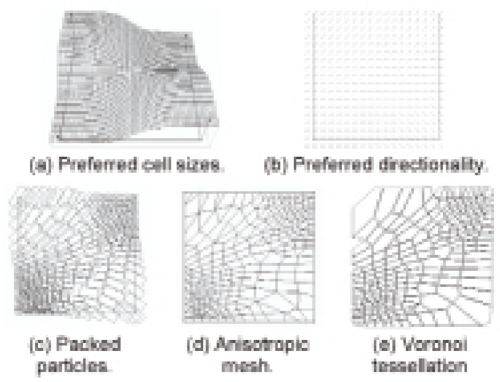


Figure 1.Three sub-steps of packing pattern generation.

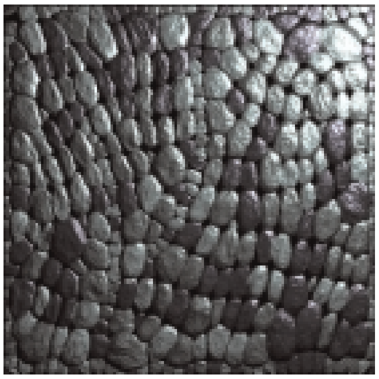


Figure 2. Example of organic textures.

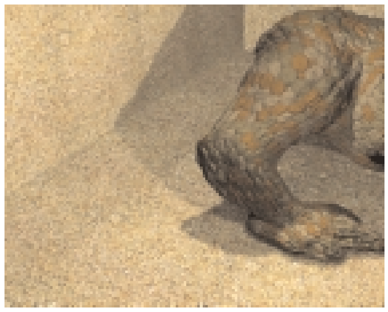


Figure 3. Textured leg.

Paint the Town Red:Recreating Stalingrad for "Enemy at the Gates"

Contact
Paul Franklin
Double Negative Ltd
paul@dneg.com

Jim Bowers
Matte Painting Supervisor
Double Negative Ltd

Åsa Svedberg
Technical Director
Double Negative Ltd

Martin Preston
Senior Programmer
Double Negative Ltd

"Enemy at the Gates" seeks to recreate the savagery and enormity of the Battle of Stalingrad. For Double Negative, one of the more challenging moments in the film was the sequence depicting a German air raid on the heart of Stalingrad. This called for a travelling aerial view of the ruined city stretching for twenty miles. All buildings would be seen in full 3D, revealing battle damage in great detail.

A previsualisation animation of camera movement and timing gained directorial approval, but it did not highlight the real star of the shot, Stalingrad itself. The model of the city's layout would be crucial to balance the need for historical accuracy with the drama of the story, but the question remained: how to quickly and simply design something as complex as the war torn city? Direct modelling was deemed too cumbersome. Instead, using Maya, a perspective grid of the city was created as seen from the animated camera. Selected frames, printed as hardcopy, became the basis for pencil sketches allowing the city plan to be rapidly approved. The sketches were then scanned back into the system, projected onto simple geometry, and tested with the animated camera as proof of concept.

Working from the layouts a team of two modellers set about creating the buildings. All structures were built intact with interior walls and floors, later to be revealed by bomb damage. Fortunately, Stalingrad was a model Soviet development constructed in the 1930s along modular lines; extensive research revealed that many forms were repeated throughout the city. Eventually a base library of approximately 50 buildings was settled on, capturing Stalingrad's essential architectural themes. Each building had a corresponding simple proxy version, which were used to recreate the sketched layout in 3D.

By this point it became impossible to avoid dealing with large amounts of geometry. Somewhere in the region of 2000 proxies had been created, and each would need to be replaced with its high-resolution version and dressed with individualised damage. Using the Maya Embedded Language (MEL) a database of the models was created allowing the team to quickly call up any city component and work on it in isolation. Additional tools enabled efficient substitution and detailing.

For the final texturing phase, the team returned to the ideas that had informed initial layout. High resolution frames of the unshaded city model were rendered from the animated camera at intervals along the timeline. These became the basis for detailed Photoshop paintings which were applied directly to the model as texture projections using custom-written RenderMan shaders. By effectively baking all lighting information into the textures, render times were minimised.

Despite the scene's complexity, a high degree of artistic control was achieved through a hand drawn and painted approach to layout. The flexible database toolset enabled a core team of only four people to manage a large number of assets. As a bonus, the efficiency of this approach allowed the team to deliver a second shot of the city seen from a different viewpoint within the original deadline.

Permanent Shutdown: Experiments in Interactivity

Contact
Justin Kent
Massachusetts Institute of Technology
Center for Advanced Visual Studies
www.justinkent.com

Artificial intelligence finds assisted suicide, online trust accesses ownership and consumption, and interaction becomes anti-interaction, as art moves towards its own inevitable permanent shutdown.

"Permanent Shutdown" is an interactive installation in which a computer has the ability to initiate its own destruction. In the initial versions of the project, this control is accomplished via serial port radio communication with an electromagnet, which suspends a cinder block over the computer. The system stands posed on the brink of self-destruction, with only a click of a button by the user to consume and complete the work.

Future versions will incorporate Web access into the piece, giving users the potential to initiate the destruction sequence from a Web terminal in any location. The site will have live video confirmation and will be password protected. Other versions will utilize other means of destruction, such as a gunshot.

In an age where the ideas of artificial intelligence and neural networks have become highly refined, and where computing power gives such programs the ability to approximate the size and complexity of the human brain, at what point does the termination of such a system become more than mere property damage? At what point will such devices be seen as more than slavish automatons, and their termination seen as murder or assisted suicide?

In experiencing any piece of art, the viewer almost always has the option of destroying it; this interaction is rarely exploited, however. It has been addressed, such as in Robert Rauschenberg's erasure of a Willem de Kooning drawing. But when a work instigates its own obliteration, does the work require that eventual destruction for it to be complete? If a work of art is created totally from commercially available stock components, does it become disposable or renewable? How does decentralized access to a work of art affect its experience?

By limiting the user's choice of interaction to one sole possibility, one single instance of that interaction, that interaction becomes exclusive and off limits to the majority of its viewers. What does it mean to have an interactive piece that cannot be interacted with? Does an interactive work require the interaction to be experienced fully? Who feels justified in completely experiencing the work themselves, thereby rendering it unusable to anyone else? Does the artist, by offering the interaction, share the responsibility with the viewer in exercising it? And how does the idea of interaction change when the cost or consequence of that interaction is greatly increased?

In an environment where art has been totally commodified and electronic trust is auctioned online, the two must inevitably intermingle. Ownership of such a work is multileveled – while only one person can possess the actual sculptural element, an unlimited number of people can be given the right to destroy (and complete) it. Furthermore, the person that initiates the piece's eventual destruction has possessed it in a unique way. This brings up issues of trust, security, and privacy, and questions the idea of art ownership.

"Permanent Shutdown" is one work in a series that explores interactivity by minimizing it through various means. Another work, "Webwheel," trivializes user interaction by using it as input into a chaotic system, thereby divorcing any specific output from a user's actions, which become noise to the system. The upcoming "Digital Gas Mask" wards off user interaction by associating with it the potential for severe physical illness.

By treating interactivity as an axis along which an artist and his or her audience must negotiate, then carefully restricting this territory, one can elucidate and differentiate interaction as art.

PlaceWorld: An Integration of Shared Virtual Worlds

Contact
Jon Cook
Advanced Interfaces Group
University of Manchester
United Kingdom
cookj@cs.man.ac.uk

Steve Pettifer
Advanced Interfaces Group
University of Manchester
United Kingdom

PlaceWorld is the result of collaboration between artists, social and computer scientists undertaken as part of the eSCAPE Project (ESPRIT 25377, 1997-2000), the aim of which was to inform the development of future large scale shared and social virtual environments. Realized as a distributed multi-user VE capable of running over wide-area networks, PlaceWorld has three goals: first, to act as a coherent metaphor for browsing and experiencing disparate electronic art installations and virtual artefacts; second, to be an evolving social environment in its own right; and third, to provide a persistent software architecture in which non-trivial virtual artefacts may be dynamically created and integrated.

Borrowing from previous multimedia installations "Place - a user's manual" and "The Virtual Museum,"[1] PlaceWorld combines a familiar planar city-like layout with unusual virtual content. Cylindrical "buildings" act as portals leading to the more extensive electronic art works, while self-contained artefacts are situated on the main landscape itself.

As well as being a container for dynamically created content, the PlaceWorld environment itself gradually evolves a structure based on its usage by its inhabitants. Artefacts and environments of interest slowly become larger, drawing attention to their popularity. Users each leave behind glowing translucent trails as they pass through the environment; these can be utilised by other inhabitants to chart the movement of companions, but also contribute automatically to the network of pathways that connect areas of the worlds. User trails following similar routes reinforce one another to become concrete "highways" allowing rapid transport from region to region, whilst trails leading to unpopular areas of the world fade over time. A user's personal trial can be used as a transport mechanism allowing multiple users to gather and interact through a text-based chat facility. Users are encouraged to personalize and contribute to the environment – from creating links to favourite areas, to using personalized objects to mark territory, and for advanced users, creating new art pieces.

The initial implementation of PlaceWorld contained two existing electronic artworks each with a number of sub-environments: The Legible City representing a 3D textural journey through three distinct virtual cityscapes; and Memory Theatre VR, in which four separate environments chart an artist's view of the development of Virtual Reality. To further populate PlaceWorld, two other applications were also embedded for the purposes of demonstration: a radiosity model of the Advanced Interfaces Group lab; and a detailed CAD model of a chemical plant.

These four works (shown in image 2) were pre-existing, stand alone single-user applications that had been developed independently on different projects. The data structures and rendering techniques required to efficiently store and display each work are radically different.

The AIG's MAVERIK system[2] was used to unite these disparate works. MAVERIK employs a callback mechanism to encapsulate the data structures and rendering techniques specific to each work. This common interface to the works allows them to coexist within a single virtual environment. The AIG's complementary Deva system[3] provided the distribution layer on top of MAVERIK that was used to implement PlaceWorld. Deva utilizes a client-server architecture with a customisable high-level language to describe schemes for synchronization. Furthermore, customisable smoothing techniques are employed to reduce the effect of network lag.

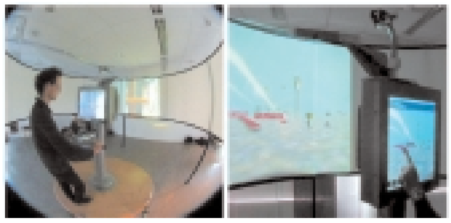
The PlaceWorld architecture, contents, and a purpose-built walk-in panoramic navigation display device with cylindrical projection screen and touch sensitive display were publicly exhibited at the ESPRIT i3NET conference in Jonkoping, Sweden, and at The Doors of Perception conference, Amsterdam, both in Q4 2000.

Further information about the architecture and future exhibitions can be found at www.placeworld.org. Contributors include: James Marsh, Simon Gibson, Andreas Kratky, Jeffrey Shaw, and Gideon May.

References
1. Dugeuet, A.M. et al. (1997). *Jeffrey Shaw - a user's manual*. Cantz Verglag, 1997.
2. Hubbold, R. et al. (1999). GNU/MAVERIK: a micro-kernel for large-scale virtual environments. In *Proceedings VRST'99*.
3. Pettifer, S. et al. (2000) DEVA3: architecture for a large scale virtual reality system. In *Proceedings VRST '00.*

1. An overview of PlaceWorld.

2. The four sub-environments of PlaceWorld.


3. PlaceWorld in use at ESPRIT i3NET conference.

Practical Image-Based Lighting for Fur and Surfaces

Contact
Deborah Carlson
Sony Pictures Imageworks
dcarlson@imageworks.com

Nickson Fong
Centropolis Effects, LLC

Introduction

This sketch describes the image-based lighting pipeline developed at Centropolis Effects. The project was inspired by recent high-dynamic range photography and image-based lighting research by Paul Debevec.[1,2] The goal of our system is to collect accurate lighting information from the set and use that data to directly light the computer graphics scene. Because the long rendering times of a physically based global illumination renderer are not yet practical in a visual effects production environment, we created a technique that could easily fit into our existing CG pipeline using photorealistic RenderMan. Our system was developed during production on "Arac Attack," which features giant furry spiders, so it was designed to handle lighting fur as well as surfaces.

Lighting from Fisheye Photographs

The basic idea we are referencing is that the incoming radiance can be captured at a given point on the set by taking fisheye photographs from that point with a range of exposures. The technique for creating high-dynamic range (HDR) radiance maps from photographs taken at multiple exposures is described in a 1997 SIGGRAPH presentation.[1] For each position on the set we wish to sample, we take two sets of 180-degree fisheye photographs to cover the entire sphere of incoming light.

Diffuse Maps

Once we have the HDR fisheye environment map, we process it to create a HDR diffuse normal map, essentially rendering a map of the surface of a small diffuse white sphere as if lit by the environment. This map is ideal for capturing the color variation of the diffuse lighting in outdoor shots. The normal map is used in a renderman diffuse light shader which returns a color given the surface normal at the point being shaded. If necessary, a variation of this light shader can be used that interpolates among several diffuse maps sampled from multiple points on the set, much like the irradiance volume technique used for synthetic scenes.[3]

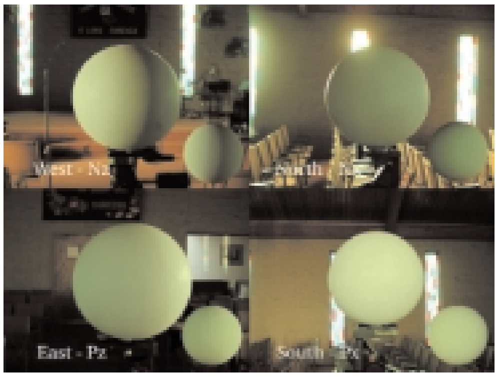
Specular Light and Shadows

In order to add specular highlights and shadows to the scene, we need to know the approximate positions, color, and intensities of the main light sources in the environment. The incoming light directions as well as the colors and intensities can be calculated directly from the HDR environment map. The distances of the lights from the sampled position can be guessed at, or they can be more accurately approximated by triangulation if the light from more than one position is sampled on the set. Extra lights that only add specular highlights and lights that only subtract diffuse shadows can be added to the scene using this information. Also, the HDR environment map itself can be used as a reflection map for shiny surfaces.

Extending the Technique to Fur

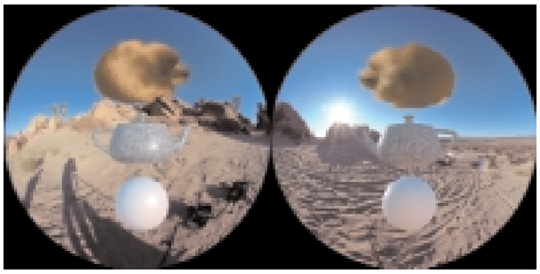
The method described above works well for standard surface shaders that use a diffuse function based on the surface normal. However, the diffuse component of a typical fur shader uses the tangent to the curve at the point being shaded, not a normal. Therefore, to extend our technique to work for fur, we create a diffuse tangent map from the HDR environment just for fur lighting, analogous to the diffuse normal map for surface lighting. The tangent maps are used in a separate light shader for fur, and our fur shaders were extended to add the diffuse illumination from the image-based fur light. The additional specular-only and shadow-only lights work as well for fur as for surfaces.

References
1. Debevec, P. & Malik, J. (1997). Recovering high dynamic range radiance maps from photographs. In *Proceeding of SIGGRAPH 97,* August 1997.
2. Debevec, P. (1998). Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and dynamic range photography. In *Proceedings of SIGGRAPH* 98.
3. Greger, G., Shirley, P., Hubbard, P., & Greenberg, D. (1998). The irradiance volume. In *IEEE Computer Graphics and Applications*, 18, (2), March/April 1998.



Comparison test with rendered diffuse ball from HDR fisheye environment maps composited over photographs of matte-painted beach ball.

Objects rendered using image-based lighting with additional specular lights and no shadows composited over background plate.


Objects rendered using image-based lighting with additional specular lights and no shadows composited over fisheye environment images.

Procedural Dynamics System for Animating the Sleigh in "How the Grinch Stole Christmas"

Contact
Markus Kurtz
Digital Domain
300 Rose Avenue
Venice, California 90291 USA
markusk@d2.com

When Ron Howard approached Digital Domain with the idea of creating a computer graphics sleigh to be used in over 50 shots of "Dr. Seuss' How the Grinch Stole Christmas," we knew that we would have to develop a system that looked realistic enough to match a set piece but was also fast and flexible enough to match the creative vision of the director in shots where the actual prop was not an option. To achieve this goal, a procedural dynamics system was written that depended solely on the sleigh's motion and its interaction with the ground surface to compute the animation of the sleigh and all of its parts.

Because our sleigh had to be able to match the physical sleigh in back to back cuts, we had to ensure that we not only captured the look of the sleigh in terms of color and lighting but also captured the behavior and dynamics of the sleigh, including its precarious payload. Traditional key frame animation was not an option due to the number of animating parts on the sleigh. Employing animators to hand animate more than 60 moving pieces on the sleigh would have been too costly and time-consuming. Additionally, standard simulation techniques are typically a very time-consuming and serial process that do not allow for much creative direction. Our sleigh needed to look realistic while still allowing for fast use of "creative license,"

Thus, a pseudo-simulation technique was developed that combined the flexibility of key frame animation with the accuracy of a simulation technique. Because of its open structure and procedural workflow, Houdini (Side Effects Software, Inc.) was chosen to build a new dynamics system, which was able to control the animation of all objects on the sleigh in a very realistic and dynamically correct way. Instead of using simulation on the entire behavior, the system was designed to be a computation of a series of individual steps based on basic physics and mathematics. Every step could be controlled and approved before moving on to the next layer of calculation. This idea developed into one single procedural flow of data, but with the option of adjusting the result after every step. Once the system was developed, workflow was pretty simple. Starting out with a given terrain and a simple key frame animation to define the rough path of the sleigh in the shot, Houdini's CHOPs (Channel Operators) were used to compute the force, acceleration, velocity, and speed of the sleigh. With that information, combined with a basic model of the underlying terrain, animation was computed so that the sleigh rode correctly on the surface, banking in curves, shifting weight, and even occasionally becoming airborne over jumps. The resulting animation could be controlled by either adjusting the original path animation or by tuning parameters in the computation.

After locking the primary animation, further procedural computtion steps calculated the animation of the sleigh's runners, the six shock absorbers, the sleigh-bag, and all the little packages, bags, and other paraphernalia that dangled off it. Every object mounted on the bag was driven not only by the sleigh's motion but also by the bag's animation based on that motion.

In the end, artists were able to set up the sleigh animation for each shot within a couple hours and the turnaround times of iterations were minimal. The procedural dynamics system returned accurate, sometimes purposely exaggerated, but always dynamically realistic animation. Because this system proved so useful on "Grinch," it has been generalized and used successfully in a number of other projects at Digital Domain. Animation of objects over any kind of terrain can be derived from this setup with only minor adjustment.

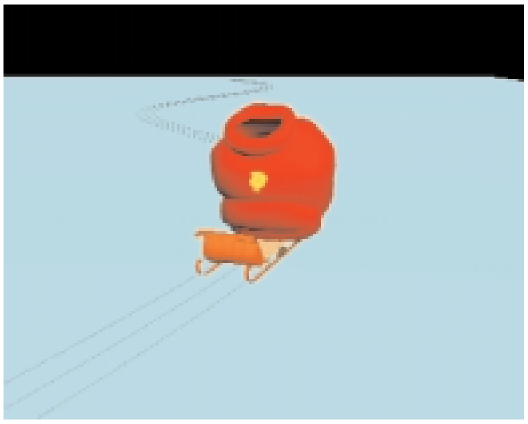


Figure 1. A simple key frame animation defines the rough path.

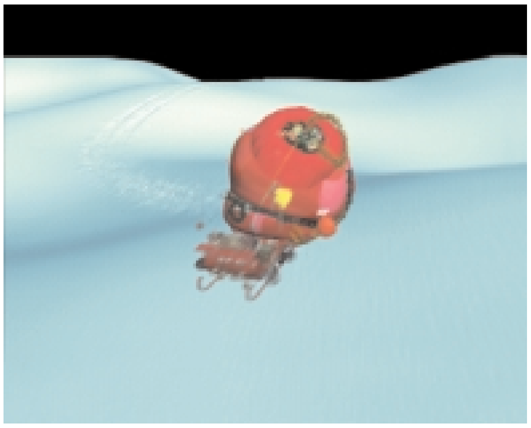


Figure 2. The procedural dynamics system computes the animation of the entire sleigh.

RainMan: Fluid Pseudodynamics with Probabilistic Control for Stylized Raindrops

Contact
Daniel L. Herman
Pixar Animation Studios
dh@digitalfish.com
www.digitalfish.com

Introduction

Simulation and production have traditionally had an uneasy relationship. Film production particularly demands that images and motion be highly directable, while simulation is generally associated with poor directability and opaque relationships between user-level control and the resulting imagery. Still there are some effects that are difficult or impossible to achieve without procedural means. This is a domain in which pseudodynamics, that is, dynamics employing a mix of theoretical and empirical bases, applies greatly.

Presented with the problem of portraying highly stylized falling rain for Pixar and Disney's film "A Bug's Life," we applied a pseudodynamic technique incorporating probabilistic control to achieve a specific look.

This control occurs via shapable Probability Distribution Functions (PDFs) that guide stochastic processes in the simulator. For fine control, the user can specify the location, timing, and precise shape of splashes produced by raindrop impacts, as well as tug on the flow to guide it in a particular direction. Alternatively, the user can allow RainMan to create falling raindrops stochastically while maintaining statistical control over the shape and timing of impacts. Finally, the overall character of flow can be broadly manipulated, providing a range of realistic or cartoon-like behavior. These multiple levels of control are a natural byproduct of the PDF representation we use.

The resulting system is fast and stable, giving intuitive and reproducible results. This stability is essential in allowing animation design iteration. These traits make RainMan well suited for use in film production.

The RainMan Particle Simulator

RainMan employs a variable mass, spatially coupled particle system. An implicit surface is rendered over the particles. Individual raindrops are composed of single large particles that may fragment on a hard impact with the ground, producing a "splat" composed of many small particles. Particles interact with each other and collision surfaces in ways that mimic the effects of surface tension and viscosity. Large streams of particles may coalesce and flow downhill, twisting, splashing, and ultimately soaking into the ground as would rainwater runoff.

Forces in the particle system include gravity, external viscous drag of particles in flight, internal viscous drag of particles in-contact moving past one another, and attractive-repulsive forces between neighboring particles (Lennard-Jones forces). Mass flows between particles of differing size in contact, equalizing surface energies between neighboring droplets, allowing particles of varying mass to interact without producing a lumpy isosurface.

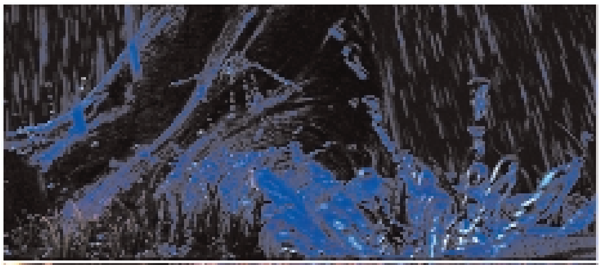
Splat Generation

The characteristic shapes formed by splats are predicted by the Navier-Stokes equations. These equations are notoriously expensive to solve in 3D and result in highly realistic motion, which we do not seek. Instead, we control splat shape indirectly by specifying how water exits from the impact. To do this, we envision a PDF that maps from the 7-dimensional particle state space (position $\mathbf{x}$, velocity $\mathbf{v}$, and mass m) to the scalar probability $P((\mathbf{x}, \mathbf{v}, m))$ that a water particle is created with a given state vector. To generate particles under this distribution P, we establish a deformation $\mathbf{W}_P$ that maps a uniform distribution to P. We may then pass a uniform random distribution of vectors through $\mathbf{W}_P$ and initialize particles with these deformed vectors.

The user-level controls in RainMan generally manipulate this $\mathbf{W}_P$. This provides a particle-independent means of controlling splat shape while providing a convenient level-of-detail mechanism: we may create more or fewer particles as appropriate to screen-projection size, yet the splat will statistically retain its shape. This *probabilistic control* further allows us to stochastically generate the $\mathbf{W}_P$'s themselves. We may populate a large area with many splats that vary in shape pseudorandomly while retaining control over broad aspects, such as to what extent they conserve energy, whether they tend to splash upwards or lay flat, or whether they are generally neat or very noisy. This higher-order control was essential on large shots containing hundreds of impacting drops.

Results

RainMan was used by a small team of technical animators in two weeks to produce the foreground rain that falls throughout most of Act 3 of "A Bug's Life." In all, nearly 10,000 RainMan raindrops fall on Ant Island by the end of the film. Simulation was a natural choice for this project, but it involves procedural and control limitations to which few directors or producers are used. The association between physical parameters and visual behavior of a simulation is often obscure, even to the simulation authors. Providing controls that corresponded to visual behavior of the simulation rather than the underlying physics enabled us to communicate more meaningfully with the directors and allowed users to become proficient with the system very quickly. This intuitive control was made possible by the use of probabilistic methods in guiding the simulation and the willingness to adopt non-physical representations. Had we employed a purely CFD technique, such high-level direction would have been impossible.



A Real-Time High Dynamic Range Light Probe

Contact
Jamie Waese
USC Institute for
Creative Technologies
13274 Fiji Way, 5th Floor
Marina del Rey, California 90292
USA
waese@ict.usc.edu

Paul Debevec
USC Institute for
Creative Technologies

In order to successfully composite computer graphics elements into live action scenes it is important that the lighting of the CG object match the lighting of the scene into which it is being composited. One technique that has been used to reproduce the incident illumination in a live-action scene is to acquire a high-dynamic-range photograph of a mirrored ball placed in the scene and then use this light-probe image as a source of illumination for image-based lighting.[1]

Previous Work

Currently, in order to create a high dynamic range image of a mirrored ball one must take an iterative series of photographs with the exposure value of each image being stopped down by a given increment from the exposure value of the one before. Later, each of the images are assembled into a single high dynamic range image using a program such as HDR Shop[3]. If an artist wished to accurately illuminate a CG object traveling through a complex lighting environment, it would be necessary to capture these iterative photographs at numerous locations (ideally at every frame) along the object's path. Clearly, this would be an ambitious task.

Technique

One solution for creating a real-time high-dynamic range light probe is to develop a system in which multiple exposures of the same image can be captured within a single video frame. We did this by modifying a five point multi-image filter (a faceted lens that is commonly used to create photographic kaleidoscope effects), and applying successively increasing values of neutral density gel to four of the five facets of the filter (3, 6, 10 and 13 stops). This modified filter effectively produces a single image that is divided into five identical regions, with the center region capturing a "direct" view and the four outer regions stopped down to their respective exposure values. This modified filter is placed on a video camera that is mounted along with a mirrored ball on a span of angle iron (see Figure 1).

Assuming the relation between the camera and the ball never changes, the light probe only needs to be calibrated once. To compensate for the angle shift introduced by parallax effects from the facets of the multi-image filter, one can compute the arctangent of the distance between facets divided by the distance between the lens and the silver ball. By determining the number of degrees each facet is offset from the center, we are able to warp each region of the filter according to the direction space of its view of the ball. In our case, each facet's view of the ball was computed to be 2.7 degrees off from center.

More accurate calibration can be done with the help of a light stage,[2] which provides a "master key" for factoring out lens distortion and imperfections in the mirrored ball. However, we found that simply computing the pixel shift and then overlapping each region of the filter was sufficient for assembling a usable image.

In order to capture high dynamic range light probe data at every frame along a path, one presses "record" on the video camera and carries the light probe along the desired path. A computer program then imports each recorded frame, isolates the five distinct images in the frame, aligns them according to predetermined calibration data, and then assembles the aligned images into a high dynamic range omnidirectional measurement of incident illumination.

Results

Figure 2 shows a raw, unprocessed image from the light probe.

Figure 3 shows several exposures of a high dynamic range image that were assembled from a single light probe frame.

Figure 4 shows a CG object, lit with captured light from the real time high dynamic range light probe.

Conclusion

This new technique will permit artists to composite CG objects into dynamic complex lighting environments, accurately reproducing high dynamic range lighting parameters for each frame. In the future, this technique would benefit from greater precision in applying the neutral density gels to the multi-image filter, a smaller camera rig, and higher resolution video cameras.

References
1. Debevec, P. (1998). Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings SIGGRAPH 98*.
2. Debevec, P., Hawkins, T., Tchou, C., Duiker, H.P., Sarokin, W., & Sagar, M. Acquiring the reflectance field of a human face. In *Proceedings SIGGRAPH 2000*.
3. Tchou, C. & Debevec, P. (2001). HDR shop. *SIGGRAPH 2001 Conference Abstracts and Applications.*

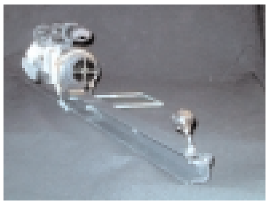


Figure 1. A real time high dynamic range light probe.

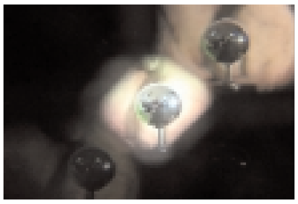


Figure 2. Five exposures of a mirrored ball in a single image.

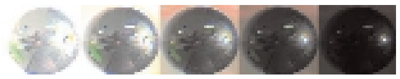


Figure 3. Five exposures of a high dynamic range image captured in a single frame.

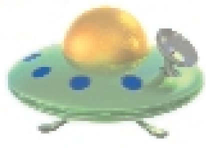


Figure 4. A CG model that is synthetically illuminated with light captured with the real time light probe.

Real-Time Rendering of Populated Urban Environments

Contact
Franco Tecchia
University College London
Gower Street
WC1E 6BT London
United Kingdom
hrysanthou/crowds/sketch/

Celine Loscos
Yiorgos Chrysanthou
University College London

Introduction

The wide use of computer graphics in games, entertainment, medical, architectural, and cultural applications, has led it to becoming a prevalent area of research. At the current stage of technology, users can interactively navigate through complex, polygon-based scenes rendered with sophisticated lighting effects and high quality antialiasing techniques. Animated characters (or agents) with whom the users can interact, are also becoming more and more common. However, the rendering of crowded scenes with thousands of different animated avatars has still not been addressed sufficiently for real-time use. In this paper we propose new methods for the rendering and culling of these highly populated scenes.

The system makes use of a subdivision of space into a 2D grid. This data structure is used to perform efficient occlusion culling on the environment and on the avatars, as well as for the agents' navigation and shadowing.

Rendering Thousands of Walking People in Real Time

Part of our work is based on the approach of Tecchia et al.[1] In order to minimize geometrical complexity, each human is represented with a single adaptive impostor. Appropriate images for the impostors are selected depending on the viewpoint position and the frame of animation. Current graphics hardware architectures often limit the maximum amount of texture memory that can be used in interactive visualizations, introducing a trade-off between memory usage and quality/variety of the rendering for image based techniques. Our current work boosts the quality of rendering using aggressive optimizations and adding important environmental effects such as shadows. We optimize the in-memory organization of the impostor images, reducing in this way the memory requirements by about three-quarters. The use of hardware-accelerated texture compression introduces an additional 1:4 compression factor. Because of this reorganization, each impostor's image can have an arbitrary size, which depends on the view direction and the frame of animation. At rendering time, the appropriate size and displacement of the impostor are computed on the fly, so as to match those of the sample image.

To enhance the crowd variety without increasing the memory usage, we use a combination of multi-pass rendering and alpha testing. In this way we can selectively address and modulate the base color of different regions on each impostor image using multiple rendering passes. Using the same impostor approach, we compute and display the shadows of the moving humans, by projecting a shadow texture of the human impostor onto the ground plane, with regard to the light direction. Shadow maps need not be regenerated, as they are simply warped versions of the existing impostors; so no additional memory is needed. Finally, a 2D environment shadow discretization is used to determine if humans are in the shade of the buildings. When a human occupies an in-shade cell, its impostor is gradually darkened. This greatly improves the visual realism of the scene.

Dynamic Culling

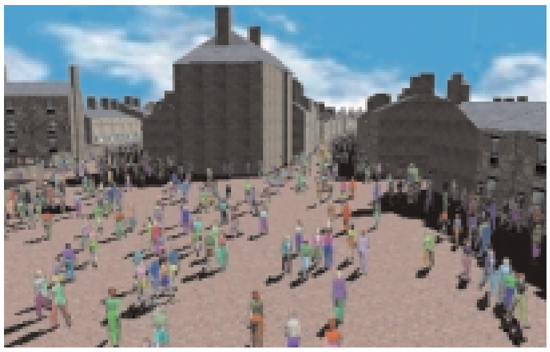
When the viewpoint gets closer to the ground, the buildings become very effective occluders. We employ a new occlusion algorithm based on binary tree merging, which uses the discretization and the properties of urban scenes to quickly cull away not only the nonvisible static geometry but also the avatars.

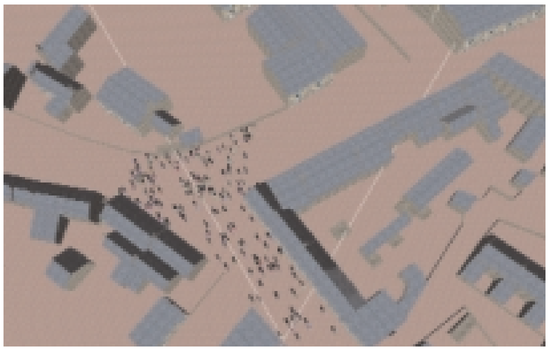
First we build a KD-tree of the scene geometry, using as partitions only planes that coincide with tile edges (given by the 2D grid). Then for each frame we build an occlusion tree from the current viewpoint and merge it with the KD-tree, marking its leaves, and indirectly the tiles, as visible or hidden. Finally, for each avatar, we check the state of the occupied cell before processing the avatar any further.

Results and Future Work

The system was tested on a PC Pentium 833Mhz with an NVIDIA GeForce GTS2 64 Mb. We populated our environment with a total of 10,000 different humans, walking around a village modeled with 41,260 polygons and performing collision avoidance against the building and between the avatars themselves. At a video resolution of 1152 x 864 pixels, the display is updated on average at 17 frames per second. The results show that the use of impostors produces interactive frame rates for crowded scenes, even on low cost hardware.
www.cs.ucl.ac.uk/staff/Y.Chrysanthou/crowds/sketch/

References
1. Tecchia, F. & Chrysanthou, Y. (2000). Real time rendering of densely populated urban environments. In *Rendering Techniques '00* (10th Eurographics Workshop on Rendering), Péroche and Rushmeier, eds., 45-56.


Crowd rendering in real time.


Occlusion culling performed on the crowd.

Real vs. Approximate Collisions: When Can We Tell the Difference?

Contact
Carol O'Sullivan
Image Synthesis Group
Trinity College
Dublin, Ireland
Carol.OSullivan@cs.tcd.ie

John Dingliana
Image Synthesis Group
Trinity College Dublin

The behaviour of objects in the physical world is described by Newtonian mechanics, using dynamic concepts such as force and mass. However, it has been reported that many people have intuitive preconceptions concerning mechanical events that, although incorrect according to Newtonian mechanics, are highly stable and widespread.[3] Profitt and Gilden showed that people use only one dimension of information when making dynamical judgements.[6] Therefore, when a dynamic event involves more than one dimension of information such as velocity and rotation (i.e. an extended body motion as opposed to a particle which has only one dimension of information), humans are less able to correctly identify anomalous physical behaviour. They also discovered that judgements about collisions were made based on heuristics and that people are influenced by kinematic data, such as velocity after impact and the way that the colliding objects ricochet.[4]

Can we exploit this imprecision of the human brain for the purpose of producing plausible real-time simulations of colliding objects? Earlier work has exploited the plausibility of certain types of approximations for simulation.[1,2] In particular, if less time is spent on processing a collision, under what circumstances will this degradation in accuracy be imperceptible? In this sketch we will present several robust factors that can significantly affect a viewer's perception of a collision and may be used to prioritise collision processing in a perceptually-adaptive system. The effect of these factors was examined in a series of psychophysical experiments.

Causality refers to the ability to detect whether one event causes another.[5] For example, a collision of a moving object with a stationary one will cause the second object to move, whereas a stationary object that starts to move by itself is perceived to be autonomous. We ran an experiment similar to Michotte's famous causality tests and found that adding a time delay between object contact and collision response reduced the perception of causality and thereby the plausibility of the collision event itself. Therefore, we can conclude that constant frame rates are imperative in any real-time collision handling system and hence interruptible collision detection is the only feasible solution for large numbers of complex objects.

Interrupting collision detection before it is complete either leads to interpenetrations, which are usually unacceptable, or more frequently to objects which bounce off each other at a distance. We found that the separation of objects when they collide provides a strong visual impression of an erroneous collision, but that this effect may be ameliorated by factors such as occlusion of the collision points, eccentricity (i.e. peripheral events) and the presence, number, and type of distractors (e.g. visually similar distractors have a stronger masking effect).

We also found that, despite reduced collision detection resolution, it is possible to produce a random collision response that is as believable as the more accurate ones, thus further masking collision anomalies. As Profitt and Gilden found, we conclude that people seem to be capable of correctly perceiving errors in collision response only when there is one salient feature (such as gap size), whereas when the simulation becomes more complex, they rely on their own naïve or common-sense judgements of dynamics, which are more often than not inaccurate. We are now conducting further experiments, using an eyetracker as shown in Figure 1, to identify the effect of these factors in more complex scenarios with large numbers of colliding entities. We will discuss the results of these experiments also.

References

1. Barzel, R., Hughes, J.F., & Wood, D.N. (1996). Plausible motion simulation for computer graphics animation. *Computer Animation and Simulation '96*, 183-197.
2. Chenney, S. & Forsythe, D.A. (2000) Sampling plausible solutions to multi-body constraint problems. In *Proceedings of SIGGRAPH 2000,* 219-228.
3. Clement, J. (1982). Students' preconceptions in introductory mechanics. *American Journal of Physics*, 50. (1), 66-71.
4. Gilden, D. & Profitt, D. (1989). Understanding collision dynamics. *Journal of Experimental Psychology: Human Perception and Performance*, 5. (2), 372-383.
5. Michotte, A. (1963). *The perception of causality*. New York: Basic Books, 1963.
6. Profitt, D. and D. Gilden. (1989). Understanding natural dynamics. *Journal of Experimental Psychology: Human Perception and Performance*, 15. (2), 384-393.

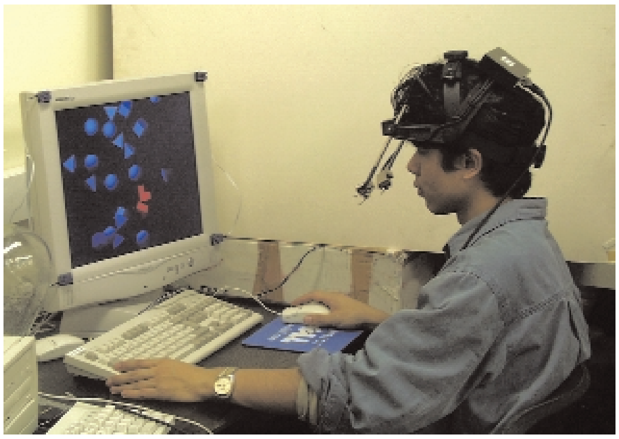


Figure 1. New Experiments using an eye-tracker.

Real-time Cloth Simulation with Sparse Particles

Contact
Masaki Oshita
Kyushu University, Japan
moshita@db.is.kyushu-u.ac.jp

Akifumi Makinouchi
Kyushu University

Physically-based dynamic simulation is a common technique for cloth animation. The clothes are modeled as a combination of mass-distributed particles and elastic forces that work between the particles. Typically between a few thousand and a hundred thousand particles are required to simulate cloth movement. Although many techniques have been developed for fast and robust simulation,[1] dynamic simulation requires too much computational time to animate clothes in real-time.

In this sketch, a novel technique for real-time cloth simulation is presented. The method combines dynamic simulation and geometric techniques. Only a small number of particles (a few hundred) are controlled using dynamic simulation to simulate global cloth behaviors such as waving and bending. The cloth surface is then smoothed based on elastic forces applied to each particle and the distance between each pair of adjacent particles. Using this geometric smoothing, local cloth behaviors such as twists and wrinkles are efficiently simulated (Figure 1).

Proposed Method

In the proposed method, PN triangles[2] is used to smooth a cloth mesh. This is a smoothing technique that substitutes a three-sided cubic Bézier patch for each triangular face of a mesh. It facilitates control of the smoothing mesh using the normal of the vertices, while other smoothing techniques such as subdivision (e.g. Catmull-Clark) and parametric (e.g. NURBS) surfaces generate a smoothed mesh from the vertices of a simple mesh. In addition, PN triangles ensures that the generated surface matches the original mesh on each vertex.

The cloth surface is smoothed by controlling particle normals and edge length using the PN triangles. Figure 2 illustrates the smoothing method. Figure 2 (a) is the original mesh and (b) is the smoothed surface using PN triangles without normal control. The normal of each particle is computed from the elastic forces that are applied to the particle from the adjacent faces. The elastic forces work so as to make the tangent of the cloth surface parallel with the elastic forces. Therefore we compute the normal direction by balancing the vectors that are orthogonal to the elastic forces – Figure 2 (c). In addition, when the length of an edge is shorter than its original length, the midpoint of the curve is moved so as to maintain the original edge length – Figure 2 (d). As a result, the smoothed surface reflects the elastic forces and maintains surface dimensions.

Implementation and Results

A particle-based dynamic simulation using the proposed method has been implemented. The proposed method is very simple and is easy to implement and integrate with an existing particle-based system. In terms of dynamic simulation, existing techniques[1] could be used. However, the proposed method efficiently smoothes the cloth surface in response to the distance between adjacent particles by setting low stiffness of the cloth, while standard systems maintain the shape of the cloth by setting high stiffness.

Figure 3 shows a snapshot of the resulting animation. Forty particles of the skirt were controlled using dynamic simulation – Figure 3 (a), and 4,096 triangles used for the smoothed surface – Figure 3 (b). The computational time was approximately 30 milliseconds for each frame of the 30 Hz simulation. As a result, the skirt was animated in real-time. The proposed method has potential for use in computer games, virtual studios, and virtual fashion shows.

References
1. Baraff, D. & Witkin, A. (1998). Large steps in cloth simulation. *In Proceedings of SIGGRAPH 98*, 43-54.
2. Vlachos, A., Peters, J., Boyd, C., & Mitchell, J. (2001). Curved PN triangles. *Proceedings of the 2001 ACM Symposium on Interactive 3D Graphics*.

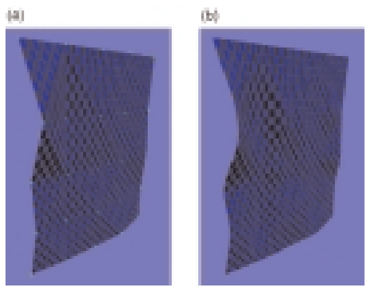


Figure 1. An example of the proposed approach: (a) a small number of particles (25 particles in this case) are controlled using dynamic simulation; (b) the cloth surface is then smoothed using geometrical techniques (2,048 triangles were rendered).

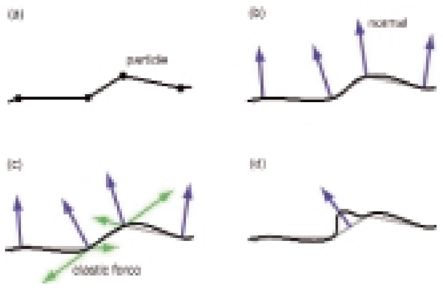


Figure 2. Smoothing method: (a) a section of an original mesh -- the middle edge is shortened and the elastic forces are working so as to part the two vertices; (b) the smoothed surface by PN triangles; (c) with normal control; and (d) edge length control.

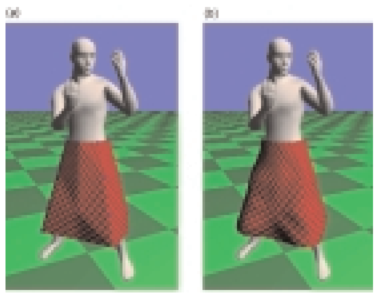


Figure 3. A snapshot from an animation of a skirt: (a) the controlled mesh (40 particles); and (b) the smoothed surface (4,096 triangles).

Real-Time Computer Graphics for On-Set Visualization: "A.I." and "The Mummy Returns"

Seth Rosenthal
Motion Capture Supervisor

Doug Griffin
Mike Sanders
Motion Capture Engineer
Industrial Light + Magic

One of the basic challenges of directing visual effects shots is imagining the result. The skill of directors can be undermined when they cannot see the elements of a shot on which they typically base their decisions. As a result, effects shots often require extensive iterations during post-production to address aesthetic issues that directors can otherwise handle instinctively and immediately when shooting non-effects shots. For "A.I." and "The Mummy Returns," we used real-time computer graphics and 3D tracking techniques to give the directors the ability to see some of the effects they were directing.

For "A.I.", we assembled a virtual set system on a large blue-screen stage to give Steven Spielberg the ability to compose shots of actors walking and dancing through a futuristic city. For "The Mummy Returns", we used real-time motion capture combined with camera match-moves to visualize Arnold Vosloo's computerized counterpart performs live over clean background plates. Although these projects required different tools, they were motivated by the same basic desire to move key artistic decisions about effects shots back into the realm of live production.

Artist Application

The Rouge City sequence in "A.I" required actors to perform in a large blue screen stage while the camera framed their action against a nonexistent skyline. The storyboards called for complex compositions of the actors and the synthetic city. For example, in one shot, after a long walking conversation, two characters step up onto a fountain and pause while the camera swings around to frame them alongside action taking place high up on a building façade. They then step off the fountain and stroll off down the boulevard while the camera pulls back to frame them against another dramatic structure. In order to visualize these shots, we assembled a virtual-set system consisting of a real-time camera tracker capable of working over an area approximately 60 feet by 120 feet with a 35 foot ceiling, a real-time blue-screen keyer/compositor, and a computer and software capable of rendering the computer model of our city directly to a video signal. In addition, we integrated this system with the film camera package in use by the "A.I." production, and devised a method for calibrating the virtual camera parameters to match the film lenses.

Our visualization system was used on the "A.I." set for three weeks of principal photography. The remarkable power of this tool was immediately apparent on set. As soon as the system generated a live image, everyone on set who usually relies on the image from the camera to guide his or her work was at home with the system; in spite of its technical complexity, it required no explanation. The system allowed the director to compose elaborate camera moves, make subtle adjustments to the timing and framing of the shots, and to play spontaneously off of the changing composition of the actors and the virtual environments exactly as he would do on a traditional live-action shoot.

Real-time motion capture for "The Mummy Returns."

As in "The Mummy," a number of shots in this sequel required that the title character, played menacingly by Arnold Vosloo, be represented by a computer generated character. Our work on the first project gave us an appreciation of some of the subtle challenges of directing dramatic performances for motion capture. In particular, we found that with conventional offline motion capture it is difficult to direct dramatic close-up shots in which small variations in posture or attitude can completely alter the composition and feel of a performance. "The Mummy Returns" provided us with the opportunity to address these challenges with a new generation of technology. We used a new real-time optical motion system, the Vicon 8, to track Arnold Vosloo's performance. We composited a low-resolution CG character driven by the real-time data over clean background plates that had been shot earlier in the production. By rendering the CG character through a match-moved camera, we maintained a perspective correct match between the live character and the pre-existing plate.

As in the case of the visualization system for "A.I.," the usefulness of this technique was immediately apparent. The ability to see a live composite of the shot made it possible to direct Arnold Vosloo's performances according to fine decisions about timing and composition and dramatic quality. In post-production, we found that these performances, for the most part, fit neatly into the final shots with little need for adjustments to the structure of the performance, thus allowing animators to concentrate on facial and hand animation.

Conclusion

We have made a concerted effort to apply real-time tracking, computer graphics, and video technology to the feature film production. Although these projects used different tools to visualize different aspects of the desired shots – the synthetic city for "A.I." and the CG performer for "The Mummy Returns" – they both illustrated the same basic advantages of real-time visualization for feature-film visual effects production: the director gains artistic control over the shot and the post-production artists get an unambiguous representation of the director's vision as a starting point for their work.

Real-Time Stroke Textures

Contact
Bert Freudenberg
Institut für Simulation
und Graphik
Otto-von-Guericke-Universität
Magdeburg
Universitätsplatz 2, 39106
Magdeburg, Germany
bert@isg.cs.uni-magdeburg.de

Shading in line drawings is expressed by varying the stroke width or density of strokes covering an object's surface. In their pioneering work, Winkenbach and Salesin introduced the concept of "prioritized stroke textures" to the emerging field of non-photorealistic rendering.[1] Despite the advances in processing power in the years since then, the sheer number of lines to draw prevents this method from running in real time. A real-time approach for hatching was presented by Lake et al,[2] which chooses from a set of textures based on the brightness at vertices, subdividing polygons if necessary. However, the method is very CPU-heavy and requires many polygons.

Our new technique uses per-pixel shading graphics hardware to implement non-photorealistic shading. The texture-combining facilities accessible via OpenGL on NVIDIA GeForce and ATI Radeon cards provide the flexibility necessary to vary the line width or number of strokes per area.

To indicate shading by variable-width hatching, a 3D halftone pattern is created as texture T and compared at every pixel with the target intensity I, creating black or white pixels.[3] Halftoning, however, yields unsatisfying results for interactive applications because of the computer screen's limited resolution. To smooth the harsh transition from white to black we instead take the inverted sum of T and I and scale it by some constant $c > 1$. After clamping the result to the range (0…1)we get a mostly black and white output while still preserving a few gray levels (see Figure 1).

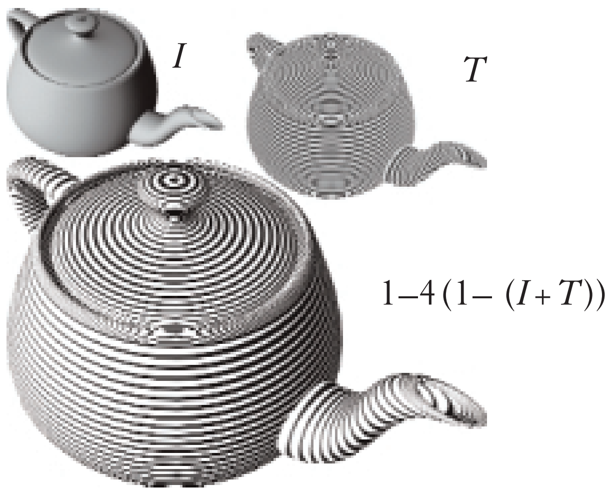


Figure 1. Lighting-dependend stroke width.

We follow a similar approach with stroke textures. Strokes are drawn on different layers. To facilitate one-pass rendering, all stroke layers are composited into the texture T in a pre-processing step, using a different gray level for each layer (Figure 2). At runtime, all layers needed to visually approximate the intensity I are selected for drawing, employing the same combiner operations as introduced above (see Figure 3).

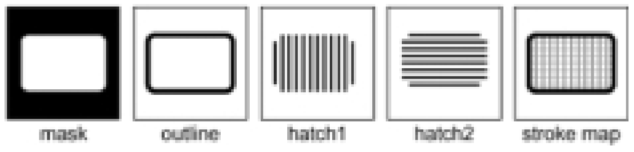


Figure 2. Stroke map construction.

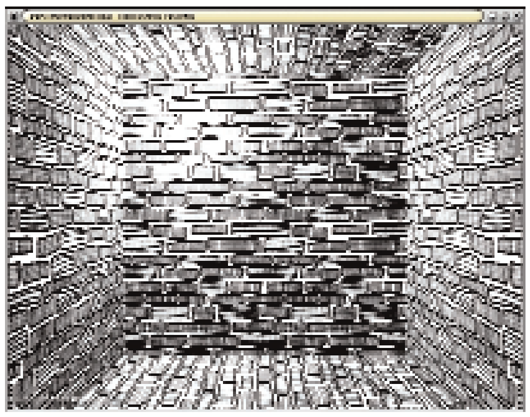


Figure 3. Real-time stroke textures.

There are many ways to extend the basic idea of stroke map shading. Lightmaps can be used to vary the amount of detail drawn on a surface. With bump maps, individual strokes can be made sensitive to lighting. Shadows can be rendered in additional passes. We are working on integrating all these into our interactive line drawings.

Our method does not require any additional CPU work at runtime. The configurability or even programmability of recent graphics hardware is a very powerful device to achieve non-standard looks. We are looking forward to seeing more visually rich interactive rendering styles emerge in the future.

References
1. Winkenbach, G. & Salesin, D.H. (1994). Computer-generated pen-and-ink illustration. In *Proceedings. SIGGRAPH 94*, 91-100.
2. Lake, A., Marshall, C., Harris, M., & Blackstein, M. (2000). Stylized rendering techniques for scalable real-time 3D animation. In *Proceedings NPAR 2000*, 13-20.
3. Haeberli, P. & Segal, M. (1993). Texture mapping as a fundamental drawing primitive. In *Fourth Eurographics Workshop on Rendering*, 259-266.

Rendering by Manifold Hopping

Contact
Heung-Yeung Shum
Microsoft Research, China
hshum@microsoft.com

Lifeng Wang
Microsoft Research, China

Jin-Xiang Chai
Carnegie Mellon University

Introduction

IBR in general simulates a continuous range of camera viewpoints from a discrete set of input images. Much of the work on IBR has depended on view interpolation, which is however a difficult task because feature correspondence needs to be known. Many IBR techniques are therefore proposed to make use of a large collection of images to avoid the correspondence. The minimum number of images needed was studied recently.[1] In this sketch, we propose a novel IBR rendering technique called manifold hopping, which breaks the Nyquist limit for perfect light field reconstruction[1] by taking human perception into account. Specifically, our technique provides users with perceptually continuous navigation by using a small number of strategically sampled manifold mosaics or multi-perspective panoramas.

Basic Ideas

Manifold hopping has two modes of navigation: moving continuously along any manifold, and discretely across manifolds. Figures (a) and (b) show these two modes of navigation using concentric mosaics.[3] A user can move continuously along any of the circles as shown in Figure (a), while the arrows in (b) indicate that the user can only hop to the viewpoints on the circle, but not stop anywhere in between. To render a novel view, we locally warp[2] the concentric mosaic where the viewpoint is located. Because we do not know the accurate scene depth and use the constant depth assumption for warping, the rendering error is inevitable. However, local warping does not introduce any structural errors such as double images, which are common for view interpolation without accurate correspondence. Moreover, the geometric distortion introduced by warping can be tolerated by human visual perception when the field of view of the rendered image is small (e.g., below 40°).

Moving discretely along the radial direction can be made perceptually smooth if the interval can be set reasonably small. A key observation is that there exists a critical hopping interval for users to perceive a smooth navigation. In other words, manifold hopping provides users with perceived continuous camera movement, without continuously rendering viewpoints using infinitesimal steps. As a result, manifold hopping significantly reduces the input data size without accurate depth information or correspondence.

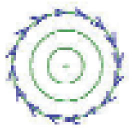
Using the signed Hough ray space, we carried out the detailed analysis of manifold hopping to address two important questions: First, what is the largest field of view that still produces acceptable local warping error? Second, how large can the hopping interval be so that continuous motion can be perceived?

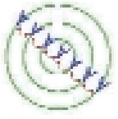
Results and Future Work

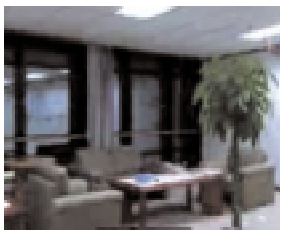
Using a Sony DV camera, a total of 5726 frames of resolution 720 x 576 are captured for a full circle. Instead of using 720 rebinned concentric mosaics of size 5726 x 576, we select only a small subset of resampled concentric mosaics. To further reduce the amount of data used in manifold hopping, we can resize the original concentric mosaics. As shown in Figures (c) and (d), two images with low resolution 184 x 144 are rendered from 11 resized smaller concentric mosaics which are compressed with a predictive coding compression algorithm. All the 11 concentric mosaics are compressed to only 88k with a compression ratio 78. An important future direction would be to conduct a more comprehensive study on the psychophysics of visualization for our technique. We also plan to extend manifold hopping to other kinds of manifold mosaics than concentric mosaics.

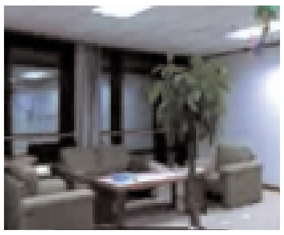
References

1. Chai, J.-X., Tong, X., Chan, S.-C. & Shum, H.-Y. (2000). Plenotpic sampling. In *Proceedings SIGGRAPH 2000*, 307-318.
2. Mark, W., McMillan, L., & Bishop, G. (1997). Post-rendering 3D warping. In *Proceedings Symposium I3D*, 7-16.
3. Shum, H.-Y. & He, L.-W. (1999). Rendering with concentric mosaics. In *Proceedings SIGGRAPH 99*, 299-306.


(a)


(b)


(c)


(d)

Rendering "Pepe" with Global Illumination

Contact
Marcos Fajardo
Institute for Creative Technologies
University of Southern California
13274 Fiji Way
Marina del Rey, California 90292
USA
fajardo@ict.usc.edu

This animation sketch focuses on lighting and rendering aspects of the animation short by Daniel Martinez Lara, featuring his Pepe digital puppet and rendered using an early version of our custom Monte Carlo ray tracing system, code-named Arnold. The short is 100-percent CG and achieves a realistic hand-held-video-camera look through the use of believable camera motion and zoom, scanned textures, exquisite character animation, and high-quality soft shadows and inter-reflections in a daylight setting.

With a traditional scanline-based renderer, the artist would have to painfully and skillfully position fill lights to mimic the soft effects of sky and bounce light. Our Monte Carlo-based renderer offers a much simpler approach. By specifying a direction and intensity for the sun (a standard CG directional light), and the color and intensity of the sky (uniform background radiance), we let the renderer automatically compute the subtle bounce light and soft shadow effects. Since the characters walk, jump, and interact with the environment, the illumination cannot be stored in advance and is in fact calculated for every pixel, for every frame. Thus, we get extraordinarily high-quality dynamic lighting effects with subpixel accuracy, at the expense of slower rendering times. We are in effect trading manual scene set-up time, which requires valuable human resources and skills, for CPU rendering time, which requires comparatively cheaper hardware resources, while achieving a unique, natural look. In particular, moving soft shadows and contact shadows are extremely accurate and contribute greatly to the overall sense of realism.

It is important for the artist to have quick feedback from the renderer, where it responds immediately to arbitrary changes in geometry, materials, or lighting. We achieved acceptable feedback by rendering in two stages. In the first stage, the renderer uses a single sample per pixel, providing a useful rough (noisy) version of the image in a small fraction of the total render time. In a second stage, each pixel is super-sampled and the image is progressively "cleaned." There is no meshing or radiosity pre-processing. The user can also interactively select a small region in the image and quickly render those pixels only. This immediate access to any pixel in the image proved invaluable in production of the animation.

Another important factor was the ability to render very dense triangle meshes with little increase in rendering time, thus letting the artist use very finely subdivided characters, on the order of 100,000 triangles each. Finite-element global illumination renderers would have a very hard time in this character animation environment. Any gathering Monte Carlo renderer will have some difficulty with concentrated sources of illumination due to occlusion, like a room with a small window to the sky. In order to further reduce rendering times, a favorable scene setup was created, where we removed the roof of the room and also the back walls. This helped sky light fill in the scene. Final render times were around 10 minutes per frame on a dual Pentium III 500MHz machine with 256 MB of RAM, at a resolution of 320 x 240 pixels. The scene contains over 600,000 triangles in total. Around 3,000 frames were distributed across a network of machines.

Key Credits

Animation
Daniel Martinez Lara, Carlos Fernandez Puertolas

Additional animation
Kike Oliva

Custom programming
Gonzalo Rueda

Core rendering software
Marcos Fajardo

See also: www.pepeland.com

Rendering Semi-solid Objects with Particles

Contact
Scott Singer
Visual Effects Animator
PDI/DreamWorks

PDI/DreamWorks' computer graphics animated film, "Shrek," required that a technique be developed to create realistic tubes that appeared to be made of mud. This sketch describes the approach used to render the deforming semi-solid mud objects without making them look like stretchy rubber sacks.

The shot called for several mud tubes to be pushed out of a hollow log and appear to be very gross; the effect was ultimately referred to as "mud poop." The direction called for each mud tube to deform and stretch for comedic effect. Early tests using traditional texture mapping always made the tubes look like stretchy, painted rubber sacks, and the approach was abandoned early on. What was clear was that in order to make the effect work, we would need a way of rendering surface detail which would not stretch with surface area changes. Instead, the surface detail needed to appear to separate, combine, and shift in response to changes in the underlying geometry.

We needed to find the right kind of renderable primitive and a method for applying it to the surface of the poop geometry. We also had to figure out how to handle surface detail that would necessarily be revealed as the surface details separated. After experimenting with instanced geometry, we settled on particles. They provided us the right level of directability with the right level of spatial ambiguity. Particles brought with them their own set of difficulties, chief among them being motion blur and specularity.

Separation was handled by having a sub-strata of particles below the rendered surface which would be revealed by the surface particles as they were moved apart by the stretching parent geometry.

RitSens - Computer Aided System for Studying Rhythmic Abilities in Bimanual Coordination

Contact
Artemis Moroni
Instituto Nacional de Tecnologia da Informação
Rod. D. Pedro I, km 143,6
13089-500, Campinas, Brazil
Artemis.Moroni@iti.br

Alaide P. Mammana
Instituto Nacional de Tecnologia da Informação

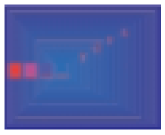
Luis Teixeira
Escola de Educação Física e Esportes
Universidade de São Paulo

Introduction

The integration of several sources of sensorial information (stimuli) and the production of motor responses involving coordination is a necessary function in several daily situations. Nowadays, a great effort is devoted to understanding which mechanisms are responsible for driving the movements and which processes occur in their learning. In particular, the study of rhythmic ability in bimanual coordination is becoming the object of intense research, in order to determine the space and time patterns that characterize the movement, and to understand the basic principles and mechanisms that are guiding the system to present specific coordination patterns. But, these investigations require the development of special methods and techniques. Computer systems are very attractive for this usage due to their capabilities for the processing, storage, and exhibition of information, if they can be used with special devices that detect the response of both hands in real time. To be largely accessible for institutions and individuals, these peripherals must be simple, dispensing previous training; versatile, for using with different computers and programs; comfortable, safe, and cheap.

Since these peripherals are not available, we developed a transparent and resistive tablet[1] that can operate with two pointers, one for each hand. The tablet can be fixed directly on the computer video screen or can be operated on a common table. It can also be used with one pointer, as a precision digital table, a light-pen, a mouse, or a special keyboard. In short, the tablet architecture was conceived so that it may be connected to any equipment with an RS232 serial interface, which makes the software development very easy. Many applications were demonstrated as an alternative and augmentative communication with patients with language limitations, and in literacy and pre-literacy of children with brain paralysis.

Besides the device driver software, the development included the software for the specific application in the rhythmic ability investigation in bimanual coordination. In this sketch, we present RitSens, an improved application speciallydesigned for investigation of rhythmic abilities in bimanual coordination.



RitSens

RitSens produces static and dynamic, visual and aural stimuli, allowing the determination of the spatial and temporal right and left hand responses. The screen must be touched by the individual in response to these visual or aural stimuli.

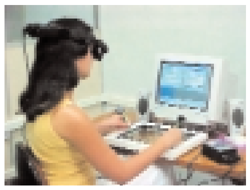


Figure 1. Tablet with two pointers being used with a head mounted display.

Four signals, two aural and two visual, may be independently produced, with programmed duration and frequency. These stimuli may also be produced in a synchronized way, with relations among them, i.e., two for one, etc., between an aural and a visual signal. The visual stimuli consist of targets with shape, size, distance, orientation, and colors that can be defined at each test and presented in the video screen. The use of a head mounted display to present the visual and aural stimuli has the advantage of producing a complete immersion in the sensorial environment.

The preliminary version worked with the common sound library available in the computer,[2] but in this last version a MIDI driver was added so that any MIDI file can be read and played. The system allows the determination of the initial and final time for the visual and aural stimuli as well as the user response time. For each response, the time and the x and y coordinates of the touch are registered, both for the right and the left hand. In this way, the temporal and spatial patterns produced by the user may be analyzed.

In order to study the rhythmic learning and memorizing, the stimuli may be presented and removed, but continue to serve as a temporal reference to the user response pattern. The system operates with a set of commands that allows a great flexibility in the creation of the essay's storyboard without previous programming language. The operation with a MIDI library opens the possibility that the transparent tablet be used as a musical keyboard, applied on the video screen. More than that, it can be used with Vox Populi,[3] which is a composition system that uses line drawings as fitness functions in an evolutionary computation approach for algorithmic composition.

Acknowledgements

The results obtained with the transparent tablet were due to the great engagement and dedication of M. A. Schreiner and M. R. R. M. Gobbo in its development. The programming and presentation of the visual and aural stimuli were founded in the extensive research which Professor A. M. Pellegrini developed in motor coordination studies, with C. Y. Hiraga. RitSens' functions were programmed by C. K. Moritani. M. A. Schreiner, M. R. R. M. Gobbo and C. K. Moritani are members of Laboratório de Mostradores de Informação, Instituto Nacional de Tecnologia da Informação. Prof. A. M. Pellegrini and C. Y. Hiraga are members of LABORDAM, Departamento de Educação Física, Instituto de Biociências, Universidade Estadual Paulista.

References
1. Schreiner, M.A., Sánchez, C.A., Gobbo, M.R.R.M. & Mammana, A.P. (2000). *with Tablete inteligente a filme fino transparente-Iberdiscap 2000*, Madrid, 353.
2. Mammana, A.P., Schreiner, M.A., Ferreira, D.V,. Bohorqueze, J.C., & Garcia, A. (1999). In *Mobilidade e Comunicação - Desafios à Tecnologia e à Inclusão Social,* A.A.F. Quevedo, J.R. Oliveira, M.T.E. Mantoan, eds. Unicamp, 1999, 255.
3. Teixeira, L.A., Pellegrini, A.N, Hiraga, C.Y., Schreiner, M.A., Sánchez, C.A., Gobbo, M.R.R.M. & Mammana, A.P. *Actas del 3er de Comunicación Alternativa y aumentativa y 1er de Tecnologias de Apoyo para la Discapacidad - Iberdiscap 2000*, Madrid, October, 2000, 77.
4. Moroni, A., Manzolli, J., Von Zuben, F., and Gudwin, R. (2000). Vox Populi. *Leonardo Music Journal*, Vol. 10, 49-54.

Shadermaps: A Method for Accelerating Procedural Shading

Thouis R. Jones
Ronald N. Perry
Michael Callahan
MERL

Introduction

Procedural shading[1] has proven to be an indispensable tool in computer generated animations. Procedural shading's primary benefit is its flexibility, since shaders can be arbitrarily complex in their actions. This flexibility has its price: it can take hours or days to render a single frame for a studio animation.

Shadermaps are a new method for accelerating procedural shading, driven by two observations. First, objects tend to have intrinsic appearances (e.g., the grain in a wooden figurine) that are static from frame to frame. Second, the intrinsic appearance of an object is usually responsible for most of its complexity and rendering cost. Shadermaps accelerate shaders by exploiting this static complexity. Shaders are separated into static and dynamic phases, and the former's output is stored and reused from frame to frame.

Algorithm

We define the static phase of a shader as the part depending only on static appearance parameters, and the dynamic phase as everything else. Since the static phase's computations depend only on static parameters, one can reuse those computations between frames. In our algorithm, the static phase's output is generated at multiple resolutions and stored in a shadermap, a mipmap of intermediate computations. These intermediate computations are a "snapshot" of the interface between the static and dynamic phases at points on the surface. For each frame, it is possible to reconstruct the intermediate computation on the surface from the shadermap. This reconstruction warps the shadermap to screen space, analogous to texture mapping, using a high quality anisotropic filter.[2] The dynamic phase uses this reconstruction to complete the shader calculation and produce the final color. Since the warp is usually much less expensive than an evaluation of the static phase, the reuse of static computations results in a significant acceleration of the shading calculation.

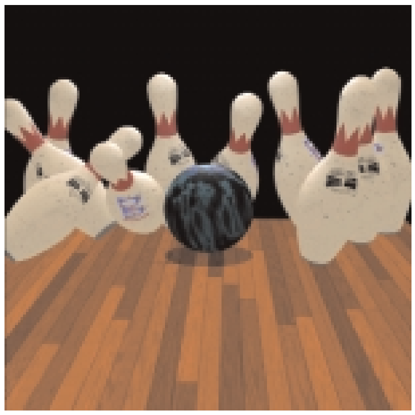
A benefit of procedural shading is resolution independence; a shader can have detail at a wide range of scales. Generating shadermap data at all scales would be wasteful, since in any single frame the dynamic phase requires only part of the shadermap. To avoid unnecessary computation, the shadermap is generated on demand. If the anisotropic filter operation requires data that has not been created, the static phase is invoked to produce the needed data, and the result stored in the shadermap. To reduce overhead, shadermaps are stored in a sparse mipmap made up of tiles, and an entire tile is generated when that tile is first accessed. Since a shadermap is stored at several resolutions, it is similar to a mipmap. Unlike mipmaps, however, the data is generated by a procedure, rather than by filtering a high-resolution image.

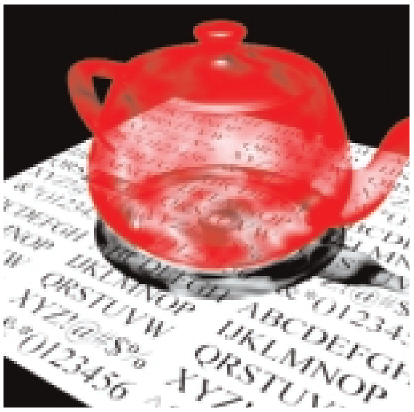
Results

One of the factors controlling the acceleration from shadermaps is reuse of shadermap data, which we measure as the ratio of the number dynamic phase evaluations to the number of static phase evaluations. A ratio of 10:1 means that for every ten evaluations of the dynamic phase, the static phase is evaluated only once. This ratio does not depend on the particular shader applied to a surface, but rather on the geometry and parameterization of the surface and the directions from which it is rendered. We have tested several animations and found good ratios, usually around 15:1, or about 95% reuse.

See www.merl.com/reports/TR2000-25

References
1. Apodaca, Anthony A., & Gritz, L. (2000). *Advanced RenderMan*. Morgan Kaufmann, 2000.
2. McCormack, J., Perry, R., Farkas, K., & Jouppi, N. Feline: fast elliptical lines for anisotropic texture mapping. In *Proceedings of SIGGRAPH 99*, 243-250.





Two images generated with shadermaps.

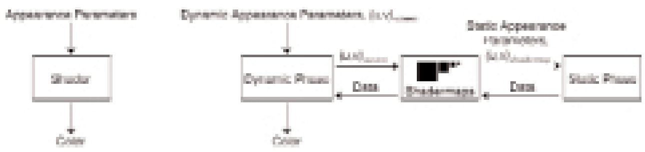


Figure 1. Dataflow diagrams for a traditional shader and a shader using shadermaps. On the right, the shader has been separated into its dynamic and static phases. The shadermap is similar to a cache between the two phases.

Simulation Fidelity Metrics for Virtual Environments Based on Memory Semantics

Contact
Katerina Mania
School of Engineering and
Information Technology
University of Sussex
United Kingdom
k.mania@sussex.ac.uk

Alan Chalmers
University of Bristol
United Kingdom

Tom Troscianko
University of Sussex
United Kingdom

Rychard Hawkes
Hewlett Packard Laboratories
Bristol, United Kingdom

A photorealistic, computer-generated interactive environment strives to achieve the same sense of space as in the real world. Subjective measures based on human spatial perception supplementary to accurate geometry, illumination, and task performance, reveal the actual cognitive mechanisms in the perception of a virtual environment that are not otherwise apparent.[2] In this sketch, we present a methodology for the assessment of simulation fidelity of virtual environments (VEs), centred on a validated theory of memory awareness states.[1] It is challenging to identify whether VE simulations, displayed on head mounted displays (HMDs) and related interaction interfaces have an effect on the actual mental processes participants employ in order to achieve a spatial memory task in a VE, compared to reality and more traditional displays.

One-hundred-and-five participants were involved in a study which investigates participants' accurate memory recall and awareness states of elements and objects in a VE replica of a real-world room displayed on a typical desktop monitor or on a HMD (mono, stereo, head-tracked, or non-head tracked). Each memory recall question included a choice between four awareness states for each object recall. Traditional memory research has established that "remember" and "know" are two subjective states of awareness linked with memory recollections. Some elements of a visual space may be "remembered" if they are linked to a specific mental image; alternatively, the information could be semantically (non-visually) recalled – in this case, it is said to be "known." Remembering refers to experiences of the past that are recreated with the awareness of re-living them mentally. Knowing refers to those in which there is no awareness of re-living any experiences. What has been experienced recently, although this recent occurrence can not be recalled, could feel "familiar." Also, recall could be reported as a "guess."

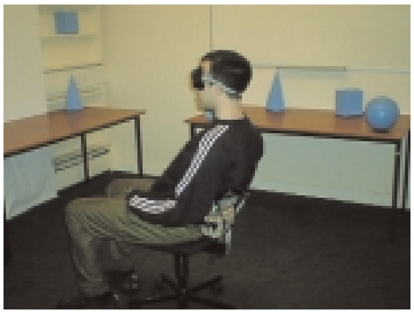
The radiosity rendering of the room was based on photometry data acquired in the real space. The resultant space memory recall and cognitive states as well as participants sense of presence is compared with that obtained from an analogous experiment in the actual physical space. The extent to which judgements of memory recall, memory awareness states and presence in the physical and VE are similar provides a measure for the fidelity of the simulation in question.

Overall, the level of presence was higher for the real condition compared to the HMD and desktop conditions. Across the technological conditions, presence and memory recall were, due to the high quality of the rendering, similar. Results show that the navigation method (head movements vs mouse) has an effect on the cognitive strategy adopted and therefore on the type of mental representation of the scene. In particular, the proportion of accurate responses under the "remember" state was significantly higher for the HMD-monocular-mouse condition compared with the HMD, mono and stereo head-tacked conditions. These responses revealed a weaker mnemonics strategy based on recollections of words for this particular condition, expressed by the lower proportion of correct responses under the "know" awareness state.

A VE system is likely to involve navigation in a synthetic space. Does the higher proportion of correct "remember" responses for the HMD-mono-mouse make this condition more "realistic?" The cognitive strategy is proven to be affected by the degree of realism of the motor response. The utilisation of a novel viewing method (HMD) plus a unreal motor response, such as the mouse, stopped the participants using this mnemonic – "unreal" – strategy and resulted in a more natural distribution of remember-know responses than even the real scene. By decreasing the degree of "reality" of the motor response, participants paradoxically adopt a more natural strategy. Something less "real," thus, but more demanding because of its novelty, may restore a more naturalistic cognitive strategy. By employing methodologies that have been examined and validated through decades of experimentation as the memory awareness states methodology, computer graphics research and VE technologies get closer to actually exploiting the human perceptual mechanisms towards successful applications.

Acknowledgements
This research is funded by Hewlett Packard Laboratories External Research Program.

References
1. Gardiner, J.M. (2000). Remembering and knowing. In *Oxford Handbook on Memory*, E. Tulving and F .I. M. Craik, eds. Oxford: Oxford University Press, 2000.
2. Mania, K. & Chalmers, A. (2001). The effects of levels of immersion on presence and memory. *Cyberpsychology and Behavior Journal*, issue 4.2.



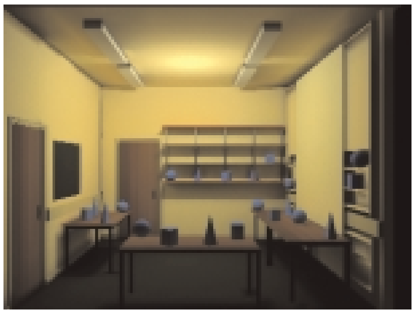


Figure 1. Real and VE.

Simulation of Deforming Elastic Solids in Contact

Contact
Gentaro Hirota
Department of Computer Science
University of North Carolina at Chapel Hill
hirota@cs.unc.edu

Susan Fisher
Andrei State
Henry Fuchs
Department of Computer Science
University of North Carolina at Chapel Hill

Chris Lee
Health Sciences Center
University of Colorado

In the simulation of human and animal bodies, complicated mechanical contact between nonlinearly viscoelastic tissues imposes a challenging numerical problem. The definition of the reaction forces that act on the interface (contact forces) is the key for designing a reliable contact handling algorithm. Traditional methods pay little attention to the continuity of contact forces as a function of deformation, which leads to a poor convergence characteristic. This convergence problem becomes especially serious in simulation scenarios that involve complicated self-contacting surfaces such as folding skin.

We introduce a novel penalty finite element formulation based on the concept of material depth, the distance between a particle inside an object and the object's boundary in a reference configuration. By linearly interpolating pre-computed material depths at node points of a finite element mesh, contact forces can be analytically integrated over contacting regions without raising computational cost. The continuity achieved by this formulation enables an efficient and reliable solution of the nonlinear system. This algorithm is implemented as a part of our implicit finite element program for dynamic, quasistatic, and static analysis of nonlinear viscoelastic solids. High nonlinearity and anisotropy, typically observed in biological materials, are also supported.

Toward Automatic, Realistic Human Motion from 3D Scans

To demonstrate the effectiveness of our method, we simulated flexion of a human knee joint. A finite element leg model with 40,000 tetrahedral elements was built based on the visible human male (Figure 1). Most of the boundaries between the various components are treated as frictionless interfaces. Various material parameters are assigned to tetrahedral elements in order to approximate the mechanical properties of different tissues. The femur is fixed in space. The cross section of the thigh is constrained on the cutting plane. The tibia is rotated 150-degree around an axis in the knee joint.

As shown in Figures 2 through 5, realistic effects such as skin fold and sliding contacts of tissues were obtained. To our knowledge, this is the first demonstrated simulation of large-scale motion of a complex model derived from the widely used visible human dataset and encompassing multiple tissue types including bone, muscle, tendons, and skin.

For more information: www.cs.unc.edu/~us/fem/

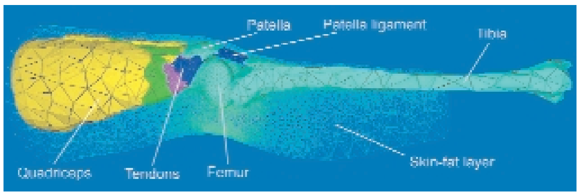


Figure 1. Constituent parts of the leg model derived from the Visible Human dataset.

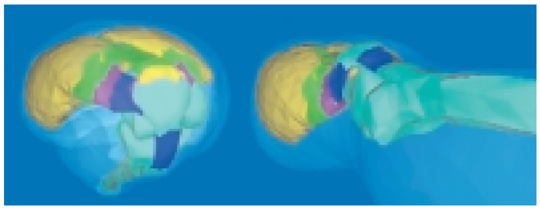


Figure 2. Knee in bent position (left) and stretched (initial) position (right). Note that the patella slides over the head of the femur.

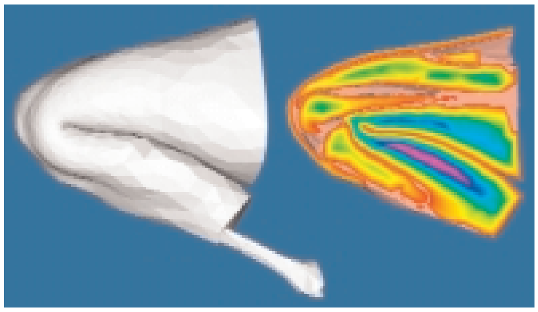


Figure 3. Skin surface of highly flexed knee (left), cut-away view of the same flexed knee (right). Only parts of the tibia and femur are visible in the cut-away since they are partly in front or behind the cutting plane. Note natural looking sliding contact between skin areas, skin and bones/muscles, patella and femur. The complex self-contact of folding skin was handled without revealing visible penetration. Pseudocolor encodes the value of material depth.

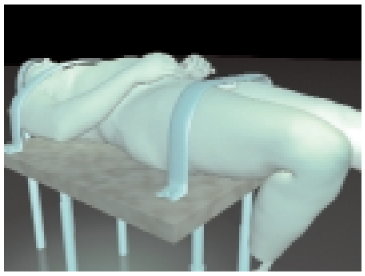


Figure 4. Visible Human dataset with flexed knee.

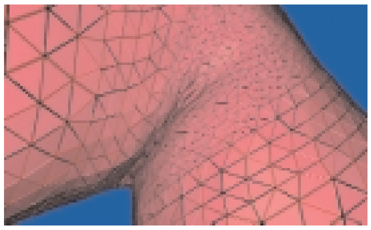


Figure 5. Close-up of the knee, illustrating the shape of skin folds.

Simultaneous Model Assembly in Real and Virtual Spaces

Contact
Eckhard Meier
GMD - German National
Research Center for Information
Technology
Schloss Birlinghoven
D-53754 Sankt Augustin
Germany
eckhard.meier@gmd.de

This sketch presents an application for the assembly of complex static models by using elements of a construction kit. Based on the Real Reality approach,[1] models are created in physical and virtual space simultaneously. By tracking the user's hand movements, the system generates a virtual reproduction of the original construction. In addition to the management of discrete model components, the system performs an analysis of the existing scenery. The spatial layout of individual building blocks is interpreted in such a way that adjacent elements are combined to interconnected units. The application's primary objective is to transfer the substantial behavior of physical building blocks to their virtual counterpart and to ensure extensive conformity of the real and virtual scene.

Connectivity

A construction kit is a set of building blocks following a consistent combinational principle. An application that aims at the reproduction of a physical construction kit's methodology has to reproduce the connecting behavior of each building block in order to prevent restrictions of the element's connecting properties on a structural level. The prototype specifically is designed to handle construction kits that are based on plug connections. For this purpose, the basic geometrical representation of physical building blocks is extended by joint structures. Joints dynamically establish links of building blocks during the term of assembly and store this information internally.

In addition, a joint structure defines a set-up for valid building block connections. The determination of valid links fundamentally relies on the path of movement of building blocks, since only this real world information is tracked by the application. To validate potential links, the system compares correlations among joints of adjacent elements and the direction of movement (DoM). Joints only become linked if their predefined direction of connection vectors (DoC) point in opposite directions and essentially lie in parallel to the DoM vector. Robustness is added to this approach by a tolerance vector perpendicular to the DoC vector. Its purpose is to eliminate discrepancies due to tracking data inaccuracies. A cone spanned by the tolerance and DoC vector is used to validate deviations of the DoC and DoM vectors to make a joint become connectable. Furthermore, the tolerance and DoC vectors dimension a cylindrical region, which is used to determine connectable target joints.

Assembly

Model assembly is an interactive and dynamic process that has to be reproduced by the application in real time. The application's main task within that process is to recognize modifications of the physical model and to map each alteration to its virtual counterpart. For this task, the in-betweens of each model state are of vital importance, since these time-frames decide about structural alterations of a model. Consequently, it requires the system to perform a precise analysis of each modification period in order to reproduce physical model iterations correctly. Each iteration can be subdivided into three sub-phases:

The first phase is the grasping phase that marks the beginning of a model alteration. A user automatically initiates it by grasping a building block. This action notifies the system that the block is taken off the model and will be rearranged. The system starts tracking the user's hand movements and revokes each disconnectable joint in dependency on the brick's motion. This separation process will subdivide the original model into a number of autonomous model subsets. The grasping phase directly terminates with model disassembly completion.

The second phase is the moving phase that lasts as long as the user keeps holding the grabbed model. Within this period the grabbed model's position continuously changes. The system's main task is to constantly track the user's hand movements, to determine the actual DoM vector and to update the spatial arrangement of the virtual models. In a following step all potentially connectable joints of the grabbed model and corresponding targets of immobile models are determined in dependency on the DoM vector. Performing this preselection is of vital importance, since the termination of the moving phase by releasing the grabbed model is in the total control of the user, and in consequence cannot be predicted by the system. Thus it is important to predetermine all relevant information for the subsequent assembly operation that depends on the grabbed model's motion.

The third phase is the release phase that realizes the reconnection of the grabbed model to the stationary models of the scene. This process does not have a real world complement due to the fact that the physical model is immediately in a valid static configuration. In contrast, this situation has to be accomplished explicitly in the simulation. Since the formerly grabbed model is still is an independent entity, it has to be integrated into the topological structure of the overall construction by linking all connectable joints. The completion of the release phase results in a consistent model scenario that graphically and structurally represents the physical model set-up adequately.

Reference
1. Brauer, V. & Bruns, W. Bridging the gap between real and virtual modeling: a new approach to human-computer interaction. In *Proceedings of IFIP 5.10 Workshop on Virtual Prototyping,* May 1996.

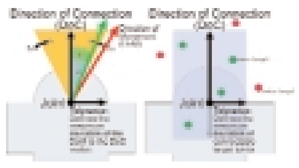
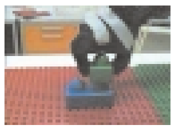
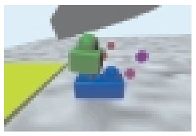

Direction of Connection (DoC)
Direction of Movement (DoM)
Joint
Direction of Connection (DoC)
Joint

Figure 1. Graphical represenation of a joint structure.





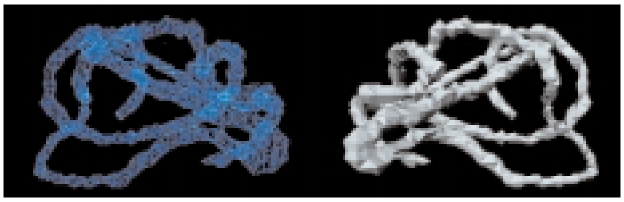
Figures 2 and 3. Synchronized real and virtual model.

Sketching Space: Virtual to Real

Bill Brody
Arctic Region
Supercomputing Center
University of Alaska, Fairbanks

Chris Hartman
Glenn G. Chappell
Arctic Region
Supercomputing Center

We have developed a method for turning our gestures into tangible reality. Eventually we hope for an interaction with computers unmediated by wires and clunky hardware, an interaction where the user can focus on the task at hand rather than the tool being used. One step on the path toward achieving this vision is our development of a way to create artistic forms in space. When we sketch, we energize space as we turn the moving position of our hand and the dynamics of our body as recorded by a wand into fields of potential. Then we go from potential energy to the creation of a mesh that meets the rigorous demands of the stereo lithography file format in order to produce an object that can be built into a tangible form.

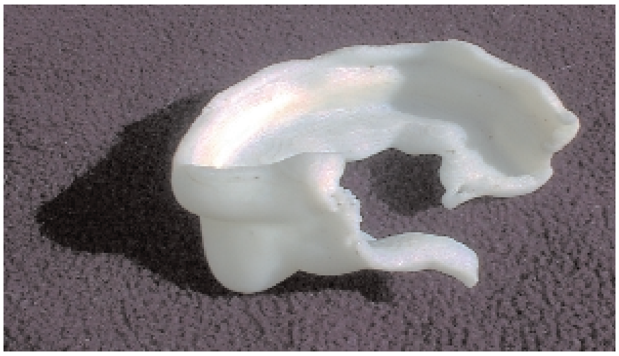

Example of an object created in BLUI and realized with rapid prototyping.

Our interface is BLUI, the Body Language User Interface. BLUI is a work in progress, a virtual reality 3D object creation and editing application that runs in the CAVE environment. One component of BLUI is our implementation of volume sketching that divides space into a cubical grid of cells. These cells are filled with potential energy dependent on the locus and dynamic character of how an artist draws in space. A surface is generated from this set of potential energy loci, translated into a stereo lithography file, and is then passed to a 3D printer for output as a solid object.

A video of an artist at work in our environment will show the body language that is a significant determinant to the character of the generated forms. The dynamic history of each developing form is reflected in the thickness, smoothness, and color of the virtual reality display. Our discussion will include the technical details involved in painting with potentials and creating isosurface meshes. We will also show a flyby of the scene; the viewpoint path and look at path are drawn using BLUI.

An elaboration of our development and implementation of a virtual reality drawing application on the way toward a body-centered user interface will explain and provide a context for the following items:

- Video of the artist at work. The relationship between the artist's body language and the scene will become apparent.
- Video recording of one channel (right-eye view) of the display.
  This will show the creation-editing history of an object.
- Tangible 3D objects sketched in BLUI and built in a rapid prototyping printer will complete the demonstration.


Example of a mesh suitable for rapid prototyping that was drawn in space.

Skinning Cats and Dogs Using Muscle Deformations

Contact
Hans Rijpkema
Rhythm & Hues Studios
Los Angeles, California
hans@rhythm.com

Brian J. Green
Rhythm & Hues Studios

Introduction

This sketch describes the techniques used for deforming animal models of cats and dogs. The process is anatomical in nature, using a rigid skeleton upon which muscles are placed. The deformations of the muscles are driven by the animation of the skeleton and in turn used to drive the deformation of a skin model. The process is entirely interactive, and the artist is given numerous controls to alter the output of the simulation.

Muscle Attachment

Since the skeleton motion is the driving force for the muscle motion, extra attention is paid during the character setup phase to ensure that the skeleton will behave in a physiologically sound manner. Muscles are then attached to the bones of the skeleton. When the skeleton is moved, attachment points may change position relative to each other. This motion causes the muscles to either stretch or bulge in order to maintain volume. The artist is given various controls to shape the muscles, such as tendon lengths and muscle volume. Since volume preservation plays a key role in muscle deformation, the artist is given controls to specify the total volume and the volume distribution of each muscle.

Muscle Models

To more accurately simulate the diverse behavior of real muscles, our software provides several muscle models. A ball muscle is ellipsoidal in shape and is useful for modeling such muscles as the biceps. A tube muscle is a surface created by interpolating two elliptical discs normal to a spline. By defining interior points for the spline, the artist can freely shape the muscle around the skeleton. The tube muscle is useful for modeling such muscles as the pectorals. In addition to ball and tube muscle, we have several other muscle models with more specialized purposes.

Skin Binding and Relaxation

Once we have placed the appropriate muscles on the skeleton, we use them to deform the skin model. The essential concept is to associate each vertex of the skin with surface points of nearby muscles. The primary factor to determine the amount of influence a muscle has on a skin vertex is distance: the closer the muscle the greater the effect. The binding process need only be performed once, at a neutral pose.

We allow multiple muscles to affect a single vertex. The resulting deformation is a weighted average of each contributing muscle's deformation. This reduces hard edges at muscle influence boundaries at the cost of increased computational complexity. The artist is given various controls to alter distance weighting and attenuation. In practice we found the use of such controls reduced the need to model large numbers of muscles and also allowed us to move the animal into extreme poses.

After the skin has been deformed by the muscles we must "relax" the skin. Distances between vertices need to remain similar between the neutral pose and the deformed positions. A dynamic relaxation algorithm is applied to the skin for this purpose. During the relaxation, we also perform collision detection with simplified bone geometry to simulate the rolling of skin over bone. As with the deformation process, the artist is provided with controls that can fine tune this process.

Solving a 3D Cube Puzzle in a Collaborative Virtual Environment: As Good as Really Being There Together?

Contact
Anthony Steed
Department of Computer Science
University College London
London, United Kingdom
A.Steed@cs.ucl.ac.uk

Ralph Schroeder
Ann-Sofie Axelsson
Alexander Nilsson
Ilona Heldal
Department of Technology and Society
Chalmers University
Göteburg, Sweden

Josef Wideström
Chalmers Medialab
Chalmers University
Göteborg, Sweden

Åsa Abelin
Department of Linguistics
Göteborg University
Göteburg, Sweden

Immersive projection technology (IPT) installations are proliferating around the world and increasingly they are being used in networked situations where users collaborate in a shared virtual environment. In these preliminary studies we investigated the ability of pairs of users to collaborate on a simple puzzle-solving task. We found that pairs of users who are both immersed in IPTs can perform almost as well as when they do the same task in the real world. But we also found that collaboration can be problematic when participants are using very different systems such as an IPT and a desktop.

Trial Settings

We compared performance and experience in solving a 3D cube puzzle between three conditions: two participants in real space (R condition), two participants in different IPTs (C2C condition), and one participant in an IPT and one on a desktop system (C2D condition). The participants in the R condition performed the task with real blocks. In the C2C condition we used a Tan VR-CUBE at Chalmers University and a Trimension ReaCToR at University College London. In the C2D condition we used the Chalmers Tan VR-CUBE and a SGI O2 workstation. The non-immersed participant used a mouse and keyboard to move and interact with the cube puzzle. The participants did not know what type of system their partner was using. The C2D condition was implemented in dVISE (now renamed DIVISION, www.ptc.com/products/division). For the C2C condition the environment was precisely replicated on the DIVE platform (www.sics.se/dive). Figure 1 shows a pair of immersed participants in the C2C condition at various stages of the task.

Performance and Findings

The R condition is a standard that we did not expect to be bettered in a simple collaborative virtual environment. Figure 2 shows the percentage of pairs that completed the task for each condition. Twenty-two pairs did each condition. Note that users were stopped after 20 minutes if they had not completed the task. The mean completion times were: R condition 5.75 minutes; C2C 8.82 minutes; and C2D 15 minutes.

In the C2C condition interaction between the two participants was very fluid, and in our opinion collaboration was much more successful than any of our previous experiences with similar systems or applications. In the C2D condition confusion seemed to arise between the participants because they did not comprehend how the other participant interacted with the world and the limitations that the other's interface might impose. Consequently we observed IPT users becoming frustrated with the slow performance of the non-immersed user and in post-trial questionnaires and interviews they rated the desktop user as contributing considerably less to the task or even being uncooperative.

Conclusions

Performance on the puzzle-solving task using IPTs was almost as good as performance on the task in real space. However collaboration and performance were very poor when one user was immersed and one user was not. We hypothesize, based on our observation and participant feedback, that this is not simply due to the effect of the non-immersed participant's poorer interface, but that a secondary factor is the confusion between the users about what the other is capable of. Further study is required to understand how to better support collaboration between virtual environment systems with such dissimilar interfaces.

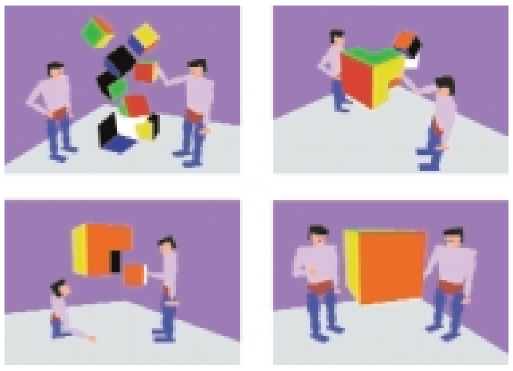


Figure 1. Two participants completing the 3D cube puzzle.

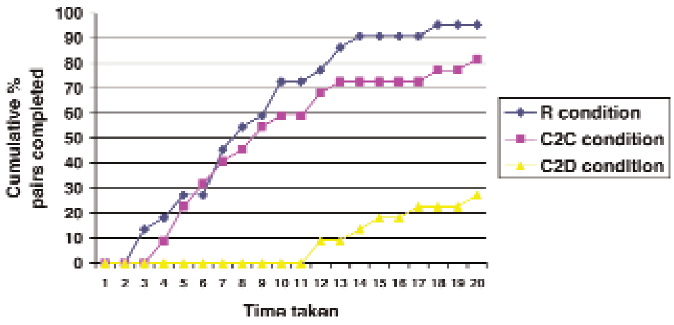


Figure 2. Cumulative percentage of pairs of participants that had completed the task by the given time for each of the three conditions. Note that not all pairs complete the task within 20 minutes.

Spatial Resolution in Haptic Rendering

Contact
Juli Yamashita
National Institute of Advanced Industrial Science and Technology
Tsukuba Central 6
1-1-1, Higashi, Tsukuba-City
Ibaraki 305-8566 Japan
yamashita-juli@aist.go.jp

Yukio Fukui
University of Tsukuba
National Institute of Advanced Industrial Science and Technology

Osamu Morikawa
Shigeru Sato
National Institute of Advanced Industrial Science and Technology

Introduction
The effect of spatial resolution in haptic rendering[1] remains to be well understood or studied. How much resolution is actually required? How sensitive are human beings? To answer such questions, we conducted psychophysical experiments to measure absolute thresholds of haptic smoothness in perceiving curved surfaces presented by two point-contact type force feedback devices with different spatial resolutions.

Experiments
Apparatus: The stimulus (Figure 1) was a polygonal approximation of a cylindrical surface by tangent planes. It was displayed haptically using PHANToM 1.0 T (nominal positional res.: 0.07mm) and A (0.03mm) (SensAble Technologies, Inc.). Subjects touched the stimulus and responded "yes" if they felt it to be a smooth cylindrical surface and "no" if not. Resolution angle α was varied in 1-degree increments and threshold angles of surface smoothness were measured by the method of limits. To avoid the effect of haptic bump mapping, the direction of feedback force was force shaded.[2]

Parameters: radius of stimulus: {30 60, 90, 120} (mm); surface stiffness: 0.5 (N/mm); force shading: {cylindrical, linear}. In the "cylindrical" condition, the feedback force vector magnitude was the surface stiffness (0.5 N/mm) times penetration depth of the cursor into the stimulus (mm), and its direction was that of an actual cylinder. In the "linear" condition, the feedback force vector was a linear interpolation of two vectors to the first and second tangent planes nearest to the cursor (Figure 1). Force applied by subjects was also recorded at several points on the stimulus.

Results
Figure 2 shows that the threshold angle is positively proportional to the curvature (1/radius) (R=0.7-0.9). The analysis of covariance on their regression lines showed that (1) inclinations do not differ statistically ($p<0.4$); (2) two lines for the linear condition do not differ ($p<0.9$); and (3) line pairs {T-Cyl., A-Cyl.} and {T-Cyl., T-Linear} are different statistically at 1% level. The magnitude of applied force was almost constant over the surface (about 0.7-1.0 N), but depended strongly on the individual subject. It is weakly positively proportional to the radius.

Discussion
Since the applied force has a constant magnitude, the shape that the subject actually touches should be a "contour surface" for force magnitude F (Figure 3). Let "*d*" be the maximum difference between the contour surface and a cylindrical surface inscribed in it. Figure 4 shows "*d*" calculated for measured threshold angles and their force magnitude. In conditions other than T-Cyl., "*d*" is independent of curvature and nearly equal to 0.1mm. Interestingly, their distribution shown by SD seems to be "digitized" by the spatial resolution of equipment.

Conclusion and Future Work
When the direction of feedback force is C^1 or more continuous, perception of haptic smoothness of a cylindrical surface is defined by the height of "bumps" on the surface, and its absolute threshold is about 0.1 mm. The value is useful for haptic display design and shape simplification. Our future work includes experiments on perceiving doubly curved surfaces and human sensitivity in the time-domain.

References
1. Salisbury, K. et al. (1995). Haptic rendering: programming touch interaction with virtual objects. In *ACM Symposium on Interactive 3D Graphics*, 1995.
2. Morgenbesser, H.B. & Srinivasan, M.A. (1996). Force shading for haptic shape perception. In *ASME Dynamics Systems and Control Division*, 1996.

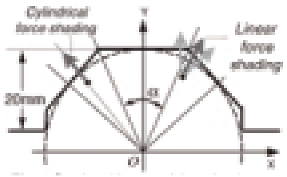


Figure 1. Sectional image of the stimulus. Angle α defines the size of each tangent plane that approximates the cylindrical surface. Black dotes denote cursor positions and dark gray arrows show feedback force directions for Cylindrical (left) and Linear (right) conditions.

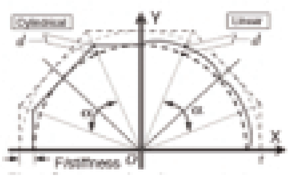


Figure 3. Contour surface for a constant force. When the subject touches teh stimulus shape with constant force F, the locus of the cursor follows the surface shown in solid lines.

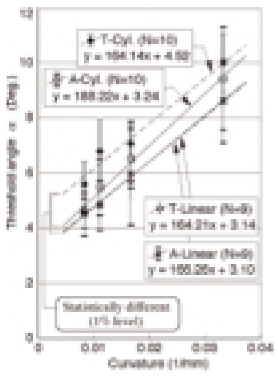


Figure 2. Results (Error bar: SD).

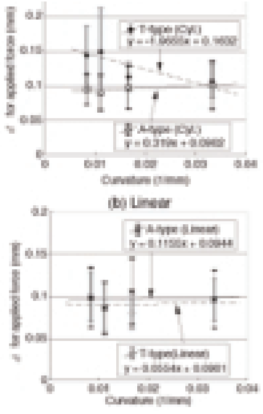


Figure 4. *d* calculated for threshold angle and applied force.

A Suggestive Interface for 3D Drawing

Contact
Takeo Igarashi
Brown University
CS Department, Box 1910
Providence, Rhode Island 02912
USA
takeo@acm.org

John F. Hughes
Brown University

We have designed an interface where users can construct 3D scenes as if they are sketching on paper without searching for editing commands in menus and buttons. Other gestural interfaces[3] have met this goal and have shown that carefully designed gestures can support fluent interaction. What we present here is a new type of interface that enhances gestural interfaces in several ways, using hinting and suggestions.

In the proposed suggestive interface, users give the system hints about the desired operation by highlighting related components in the scene, and the system suggests subsequent operations in an array of small thumbnails derived from the hints and the overall scene configuration (Figure 1). Users can complete an operation by choosing one of these suggestions, or ignore them and continue constructing and/or hinting.

A suggestive interface can be viewed as a mediated version of a gestural interface. Instead of responding to the user's input by updating the scene immediately, the system asks for the user's confirmation after showing multiple suggestions. This approach has several advantages over earlier gestural interfaces. First, the hinting mechanism lets us use existing components as input. This naturally helps in specification of geometric relations among components in the scene. Second, because suggestions are merely offered, a single collection of hints can serve both as a gesture and as a subset of a more complex gesture. Third, the suggestive interface helps the learning process because users can progressively refine their hints until the desired result appears among the suggestions.

The Chateau System

We have implemented a simple proof-of-concept 3D modeling tool (called Chateau) to explore the suggestive interface idea. Users construct 3D scenes by drawing 2D lines on the screen. The system converts 2D lines on the screen into 3D lines by projecting them onto 3D elements in the scene. Prediction[1] and suggestion facilitate this drawing process by offering possible subsequent operations. Highlighting plays an essential role throughout; highlighted lines guide the snapping during line drawing and provide hints for prediction and suggestions.

Figure 1 shows an example operation sequence. Users first highlight two lines, and the system presents three suggestions (a). Then they choose the first suggestion, which creates a new drawing plane (b). They draw two new lines on the drawing plane, unhighlight the vertical line, and highlight a horizontal line. The system presents three new suggestions based on the three highlighted lines (c). They choose the second suggestion and see the scene shown in (d). In this way, users can construct various 3D scenes by drawing, highlighting, and choosing suggestions with no explicit editing commands.

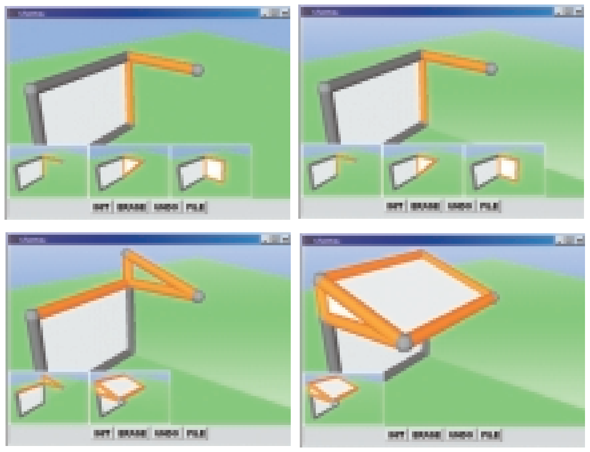


Figure 1. Example operation sequence: (a) highlight two lines; (b) choose the first candidate; (c) after a few drawing and highlighting; (d) choose the second candidate.

Candidates are generated by a set of suggestion engines. Each engine observes the scene, and when the current scene configuration matches its input test pattern, it returns the updated scene as a candidate. The behavior of an individual suggestion engine can be seen as a variation of the constraint-based search-and-replace operation in Chimera,[2] but our engines focus only on the highlighted lines instead of pattern-matching against the entire scene. The current implementation contains 21 suggestion engines, and Figure 2 shows a few examples.

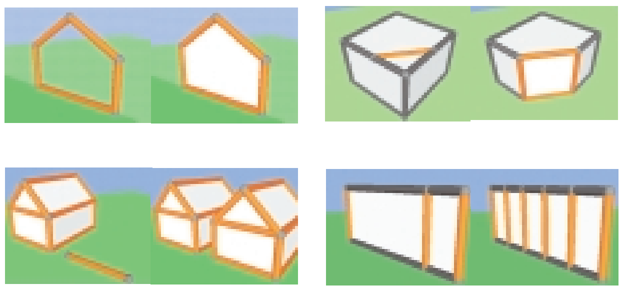


Figure 2. Example of suggestion engines (left: hints, right: result):
(a) highlight two lines; (b) choose the first candidate; (c) after a few drawing and highlighting; (d) choose the second candidate.
(a) polygon in a closed loop; (b) chamfering; (c) block copy; (d) equal division.

References

1. Igarashi, T., Matsuoka, S., Kawachiya, S., & Tanaka, H. (1998). Pegasus: A drawing system for rapid geometric design. *CHI'98 Summary*, 24-2.
2. Kurlander, D. & Feiner, S. (1992). Interactive constraint-based search and replace. *Proceedings of CHI'92*, 609-618.
3. Zeleznik, R.C., Herndon, K.P., & Hughes, J.F. (1996). SKETCH: An Interface for Sketching 3D Scenes. *Proceedings of SIGGRAPH 96*, 163-170.

"Summer Breeze:" Creating a High-End 3D Movie Using Limited Resources

Contact
Yaron Canetti
Sheng-Fang Chen
School of Visual Arts
133 West 21st Street, 10F
New York, New York 10010
USA
yaron@sva.edu

Cloth Simulations
Jimmy Chim

Vegetation
Anthony Patti

Collaborators
Tami Zori, Alen Lai,
Charlotte M. Griffin

Until recently, only a handful of studios could attain a certain degree of complexity in CGI. Working in a high-profile studio with a highly budgeted project has obvious advantages. However, this environment can be limiting, especially in regard to content. We believe that CGI technologies are quickly acquiring a degree of accessibility that will enable individuals to intuitively create sophisticated moving imagery that is competitive with major studios.

The purpose of this sketch is to demonstrate some of the techniques developed during production of "Summer Breeze" (working title), a short animation that is still in production by a group of students at the School of Visual Arts MFA Computer Art Department. These solutions are distinctive in that they were developed by art students, rather than programmers, using only off-the-shelf tools, so the solutions were "artist-friendly" and did not require extensive computer science knowledge.

The Movie

Our seven minute movie tells the story of a young woman's evolution into an independent person during the course of 24 dramatic hours. The story takes place in a Tuscan village in the 1970s. The visuals re-create the sensual beauty of a place we experienced and bring a warm atmosphere to life, setting the stage for expressive characters to live and unfold their story. Our story is subtle and aimed at adults. Our sources of inspiration are rooted in traditional art and movie-making history. They range from Miyazaki to Degas and Monet, from European comics art to the great tradition of American animation.

The challenges in achieving our artistic vision are many. Most of the past two years were spent on research and development. The School of Visual Arts gave us the space and resources we needed to start realizing this vision and the opportunity to build a versatile and committed team of people who are working without budget and free of commercial constraints.

The Tools

Technology is becoming more accessible to artists on two fronts: affordability and ease of use. This project cannot be realized without the latest technology available to the artist, and it is dependent on new tools that are yet to be released. However, none of us is a computer engineer. Our technical skills were acquired in conjunction with our artistic goals. We have tried to establish a unique, yet coherent, visual style for both characters and environments while keeping our files efficient and manageable within the limited resources available.

The characters are built to be free-moving and capable of expressing the wide range of feelings that our story evokes. They are fully modeled and textured nude, and their clothing doesn't serve as skin. Instead, clothing is a separate layer, caressing and complementing the body beneath. The environments are modeled and textured with the highest sensitivity to details, but at the same time our files are kept as light as possible.

Symplectic Ray Tracing: Ray Tracing with Hamiltonian Dynamics in Black-Hole Spacetime

Contact
Tetsu Satoh
JST/CREST
tetu-s@is.aist-nara.ac.jp

Haruo Takemura
Naokazu Yokoya
Nara Institute of Science and Technology

This technical sketch presents symplectic ray tracing, a novel approach to extend ray tracing in curved spacetime with black-holes. In conventional studies of visualizing black-hole spacetime, the path of light is computed by solving geodesic equations numerically. However, ray tracing based on the geodesic equation suffers from some problems concerning computational cost and accuracy of results. In order to overcome such problems, we have developed symplectic ray tracing based on Hamilton's canonical equation instead of the geodesic equation. Hamilton's canonical equation can be numerically solved by a symplectic process suited to long-time computation in black-hole spacetime.

Symplectic Ray Tracing

Original ray tracing technique[1] traces the light projected from light sources (or a viewpoint), assuming that the path of light draws a straight line. An equation of this straight line is given by:

$$\frac{d^2x^i}{ds^2} = 0, \qquad (1)$$

where x^i indicates the component of three-dimensional coordinates and *s* is a parameter of the straight line. In gravitational ray tracing,[2] extended from original raytracing, the following geodesic equation is employed to compute a path of light:

$$\frac{d^2x^i}{ds^2} + \Gamma^i_{kl}\frac{dx^k}{ds}\frac{dx^l}{ds} = 0, \qquad (2)$$

where Γ^i_{kl}, called Christoffel's symbol, a function to calculate curvature of space and other variables, is the same as in Eq. (1). By adding the term of Γ^i_{kl} to Eq. (1), Eq. (2) have some difficulties as follows. First, Eq. (2) is a second-order nonlinear differential equation of parameter s. Secondly, a suitable numerical method for solving Eq. (2) has been not proposed. Thirdly, it is not easy to concretely derive the geodesic equation of black-hole spacetime. From the above considerations, we propose symplectic ray tracing, is a new ray-tracing method using the following Hamilton's canonical equation instead of the geodesic equation:

$$\left\{ \begin{array}{ccc} \frac{dp_i}{ds} & = & -\frac{\partial H}{\partial q_i} \\ \frac{dq_i}{ds} & = & \frac{\partial H}{\partial p_i} \end{array} \right. , \qquad (3)$$

where q^i indicates the component of 4D coordinates, the same as x^i in Eqs. (1) and (2), p^i indicates the momentum of q^i, and H (Hamiltonian) is a function of q^i and p^i. Note that symplectic ray tracing needs eight-dimensional phase-space constructed of four coordinate components and four momentum components. Compared with Eq. (2), Eq. (3) is a first-order linear differential equation so that the suitable numerical method, symplectic numerical analysis[3] is applicable, and the concrete derivation of Hamilton's canonical equation for the black-hole spacetime is simple. It is especially important to introduce symplectic numerical analysis. Because non-symplectic numerical analyses such as classical Runge-Kutta methods and Euler methods are known to break the energy conservation law, they are not suitable for long-term calculation to trace the light in the universe. The main difference between the proposed and conventional methods for ray tracing in curved spacetime is whether symplectic numerical analysis is applicable or not.

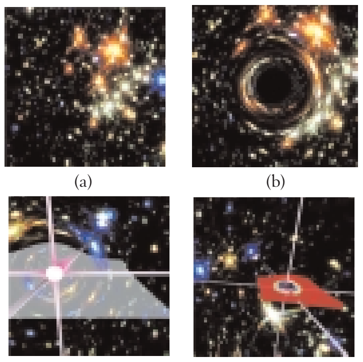


Figure 1. Visualization of a spherically symmetric blackhole: (a) no blackhole; (b) blackhole in the center; (c) superimposing x-y-z axes and x-y plane; (d) volume rendering of scalar curvature as well as three axes.

Visualization of Black-Hole Spacetime

We assume the universe can be modeled by a sphere and that a black hole is located at the center of the sphere. An image of galaxies[1] (credit: Don Figer and NASA) is mapped on the inside surface of the sphere. An observer is in the celestial sphere. Images perceived by the observer are generated with the proposed method as shown in Figure 1.

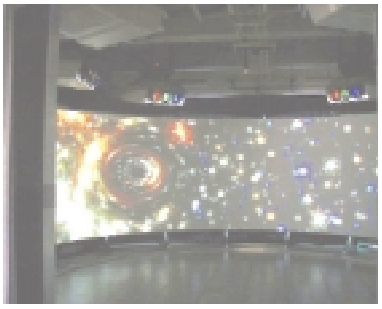


Figure 2. Immersive environment for observing a black-hole spacetime.

Because symplectic ray tracing is specialized for the Hamilton system, precise, fast computation is more possible than in conventional ray tracing with the geodesic equation. All images in Figure 1 are rendered on the SGI Onyx2 and Origin2000 with parallel computation by OpenMP. Each rendering took 30-60 minutes using 13 MIPS R10000 CPUs. We have also implemented the method on the cylindrical immersive projection display CYLINDRA(See Figure 2).

References
1. Goldstein, R.A. & Nagel, R. (1971). 3-d visual simulation. *Simulation, 23* (6), 25-31.
2. Yamashita, Y. (1989). Computer graphics of black holes: The extension of ray-tracings to 4-dimensional curved space-time. *Trans. Information Processing Society of Japan, 30* (5), 642-651, (in Japanese).
3. Yoshida, H. (1993). Recent progress in the theory and application of symplectic integrators. *Celestial Mechanics and Dynamical Astronomy, 56*, 27-43.

Tangible Communication

Contact
Ken-ichi Kameyama
Toshiba Corporate R&D Center
1, Komukai-toshiba-cho
Saiwai-ku, Kawasaki 212-8582
Japan
+81.44.5492286
kenichi.kameyama@toshiba.co.jp

Yasunobu Yamauchi
Toshiba Corporate R&D Center

In face-to-face communication, we establish mutual understanding and intimate friendship easily and naturally. However, it is not so easy to do the same thing in networked environments. For example, phones or email systems sometimes cause mutual misunderstanding. This is because current systems focus on transmitting only verbal or semantic messages. This sketch shows how to achieve more realistic communication over a narrow network with limited hardware resources like mobile computers. It focuses on developing a virtual handshake and interactive facial image deformation because these are simple but effective techniques for influencing affection.[1]

To produce a virtual handshake, we use small vibrators that represent the grasping motion of another human being. Bend sensors are used to detect handshake action. Figure 1 shows a rear view of the prototype system. About half of its surface is covered by black film backed with sponge, which simulates human skin. The vibrators stimulate hand locations (Figure 1) with very low frequencies. The bend sensors are attached to a soft plastic board at the front of the terminal (Figure 2).

The facial-image deformation algorithm is very simple. It applies the fact that we receive multiple impressions from only a single-fold facial image on paper if we look at it from different angles (Figure 3). Because its deformation can be realized by shearing and changing the ratio of only four stripe sections of the facial image (only four quad polygons), the facial animation can be smoothly updated without a powerful computational environment. The deformation is mapped into a tilt motion in the device. The facial image turns downward if another user tilts the device to the side; it turns right if the user tilts the device to the left, and so on. The image can also be shrunk by the user's handshake motion. To exaggerate the situation, funny paintings are imposed on the image (Figure 2). Sound effects are also added to the image deformation. The quality of the images is not high, but they are sufficient to make communication rich and interesting.

Related works include inTouch,[2] HandJive,[3] and Vibrobod.[4] But our approach is different because we employ multiple modalities. The prototype system is based on a small PC (Pentium II 266MHz), and all the data transmitted among terminal devices are managed by a server PC. Its data transfer rate is only 10KBps even when the facial images are updated every 30 ms.

References
1. Basics of Physiology, Vol.1, K. Fujisawa, et. al., Eds. ISBN4-7628-2114, 1998 (in Japanese).
2. Brave, S. & Dahley. A. (1997). inTouch: A medium for haptic interpersonal communication. *CHI'97 Extended Abstracts*, 363-364.
3. Fogg, B., Culter, L., Arnold, P., & Eisbach, C. (1998). HandJive: A device for interpersonal haptic entertainment. *Proceedings CHI'98*, 57-64.
4. Dobson, K., Boyd, D., Ju,W., Donath, J., & Ishii, H. (2001). Creating visceral personal and social interactions in mediated spaces. *CHI'2001 Extended Abstracts*, 151-152.

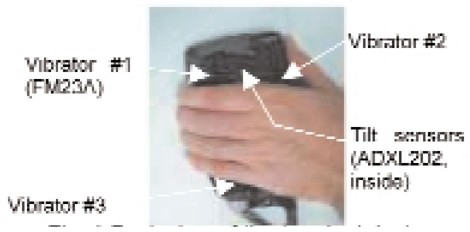


Figure 1. Rear view of the terminal device.

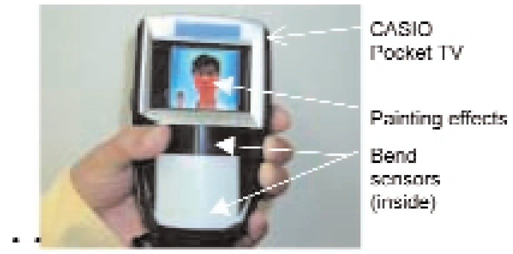


Figure 2. Front view of the terminal device.

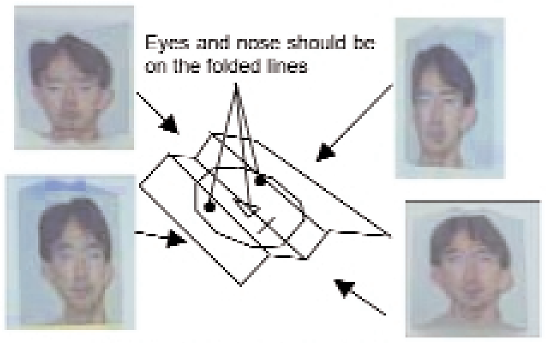


Figure 3. The principle.

Texture and Shape Synthesis on Surfaces

Contact
Lexing Ying
NYU Media Research Lab
719 Broadway, 12th Floor
New York, New York 10002
USA
+1.212.998.3389
lexing@mrl.nyu.edu

Aaron Hertzmann
Henning Biermann
Denis Zorin
New York University
www.mrl.nyu.edu/publications
/synthsurf

This sketch presents a novel method for synthesizing functions on surfaces from examples, including various types of textures and displacement maps. Our method synthesizes the function directly on the surface, avoiding many problems of traditional texture mapping. The synthesized results have the same qualitative visual appearance as a texture example provided by the user, without simply tiling the example, and without producing seams or distortion. The method is independent of the surface representation and works on meshes as well as smooth surfaces.

Synthesis Procedures

Our method generalizes recent image texture synthesis techniques[1,2,4] to surfaces. These algorithms generate each new pixel by matching its texel neighborhoods with the neighborhoods in the example. The center of the closest matching neighborhood is then copied to the target image. We have developed two synthesis procedures:

1. Multiscale synthesis,[4] in which textures are generated from coarse to fine
2. Coherent synthesis,[1] which greedily copies coherent patches from the example image.

We find that the algorithm synthesizes surface textures that are perceptually similar to those produced by their 2D counterparts.

Neighborhood Sampling

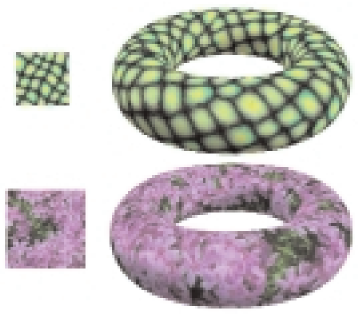
Unlike images, most surfaces of interest lack global parameterization or orientation. We establish orientation on the surface by computing a vector field in advance.[3] This field specifies the correspondence between orientation on the surface and orientation in the example texture. In order to compare surface neighborhoods with image neighborhoods, we first resample the function values on a regular sampling pattern near the surface point. This neighborhood can then be compared to an image neighborhood.

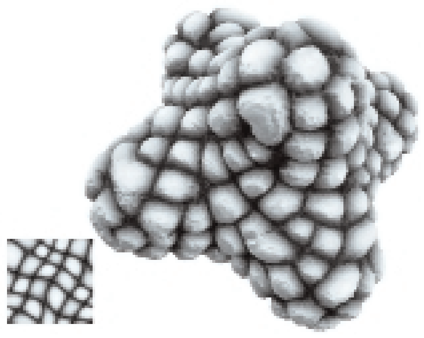
One approach to sampling a regular pattern is to march directly on the surface. However, this method is slow and sensitive to noisy surface features. Our implementation uses predistorted sampling in the parametric domain, which is much faster than surface marching. We synthesize directly to texture or displacement maps, allowing for efficient rendering without extra resampling steps.

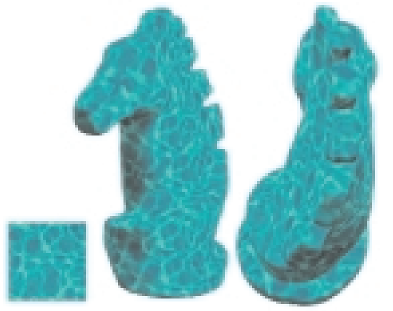
Results

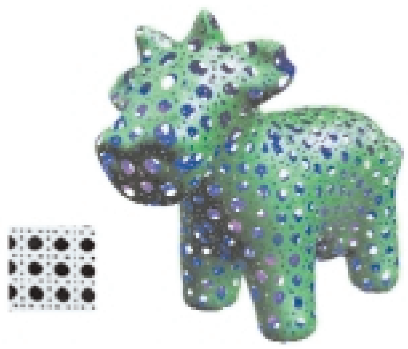
The images shown here demonstrate synthesis of texture, transparency, and displacement maps. They took between one to three minutes each to generate. The source textures are shown next to the resulting surfaces. For more results and details of the algorithm, see: www.mrl.nyu.edu/publications/synthsurf

References
1. Ashikhmin, M. (2001). Synthesizing natural textures. I3D 2001.
2. Efros, A. & Leung, T. (1999). Texture synthesis by non-parametric sampling. *ICCV 99*.
3. Hertzmann, A. & Zorin, D. (2000). Illustrating smooth surfaces. *Proceedings of SIGGRAPH 2000*.
4. Wei, L.Y. & Levoy, M. (2000). Fast texture synthesis using tree-structured vector quantization. *Proceedings of SIGGRAPH 2000*.


Top: multiscale synthesis; bottom: coherent synthesis.


Synthesis of texture/displacement map.


Sea horse. Synthesis on complex geometry.


Wicker cow: transparency map synthesis.

Time-Critical Volume Rendering

Contact
Han-Wei Shen
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210 USA
hwshen@cis.ohio-state.edu

Xinyue Li
Department of Computer and Information Science
The Ohio State University

Although the speed of volume rendering has significantly increased in the past decade, the size of the average volumetric dataset also continues to grow. The amount of data from a large-scale volume can easily overload underlying computational resources such as CPU speed or texture-memory capacity, which makes interactive volume rendering very difficult. To overcome this challenge, researchers have proposed to use hierarchical rendering algorithms to trade image quality for speed. While effective, most of the existing hierarchical methods rely on the user to make run-time decisions to select appropriate levels of detail. As the speed of volume rendering is a combination effect of 3D projection, transfer function, and visibility, it is a non-trivial task to predict the rendering speed for an arbitrary level of detail in the volume hierarchy. As a result, the quality/speed tradeoff is often done on a trial-and-error basis. For real-time applications such as virtual reality or computer-assisted medical surgery, relying on trial and error to tune the performance is impractical, as these applications require immediate responses, and performance requirements can change very frequently.

In this technical sketch, we present a time-critical volume rendering algorithm to tackle the aforementioned problem. The goal of our algorithm is to alleviate the user's burden of searching the levels of detail that can meet the performance requirement. With our algorithm, the user only needs to specify a desired wall clock time for each frame, and the algorithm will adaptively render data of appropriate resolutions from the volume hierarchy to complete the computation just in time. In addition, our algorithm allows the user to specify regions of interest (ROI) in the underlying volume. This ROI information is taken into account when allocating the computation time budget required to render the volume, so that overall image quality can be maximized.

In general, the run-time performance of hierarchical volume rendering is controlled by an error tolerance. A low error tolerance produces better quality, while a high error tolerance allows a faster rendering speed. We tackle the problem of time-critical volume rendering by designing an automatic error tolerance specification algorithm to guarantee the frame time. To achieve the goal, a feedback control system is constructed to monitor the rendering speed and adjust the error tolerance whenever necessary. We use a divide-and-conquer algorithm to implement the feedback control system. The algorithm starts by subdividing the underlying volume into several subdomains, hereafter called sub-volumes, and splitting the total computation time budget into smaller shares. Each sub-volume is assigned a share of the time budget. Given the subdivision, we feed each sub-volume into a feedback-control loop in a front-to-back depth order and perform a hierarchical rendering for each sub-volume using the corresponding branch in the volume hierarchy. For the first sub-volume, as there is no error tolerance given by the user, an initial guess is used for the rendering. At the end of this rendering, the actual rendering time is fed back to the control unit. With the feedback information, the control unit updates the available computation time and revises a new time budget for each of the remaining sub-volumes. In addition, the control unit estimates a rendering time for the next sub-volume based on the previous rendering result. This estimated rendering time will be compared with the sub-volume's allocated time budget, and the difference is then used to calculate a new error tolerance. This feedback control loop continues until all sub-volumes are rendered.

The key component in our feedback control system is a fuzzy controller that can choose an appropriate error tolerance based on past experience so that the next sub-volume can be rendered at the expected speed. The fuzzy controller adjusts the error tolerance at the end of each feedback iteration based on the difference between the estimated rendering time and the budgeted time for the next sub-volume. Intuitively, if the difference is high, a larger change to the error tolerance will be needed. Otherwise, the control unit will make a small or no change. Exactly how much change is needed is determined by our fuzzy inference rules.

A unique feature of our time-critical algorithm is that it is possible to take into account the importance of the sub-volumes and spend different fractions of the time budget on different sub-volumes. This is done by defining an importance function for the subvolumes and using the importance function to calculate the computation time budget for each sub-volume. We have designed a flexible time-budget allocation algorithm that can take into account multiple factors such as opacities, data errors, or gaze direction to control the rendering algorithm's interactivity.

We have conducted preliminary experiments for the proposed time-critical volume rendering algorithm and received very promising results. We integrated both the software-based ray casting algorithm and 3D texture mapping hardware volume rendering method into our control system. We are able to guarantee a five percent difference between the actual and the desired rendering time. In addition, we are able to accelerate both software and hardware volume rendering and receive high-quality rendering results by allocating more computation time to render important regions. Figure 1 shows two images generated by our time-critical volume rendering algorithm.

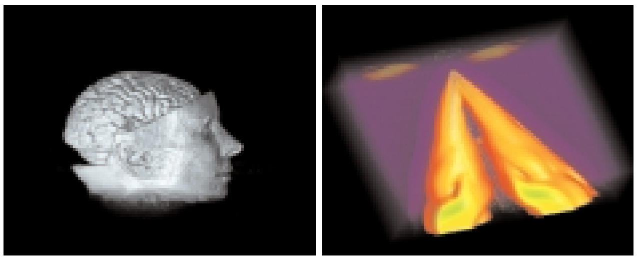


Figure 1.

Reference
1. Li, X. & Shen, H.W. (2001) Adaptive volume rendering using fuzzy logic control. Joint Eurographics-IEEE TCVG Symposium on Visualization. May, 2001.

Trees in "How The Grinch Stole Christmas:" Automating the Task of Creating and Rendering Large Numbers of Complex Objects

Contact:
Charlotte Manning
Digital Domain
300 Rose Avenue
Venice, California 90291 USA
charo@d2.com

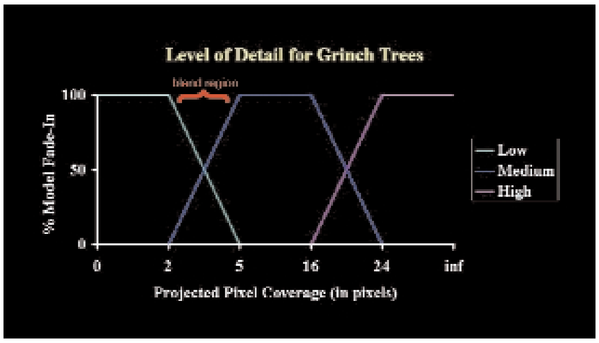
Many of the shots in "Dr. Seuss' How the Grinch Stole Christmas" involved partial or entire views of the terrain surrounding Whoville. Since none of the surroundings were built as a set, an entire CG environment had to be constructed so the director could roam through it freely with the camera. This environment had to be highly detailed, it had to be stylized yet photo-realistic, and the pipeline behind it had to be robust enough to handle a large volume of shots. The mountainous terrain had to be populated with tens of thousands of coniferous trees, complete with Seussian curlicues and sprinkled with snow. The trees represented a huge task, because there were so many of them, and they were present in nearly all the outdoor shots. As much as possible, we had to automate and streamline the tree-placement task in all these shots. This was accomplished at three different points in the pipeline: modeling, placement, and rendering.

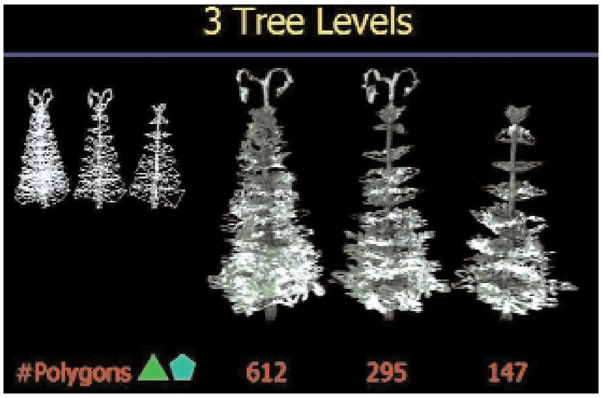
Automation at the modeling stage was done through procedural modeling with Houdini SOPs. A template network of SOPs was set up for a basic tree, and generating new tree variants was just a matter of making a few adjustments at different points in the SOP network. This was also useful for quickly increasing or decreasing the overall polygonal count of a tree without altering the look. Fifteen different tree types were modeled, each in different resolutions. This may not seem like enough for a whole population of trees, but it turned out to be sufficient when combined with our placement methods.

Placement of the trees was done with an object-instancing system constructed using the SOP network and particle system within Houdini. Each of the various tree models was randomly assigned to a particle-holding random scaling-and-rotation information that would later be translated as a transformation matrix for each tree instance. Although only 15 tree variants were modeled, the random transformations made it look as if all trees were individual and different in some way. Object instancing also greatly reduced the size of the .rib file (only the source models are geometrically defined; each instance then refers back to the original model definition and then applies its own unique transformation). The tree positions were also derived from the particles. Using a copy of the terrain geometry, the particles were manipulated to stick to the surface of the terrain and point upward (within an allowed range of deviation). In nature, trees tend to grow together in clumping patterns. Certain rules were established to cause the particles to stick to the terrain in clumps. Once an initial template system was set up, all the various control parameters could be adjusted for different shots: density, global size, clumpiness, randomness, etc. Setting up a new shot was then only a matter of loading new terrain or swapping out tree models if necessary, adjusting parameters and re-running the simulation.

When the trees were rendered, if the camera was static, regions of trees were classified for low-, medium-, or high-resolution models based on proximity to the camera. However, many of the shots involved a moving camera, so this step had to be automated. This was accomplished with RenderMan's Level of Detail feature, which involves specifying multiple resolutions of a model ranging from coarse to detailed, and a pixel-projection range for each resolution. The renderer automatically loads the appropriately sized model based on its screen projection. This greatly reducd geometric complexity, and the best performance gain (speed-up factor of 69.4! with 100 MB shaved off the memory consumption) occurred in the worst-case scenario, in which up to 40,000 trees were in frame at one time, many of them with a minute projection area on the image.

Completing the massive task of tree generation in a timely manner required taking these steps to make the pipeline fast, flexible, and user-friendly, so the artists could quickly and efficiently incorporate trees into their assigned shots.





Using Color-Changing Textiles as a Computer Graphics Display

Contact
Lars Erik Holmquist
PLAY research studio
Interactive Institute
Box 620, SE-405 30
Göteborg, Sweden
leh@interactiveinstitute.se
www.playresearch.com/

Linda Melin
PLAY research studio
Interactive Institute

The connection between textiles and computer graphics goes as far back as the early 19th century, when Joseph Jacquard developed punch cards to store textile patterns for weaving on automatic looms (the first digital image format). Surprisingly little has happened in the textile industry since then (punch cards are still widely used), but through the introduction of affordable computer-controlled looms and knitting machines, it is now becoming possible to weave or knit textiles with an ease approaching that of printing computer graphics on paper.

We are using these facilities of modern textile production in conjunction with another development: photochromically treated materials. Companies such as SolarActive (www.solaractiveintl.com) have created threads that change color when subjected to ultraviolet (UV) light (for example, from the rays of the sun or from a special lamp). For instance, a thread might be white in its original state, but change color to an intense blue or red when subjected to UV light. When the UV light is removed, the new color fades away after five to 20 minutes, depending on the material.

In our first experiment, we wove a fabric out of threads with different photochromic properties to create a curtain that would appear white during at night but become colorfully striped when subjected to the rays of the sun.[1] We are now developing ways to selectively turn portions of a fabric's color "on" and "off," much as if they were pixels, to create a dynamic display made entirely of textile. We believe it would be appropriate to use this material to construct displays that are more "calm" or "ambient" than traditional computer displays,[2] (for instance, by creating displays in the form of draperies or tapestries).

Figure 1 shows our first attempt at constructing such a device, where a UV lamp is mounted in a rack so that it can be raised and lowered by a computer-controlled motor. A piece of fabric is attached to the rack, and by moving the lamp, we expose different areas of the fabric to the UV light, thus revealing the "hidden" patterns. The fabric was created from two threads to get a red pattern on a blue background, and to demonstrate how it is now possible to incorporate digital images into textiles, we based the pattern on a version of the Utah teapot downloaded from the Web. This arrangement might be used as a simple ambient information display. For instance, if each teapot is associated with a person in a workgroup, the pots could indicate if the person is online (or perhaps even if the person is having a cup of tea, if that information could be made digitally available using some appropriate sensor technology!). Figure 2 shows another experiment, where we created a pattern that reveals new information when subjected to UV light. First, the textile is shown in its normal state; when it has been subjected to UV light, an additional piece of text becomes visible.

In the future, we intend to extend control over our display, either by adding a second motor, or by introducing a system of several lamps in parallel. We will also experiment with custom-made textile patterns for different types of information. Although these are early steps, we think they indicate one possibility for computer graphics to go beyond current screens and projectors to find new materials and modalities to display computer-generated information and images.

References
1. Melin, L. (2001). The information curtain: Creating digital patterns with dynamic textiles. *Extended Abstracts of CHI 2001*, ACM Press, 2001.
2. Wisneski, C., et al. (1998). Ambient displays: Turning architectural space into an interface between people and digital information. In *Proceedings of CoBuild '98*, 22-32, Springer.

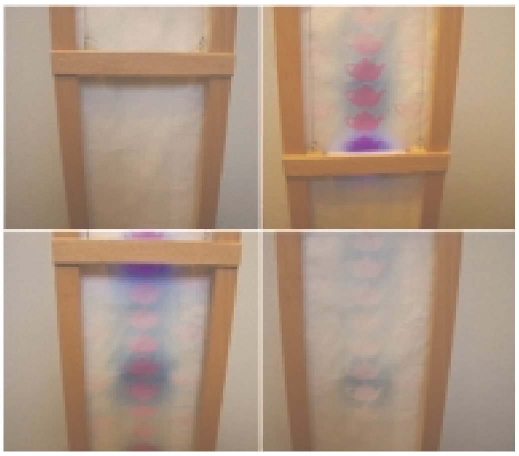


Figure 1. A textile is mounted in a wooden rack, and a computer-controlled motor moves a UV lamp over the fabric to create different patterns. The color changes remain visible for 5-10 minutes after the last UV exposure.

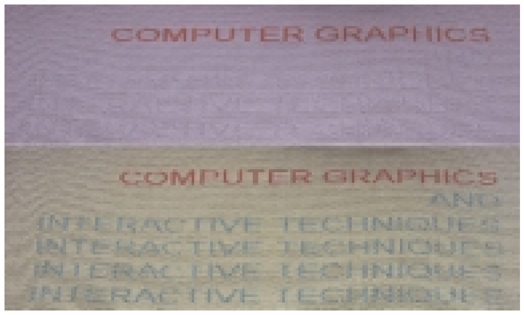


Figure 2. The top image shows another custom-created textile in its initial state; the bottom image shows it after it has been subjected to UV light, revealing additional text woven into the fabric. (The background has also changed color.) The new text remains visible for 5-10 minutes after the UV light has been removed.

Using Precomputed Cloth Simulations for Interactive Applications

Contact
Daniel L. Herman
Pixar Animation Studios
dh@digitalfish.com
www.digitalfish.com

This sketch presents a method for reconstructing the form of cloth dynamically draped on an animated body without the need to run a simulator during animation iteration. The reconstruction is interpolation-based and does not display hysteresis. Because the simulation is run off-line, the technique is suitable for use as a real-time preview mechanism to see roughly how clothing falls on a body while an animator manipulates the body interactively. It would also be useful in other contexts such as games.

Simulation Sampling

This technique is a hybrid of physically based simulation and kinematic pose-based animation. A computer model (to be used as input to a simulator) is automatically animated through a range of animation control values that fully explore the configuration space through which fast reconstruction is desired. The animation is "paused" (the rate of change in control values is held at zero) at discrete *sample points* that fall on a regular grid in the configuration space. The simulator is run on this animated motion sequence. Simulator output is extracted as a number of *poses*, one at each sample point, indexed with the animation-control values corresponding to that sample point. This yields a lattice of poses embedded in a space whose dimensionality equals the number of exercised animation controls (the dimensionality of the configuration space through which the model was exercised).

Rather than storing cloth poses directly, we compute a difference vector $\mathbf{D}_P$ between the simulated pose and some predictor pose that is easily computed from the animation-control values. $\mathbf{D}_P$ is expressed as per-point offsets relative to the predictor surface, and we store the lattice of $\mathbf{D}_P$'s. A good predictor algorithm is to feed the cloth mesh through the same sequence of deformations that affects the underlying body.

Pose Reconstruction

Reconstruction involves the synthesis of an output form given a particular setting of animation controls. Each animation control corresponds to an axis in the lattice space. For a lattice of dimension d, reconstruction involves multilinear interpolation among the 2^d nearest difference vectors, corresponding to the corners of the enclosing d-cube in pose space. Thus, reconstruction time depends only on the dimensionality of the space and is $O(2^d)$, which is constant-time relative to the total number of sampled poses n (given that the poses are gridded; ungridded sampling is discussed below). We use the predictor algorithm to generate a predicted form for the current animation control values, then apply the interpolated difference vector to obtain our final cloth geometry. This output exactly recovers the simulated cloth pose at sample points and interpolates nearby poses between sample points.

Data Representation

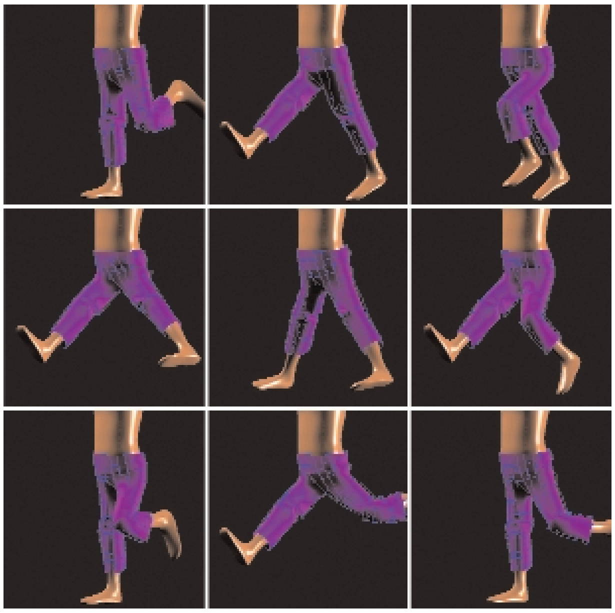
Systematic exploration of the pose space can lead to an enormous amount of data. We can take either or both of two strategies to combat this data explosion. The description above assumes we are sampling the space on a rectangular grid, but we could extend this to ungridded sampling, allowing us to selectively add samples wherever in the pose space we need more detail while throwing out samples in areas of the space that won't often be used. Ungridded sampling complicates the reconstruction process, however, as interpolating nearest points involves inserting a new site into the Delauney triangulation of the pose points, which is $O(\sqrt{n})$. However, ungridded sampling potentially allows a dramatic reduction in the total sample points n.

Alternatively, we can apply multi-resolution techniques in representing the sample lattice. The difference vectors generally vary by small amounts across local surface regions. We can exploit this to simultaneously reduce our data storage needs while reducing reconstruction cost by representing the pose data with wavelets. We can selectively throw away subtle surface irregularities while retaining major features such as pronounced creases and the general form of the cloth. In fact, this technique gives us a continuous level-of-detail control that trades off reconstruction accuracy for speed and storage.

Results

The images below show this technique applied to pants on animated legs. Both legs were considered together as a 6-dof articulated figure (2-dof in the hip + 1-dof in the knee, for two legs). Each dof was sampled at three values, giving a total of $3^6 = 729$ sampled poses. The predicted poses were computed by feeding the cloth mesh points through the body deformer network.

On this dataset, the results are very good. However, the rejection of hysteresis is both a strength and a limitation of the system. On a looser-fitting garment that displayed marked hysteresis, it is likely that our reconstruction would significantly misrepresent the actual motion of the simulated cloth.


Blue lines show simulated cloth. Magenta surface shows real-time reconstruction. Top row: exact recovery of simulated pose at sample points. Lower rows: approximate recovery via interpolation between sampled poses.

Variational Classification for Visualization of 3D Ultrasound Data

Contact
Raanan Fattal
The Hebrew University of
Jerusalem, Israel
raananf@cs.huji.ac.il

Dani Lischinski
The Hebrew University
of Jerusalem

Three-dimensional ultrasound (3DUS) is a relatively new technology for imaging distribution of ultrasonic echo information throughout a 3D volume of interest inside a patient. The main advantages of 3DUS over other imaging technologies is that the acquisition procedure is fast, non-invasive, non-radiative, and relatively inexpensive, which enables acquisition to be performed in clinicians' offices.

Unfortunately, compared to other volumetric medical imaging technologies, such as CT and MRI, high-quality visualization of 3DUS datasets is an extremely challenging task, since such datasets typically suffer from considerable noise and speckle, low dynamic range, fuzzy boundaries, and several other problems. These properties make extraction of smooth, or even continuous, surfaces extremely difficult.

In this sketch, we present a new method for opacity classification of 3DUS datasets, which is an essential step for displaying smooth surfaces of interest from volumetric data. Our method is based on the Variational Principle, a mathematical framework for optimization over function spaces. More specifically, we design a functional that imposes several simultaneous requirements on the opacity function. An optimal opacity function that minimizes the integral of the functional over the entire volume is then computed by solving the Euler-Lagrange equation, which in our method gives rise to a large but sparse system of linear equations.

The functional is defined as a weighted sum of three terms. The first term allows the function to attain non-zero values only in areas where the original volume has values near a user-specified isovalue. The second term forces the opacity function gradients to point in the same directions as the gradients of a smoothed version of the original volume. The third term requires the function to have a user-specified non-zero value, whenever possible. In particular, it pulls the function toward this value in areas in which the original volume exhibits high gradients. The resulting optimal opacity function essentially defines soft shells of finite, approximately constant thickness around isosurfaces in the volume. At the same time, the function is smooth and insensitive to noise and speckle in the data. It is also quite sparse (typically over 80 percent of the voxels are zero).

Once the opacity function has been computed, it becomes possible to visualize the corresponding surfaces at interactive rates, using existing techniques, such as Marching Cubes surface extraction or shear-warp volume rendering. Utilizing the sparsity of our opacity function, we also propose a new visualization method, oriented splatting, which associates a single splat polygon (oriented perpendicular to the opacity gradient) with each non-zero opacity voxel. The rasterization and texture mapping hardware is then used to compute the corresponding footprints on the image plane.

Figure 1 shows a visualization of a fetal 3DUS dataset before (left) and after (right) variational classification. Figure 2 shows images of four different 3DUS datasets produced using variational classification and oriented splatting. Such images are generated at interactive rates on today's commodity 3D graphics accelerators.

Additional information can be found at:
www.cs.huji.ac.il/~danix/research/3dus

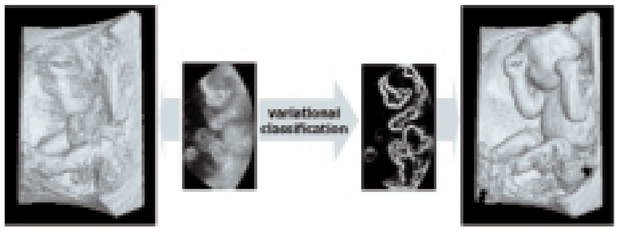


Figure 1. A surface extracted from 3D ultrasound data (left) vs. a surface displayed after variational opacity classification (right).

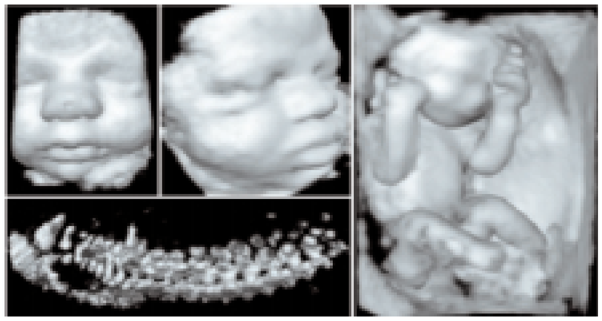


Figure 2. Examples: two different fetal faces (top); a spine (bottom); and an entire fetus (right).

VideoFOCUS and VideoFOCUSWire: Transforming Moving Images to Stills

Contact
Laura Teodosio
Salient Stills Inc.
268 Summer Street, Sixth Floor
Boston, Massachusetts 02210
USA
teo@salientstills.com

Joanna Berzowska
Salient Stills Inc.

In the pervasive discussion of media convergence, there is one type that is rarely hyped: the convergence between moving images and still images. Certainly, capture devices are beginning to blur the lines between these types of imagery. The latest video cameras include a still-capture mode, and many still cameras can produce a small digital movie. Editors looking for still images for print or electronic publications are also crossing the boundaries. Still frames from video footage, in particular, are often the only timely source for newsworthy still images.

We present a new model for image acquisition and distribution that uses video and other moving imagery as input. VideoFOCUS is a software environment that allows easy creation of high-quality still imagery from low-resolution moving imagery for use in print or online. And we introduce VideoFOCUSWire, a distributed version for still image creation and dissemination online. The vision is to make high-quality, timely images of all broadcast and video events generally available.

Problem

When images of a particular event are not yet available or do not exist in photograph form, publications turn to video. But there are many problems with using video for still imagery. The low resolution of video is inappropriate for print publications. Furthermore, there is no efficient method for finding a frame from a moving image sequence that can stand on its own as a still image.

Our goal was to create a system that is simple enough to make high-quality stills from video in the time-critical environment of a newsroom but is also robust and versatile enough to be used by a sophisticated imaging professional.

The result is the VideoFOCUS software. Image quality is addressed by using algorithms to take advantage of the great redundancy in moving-image frames. Images are aligned by sub-pixel optical flow and then statistically combined. By combining the data over frames, we can produce an image with less noise and higher overall resolution than any individual frame.[1]

The software allows monitoring and digitization of video. When capturing to disk, it records every single frame. This large body of data can be browsed in a movie player or as a storyboard of frames at eight different temporal resolutions, providing context from the surrounding frame content. This allows editors to easily find the frame that conveys editorial intent.

The created images can be exported at resolutions appropriate to print media, and they incorporate metadata in the news-industry standard IPTC headers.

VideoFOCUSWire

To take advantage of the large volume of existing video and film content and to address the issues of rights management and timely distribution, VideoFOCUSWire, a new model for still acquisition, is predicated on archiving, indexing, and dissemination of digital video in near-real time. It will give users access to a previously inaccessible body of video and images made from live broadcast events. By working directly with owners of news, sports, and entertainment video content, VideoFOCUSWire will provide a stream of time-critical still images and video clips to print and online media customers.

VideoFOCUSWire will continuously capture, index, and process licensed content. From a video server, editors can mark interesting video and images. Users then can either select from a pre-edited selection of stills or view a video stream of a live event. While viewing the stream, they can select frames to process using VideoFOCUS algorithms. Services such as user notification of new content, editorial decision-making, and transactions can all be delivered online or via a wireless device. VideoFOCUSWire will also incorporate security features and rights management.

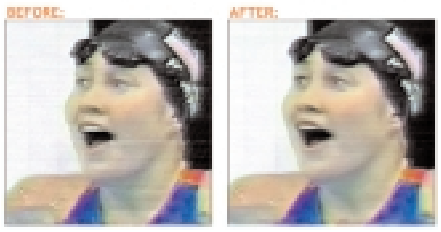
Future Work

One can imagine recording devices that would capture enough metadata to make the still extraction process more rapid and automatic. Additionally, we are exploring uses of the system for repurposing film and video content to create still products such as picture books from movies and photo-novelas from soap operas.

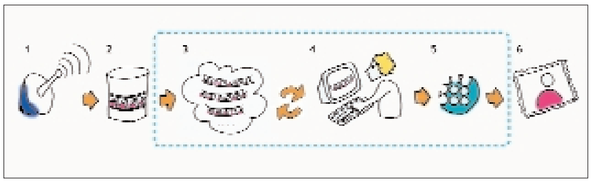
Contributors

Thanks to the many contributors, including: Walter Bender, Matt Antone, George Kierstein, and Jeff Hunter.

References
1. Teodosio, L. (1992). Masters thesis, Massachusetts Institute of Technology, September 1992.
2. Teodosio, L. & Bender, W. (1993). Salient video stills: Content and context preserved. *Proceedings of ACM Multimedia*, 1993.
3. Massey, M., & Bender, W. (1996). Salient stills: Process and practice. *IBM Systems Journal*, 1996.



By combining the data over frames, VideoFOCUS can produce an image with less noise and higher overall resolution than any individual frame.



VideoFOCUSWire will allow users to select any frame from a video stream of a live event.

View-Dependent Texture Mapping of Video for Realistic Avatars in Collaborative Virtual Environments

Contact
Vivek Rajan
Electronic Visualization Laboratory
The University of Illinois at Chicago
Chicago, Illinois 60607 USA
vrajan@evl.uic.edu

Damin M. Keenan
Daniel J. Sandin
Thomas A. DeFanti
Electronic Visualization Laboratory
The University of Illinois at Chicago

Satheesh G. Subramanian
Intel Corporation

Representing human users in virtual environments has always been a challenging problem in collaborative virtual reality applications. Earlier avatar representations use various techniques such as real-time animation of users' facial features, or green/blue screen chroma-keying, or stereo-based methods to retrieve users' video cutouts. The effort required to obtain a good degree of realism was enormous and was dependent on constrained lighting and background to extract the cutout of the user. So these methods were not applicable for situations outside of controlled studio environments or in virtual environments like the CAVE[1] that are inherently dark.

This sketch presents how view-dependent texture mapping[2] can be used to produce realistic avatars and, in the process, eliminate constraints posed by background and lighting requirements. A two-step approach is taken to achieve realistic 3D video avatars using projective texture mapping of video.

Approach

As a first step, an accurate 3D model of the user's head is obtained using a Cyberware laser scanner. This model is used as the head of the avatar, and the head tracking in the CAVE controls its motion. The coordinate system of the model and the tracker sensor element are aligned manually.

The camera that captures the video of the user is calibrated in the CAVE/tracker space using the method suggested by Tsai.[3] The tracking in the CAVE is exploited to calibrate the camera. Hence the accuracy of the calibration is dependent on the accuracy of the tracking system. With the hybrid tracking systems from Intersense, the accuracy of the tracking and the calibration is very good. Registration of the video image with the head model controls the realism of the final 3D avatar. The registration is not perfect when the user's lips move, since the head model is static. This error in registration is not very noticeable.

The camera is calibrated by tracking an LED in the tracker's coordinate system with its corresponding location on the image. Tracking the LED on the image is easily done using simple computer vision techniques to locate the point of high saturation and hue in the red region. The intrinsic and extrinsic properties of the camera obtained from the camera calibration are used to fix the camera in virtual space as a virtual projector. Since the head model is a reasonably accurate geometric model of the user's head, we can project the video (obtained from the camera) through the virtual projector to obtain a realistic avatar. The need for segmentation of the head from the background is eliminated, since the video is projected only onto the head model. This can be implemented efficiently in real time, as projective texture mapping is a commonly available feature in most polygon graphics hardware.[4]

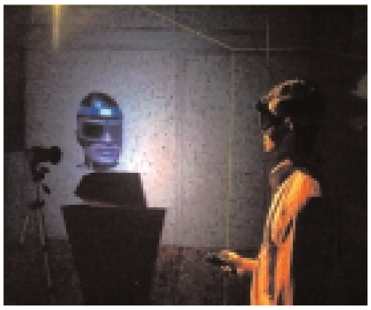
Transferring whole frames of video can be extremely network-intensive. By sending only the rectangular portion corresponding to a bounding box of video containing the user's head, bandwidth requirements are reduced considerably. The coordinates of this portion can be computed by projecting the corners of the bounding box of the user's head model onto the video image using the camera parameters obtained. Compressing this rectangular portion reduces the bandwidth further.

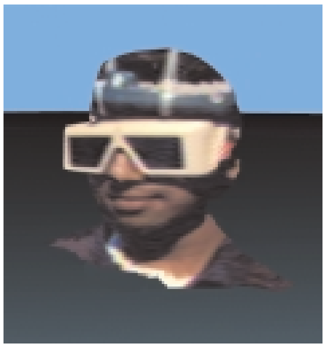
Using a single camera, the portion of the head facing away from the camera is not texture mapped. To overcome this problem we use another camera that captures the video of the person from another angle, and the video is projected onto the head model from the viewpoint corresponding to the second camera.

The result of the aforementioned method is a realistic video avatar that can be easily used in virtual environments like the CAVE.

References

1. Cruz-Neira, C., Sandin, D.J., DeFanti, T.A., Kenyon, R.V., & Hart, J.C. (1992). The CAVE: Audio visual experience automatic virtual environment. *Communications of the ACM, 35* (6), 65-72.
2. Debevec, P.E., Taylor, C.J., & Malik, J. (1996). Modeling and rendering architecture from photographs: A hybrid geometry and imagebased approach. *Proceedings of SIGGRAPH 96*, 11-20.
3. Tsai, R. (1986). An efficient and accurate camera calibration technique for 3D machine vision. In *IEEE CVPR* 1986.
4. Haeberli, P. (1992). Fast shadows and lighting effects using texture mapping. *Proceedings of SIGGRAPH 92*.


3D model of the user's head with the projected video texture.


Closer view of a video avatar.

Virtual Showcases

Contact
Oliver Bimber
Fraunhofer Institute for Computer Graphics
obimber@rostock.igd.fhg.de

L. Miguel Encarnação
Fraunhofer Center for Research in Computer Graphics

Bernd Fröhlich
German National Research Center for Information Technology

Dieter Schmalstieg
Vienna University of Technology

Introduction and Motivation

Intuitive access to information in habitual environments is a grand challenge for information technology. An important question is how established and well-functioning everyday environments can be enhanced by rather than replaced with virtual environments. Augmented reality (AR) technology has a lot of potential in this respect, since it allows augmentation of real world environments with computer-generated imagery. Traditional (even see-through) head-mounted AR displays, however, present many shortcomings with respect to unencumbered use, which prohibits them from being seamlessly integrated with habitual environments. In this technical sketch, we describe a new AR display system – the Virtual Showcase (see Figure 1) – and introduce developed real-time rendering and image transformation techniques.

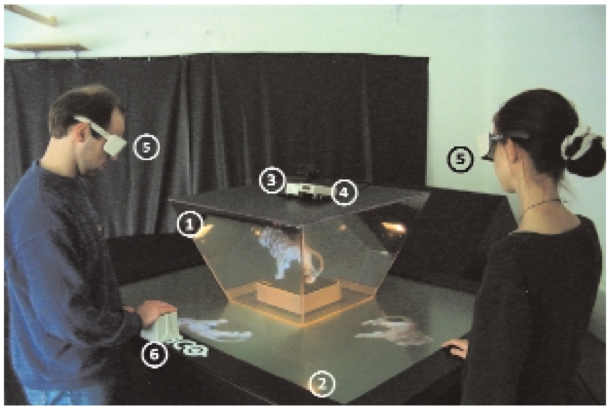
Conceptual Design

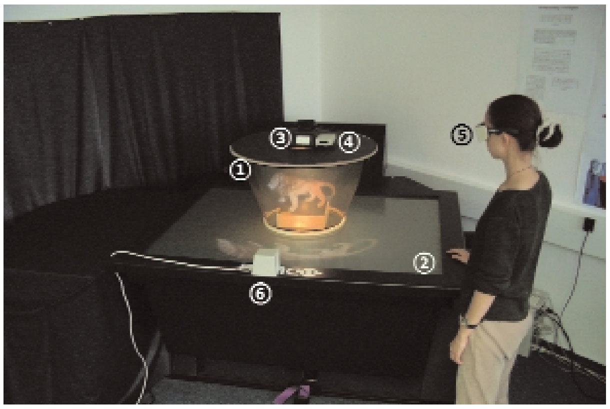
The Virtual Showcase has the same form factor as a real showcase, so it is compatible with traditional museum displays. Physical scientific and cultural artifacts can be placed inside the Virtual Showcase, where they can share the same space with virtual representations. The showcase's visuals can respond in various ways to a visitor's input. These interactive showcases are an important step in the direction of ambient intelligent landscapes, where the computer acts as an intelligent server in the background and visitors can focus on exploring the exhibited content rather than on operating computers.

Technological Setup

The Virtual Showcase consists of two main parts: a convex assembly of half-silvered mirrors[1] and a graphics display.[2] So far, we have built two different mirror configurations. Our first prototype consists of four half-silvered mirrors assembled as a truncated pyramid. Our second prototype uses a single mirror sheet to form a truncated cone. These mirror assemblies are placed on top of a projection screen[2] in both setups. Through the half-silvered mirrors, multiple users can see real objects merged with the graphics displayed on the projection screen. The showcase contents are illuminated using a controllable light source,[3] while view-dependent stereoscopic graphics are presented to the observer(s). For our current prototypes, stereo separation and graphics synchronization are achieved using active shutter glasses[5] in combination with infrared emitters,[4] and headtracking is realized using an electro-magnetic tracking device.[6] The cone-shaped prototype is particularly intriguing because it provides a seamless surround view onto the displayed artifact.

This work is supported by the European Union: IST-2001-28610.





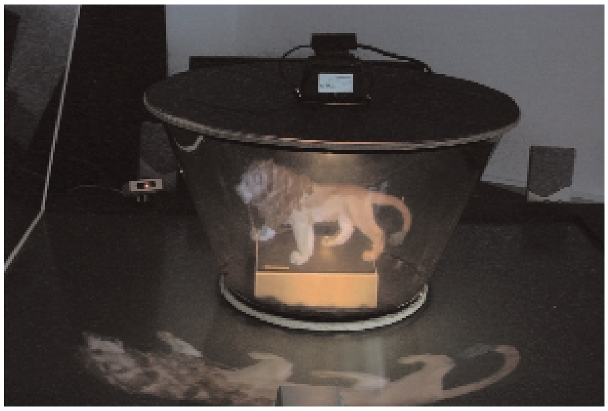


Figure 1. Two different Virtual Showcase configurations: a truncated pyramid (top) and a cone-shaped device (middle/bottom).

Visual Attention Driven Progressive Rendering for Interactive Walkthroughs

Contact
Jörg Haber
Max-Planck-Institut für Informatik
Saarbrücken, Germany
haberj@mpi-sb.mpg.de

Karol Myszkowski
Hitoshi Yamauchi
Hans-Peter Seidel
Max-Planck-Institut für Informatik

Introduction and Overview

Despite of the progress in (graphics) hardware development, interactive walkthroughs in photometrically complex scenes are still a challenging application, if high-quality rendering is required. Since full ray tracing at interactive rates is usually impossible, we render a pre-computed global illumination solution using graphics hardware and use remaining computational power to correct the appearance of non-diffuse objects on the fly. To obtain the best image quality as perceived by a human observer within a limited amount of time for each frame, we control corrective computation of non-diffuse objects according to a computational model of visual attention.

Our multi-threaded implementation consists of one *rendering thread* T_R , one *analyzer thread* T_A , and one or more *ray tracing threads* T_{RT_i} . T_R renders the scene using OpenGL hardware. Non-diffuse objects are rendered simultaneously into the stencil buffer with a unique ID code. The frame buffer is analyzed by T_A, and the resulting order of corrections is passed to the T_{RT_i} via a priority queue. Every sample that is computed by one of the T_{RT_i} is sent back to T_R , where it is splatted into the frame buffer using a stencil test to ensure that only the corresponding (non- diffuse) object will be affected by this correction.

Visual Attention Processing

To select and order the non-diffuse objects that need to be corrected, we extend the state-of-the-art model of attention developed by Itti et al.[1] This bottom-up model performs very well for static images and has been soundly validated by its developers in many demanding applications. Every input image is decomposed into a set of channels (intensity, color components), which are further decomposed into eight frequency scales each using Gaussian pyramids. By applying center-surround differences and across-scale normalization, a *saliency map* is generated, which encodes the saliency of objects into grey levels. To take into account the user's volitional focus of attention during interactive walkthroughs, we additionally consider task-driven factors such as distance from image center and pixel coverage for non-diffuse objects. Weighted blending between these top-down components and the saliency obtained from the Itti model results in a *visual-importance priority* assigned to each non-diffuse object.

Sampling and Splatting

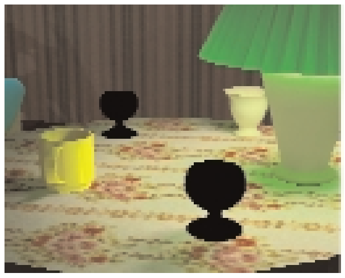
We use a hierarchical image-space sampling scheme to control ray tracing and splat the generated point samples with a square footprint. Due to the layout of the sampling scheme, which is adapted from,[2] the resulting image converges progressively to a ray-traced solution if the viewing parameters remain unchanged. Moreover, a sample cache is used to enhance visual appearance if the time budget for correction has been too low for some frame. We measure the validity of the cached samples based on the deviation of the dot product between the surface normal in the hit point and the current viewing direction with respect to the value of the dot product during sample generation. Valid samples are then reprojected into the current view.

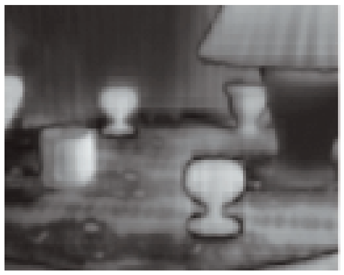
Results

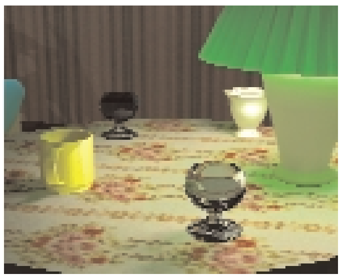
In our current implementation, we obtain frame rates of about 10 fps on an sgi Onyx3 for interactive walkthroughs in scenes with up to 100,000 primitives of various photometrical complexity such as mirrors, glasses, and glossy objects. We found the predictions of our visual attention model to be usually in good agreement with the user's fixations. We would like to extend our approach using some level-of-detail and occlusion culling techniques to speed up rendering for scenes of higher geometric complexity.

References

1. Itti, L., Koch, C., & Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *Pattern Analysis and Machine Intelligence, 20* (11), 1254-1259.
2. Stamminger, M. & Drettakis, G. (2001). Interactive sampling and rendering for complex and procedural geometry. *Rendering Techniques 2001 (Proceedings 12th EG Workshop on Rendering),* in press.







Different stages of our correction process. Top: An input image showing a view-independent global illumination solution computed in a pre-processing step. Middle: The saliency map resulting from visual-attention processing of the input image. Grey levels encode the saliency of objects. Bottom: A fully converged solution is obtained after corrections have been splatted according to the results of our attention model for about five seconds.

Whippin' up a Storm

Contact
Alex Seiden
Cyclotron VFX Studios
220 East 42nd Street
New York, New York 10017
USA
alexs@cyclotronNYC.com

What happens when two crusty old cowboys get hit by a tornado? A furious frenzy of visual effects work, especially when confronted by a looming Super Bowl air date and a dangerously delayed shoot. In only 14 days of CG animation, the team at Cyclotron's VFX studios whipped up a storm.

The rush began with principal photography in late December. Since the schedule didn't permit waiting until the spot was fully edited to start animation, selections for the backgrounds for CG shots were made before the final edit was completed. More than ever, carefully choosing which shots could be accomplished practically, which would need 3D CG, and which could be done with composites was critical.

The effects challenge began in early January. Our tools were Houdini for motion, RenderMan for rendering, and Inferno for compositing. We decided against building a complicated simulation or using any kind of volumetric approach. Our motion systems were relatively coarse (on the order of hundreds of particles, rather than hundreds of thousands); procedural shading filled in fine details.

Our usual procedure is to have our CG technical directors composite their own shots, since they are most familiar with the technical and aesthetic details. Also, the interplay between issues of rendering and compositing is often too tight to make splitting up the different pieces sensible. However, because of the tight time constraints, we had the CG team do rough comps and used our Inferno for the finals.

We deliberately kept the crew small. Although the tendency when deadlines loom is to throw bodies at the problem, we felt that this would only make things slower and more complicated (the "mythical man-month" fallacy).

Cyclotron's team for this project:

Visual Effects Supervisor
Alex Seiden

Visual Effects Producer
Aliciane Smythe

Senior Technical Director
Scott Petill

Technical Director
Kevin Gillen

Digital Compositing Artist
Joanne Ungar

Modeller
Scott Fink

A montage of in-progress images.

X3D Content Examples, Editing, Conformance Suite, and Software Development Kit (SDK) CDs

Contact
Don Brutzman
brutzman@nps.navy.mil

Mike Kass
Nicholas Polys

Extensible 3D (X3D) graphics is the XML version of the next-generation VRML 200x specification. X3D-Edit is a free graphics file editor for X3D that enables simple error-free editing, authoring, and validation of X3D or VRML scene- graph files. In this sketch, we show how X3D-Edit has been used to create hundreds of example scenes, including all examples from the VRML 2.0 Sourcebook. We also show how to install and use X3D-Edit. Validation of content before it is published is immensely powerful. Reducing content bugs and eliminating the syntax idiosyncrasies of VRML really helps! Context-sensitive tooltips for every node and attribute in a scene graph have made X3D-Edit an excellent tool for teaching beginning and intermediate 3D graphics.

The Web3D VRML and X3D Conformance Test Suite combine a body of work that originated from the NIST VRML97 Conformance Suite. Those tests are now available in XML format, allowing the tests to evolve with the VRML and X3D specifications. This suite has been developed primarily by the U.S. National Institute for Standards and Technology (NIST) and is now maintained by the nonprofit Web3D Consortium.

The test suite, consisting of approximately 800 tests, allows viewers to evaluate their VRML97 and (in the future) X3D browsers online, by simply browsing through an online directory of documented tests. Tests are broken down by node-group functionality (for example geometry, lights, sounds) and include tests of browser state, field-range testing, audio/graphical rendering, scene-graph state, generated events, and minimum conformance requirements. Each test consists of an XML (X3D) file, its VRML97 equivalent file, and an HTML "pretty print" version of its XML content for inspection. Also included with each test is a complete description of initial conditions of the test and expected results. In addition, "sample results" in the form of JPG or MPG video are also provided via hyperlink to give the tester a complete "picture" of a successful test result.

Because of its simple design and usability, the Web3D VRML/X3D Conformance Test Suite lends itself to easy test submission as well. The future of Web3D testing lies in public, Web-based submission of XML files with a simple test design structure Web-based test reporting is another feature that will be added to this suite in the near future, further empowering people to participate in the evolution of 3D on the Web.

Finally, we present the Web3D X3D Software Development Kit (SDK) CDs (sdk.web3d.org). The kit provides a huge range of content, tools, applications, viewers, and source code. The primary purpose of the SDK is to enable further development of X3D-aware applications and content. The SDK CDs support each of the other X3D presentations at SIGGRAPH 2001. Primary contributions are demonstrated, and free CDs are given to attendees.