A Platform for Mobile 3D Map Navigation Development

Antti Nurminen
Helsinki University of Technology
antti.nurminen@hut.fi

ABSTRACT

We present a platform for developing mobile 3D map navigation interfaces in highly occluded urban environments. The platform enables use of realistically textured city models in mobile devices such as PDA's and smart phones without 3D hardware, being able to render entire city centers at interactive rates. The platform supports network-distributed annotation of the environment with location-based information. Texture resolution can be varied per building. Maneuvering is not restricted, allowing implementation of any navigation metaphor. Efficient collision avoidance is provided. Key optimization methods, the overall system and supported features are presented along with guidelines for efficient modeling. Generic feedback from a focused field experiment is discussed.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: Graphical user interfaces; I.3.6 [**Computer Graphics**]: Interaction Techniqes; I.3.7 [**Computer Graphics**]: Virtual Reality

General Terms

Experimentation, performance

Keywords

Mobile 3D maps, 3D user interfaces

1. INTRODUCTION

Mobile 3D maps are applications that represent environments using true 3D data for visualization, which can be realistic. Such a view is supposedly more intuitive, as the representation allows direct matching of visual cues without interpreting and mentally transforming the symbols and two-dimensional shapes of the classical 2D map to the real environment. Subsequently, orientation to a new environment and navigation in it should become easy. In urban environments, such applications could also allow annotation of the scene by location-based information, or provide access to dynamic, real-time data.

Early studies on 3D maps often attempted to use PDA devices with direct model viewing software such as PocketCortona libraries. However, due to insufficient rendering performance even with simplified 3D city models, field experiments were conducted using prerendered images [9] or laptop emulation [10]. To further limit the required rendering power, models have been divided into a simple grid structure [8], or visibility information has been included into the model [3]. Most of the research hints that mobile 3D hardware would be in key position in solving the problem of poor performance [9, 3], or that sofware-based rendering of realistic models would even be impossible at interactive rates [2]. The development of mobile 3D maps and navigation interfaces has been seriously hindered by the lack of an efficient 3D engine and suitable 3D models that would allow such development and field experiments.

According to our benchmark tests, current mobile devices such as smart phones and PDA devices possess sufficient computational resources for producing realistic full-screen 3D graphics at interactive rates. Therefore, application of real-time 3D rendering for large scenes, such as cities, should be a mere question of optimization. The project *m-LOMA* attacked this technical challenge successfully [6]. We present the resulting platform and the features for 3D navigation development.

2. A 3D ENGINE FOR CITIES.

In the m-LOMA project, a 3D engine was created that would run realistic but lightweight 3D city models at interactive rates from any viewpoint, without restrictions on navigation metaphors. Mobile platforms have limited computational resources (CPU power, memory, storage and wireless network speed), and all computations consume batteries, so it is advisable to perform as much computations offline as possible. The following shortly describes a few selected optimization principles used in the m-LOMA engine, which are applicable to any efficient 3D city map.

Visibility. Urban environments are highly occluded: buildings populate the environment, and only the closest façades are visible from the street level. This scenario suits the *potentially visible set* algorithm [1] well. We divide view space to a three-dimensional grid and perform approximate visibility calculations as a preprocess with a predefined visibility range. Furthermore, objects contributing only little to the rendered image (the *hardly visible set*) can be discarded.

MobileHCI'06, September 12–15, 2006, Helsinki, Finland.

The result is a set of visibility lists, difference packed to clusters. At run time, we need to render only those objects already deemed visible.

Lightweight modeling. In computer graphics, visual information is well conveyed with textures, digital samples of surface colors, which are fast to render. A color texture doesn't contain all surface properties such as reflection, and is dependent on lighting conditions at sample time, but nevertheless provides a good visual approximation of an otherwise potentially complex surface. Contrary to textures, geometry requires lots of computations. For buildings with relatively flat façades, the geometry can be rather simple, at least to a certain scale. We choose to create a reference model for mobile 3D maps by approximating façades with flat, textured surfaces and let later user studies validate the decision. Textures are created by digital photography, applying image processing. In addition to buildings, we include statues in our model, to provide unique cues to the environment. We approximate them as *billboards*, flat textures that always face the viewer. Figure 1 compares the reality and the m-LOMA model.

Figure 1: A lightweight, textured model (below) and the real world (up).

Memory management and textures. Mobile devices lack run-time memory, which is easily consumed by textures. m-LOMA features an explicit memory management system with internal cache to limit memory usage. Only one level-of-detail (LOD) version of a texture at a time is used, namely the one that matches the estimated resolution at screen. LOD textures are created during the preprocess. The internal cache holds JPG versions of textures, and prioritizes small, recently used textures. In certain situations, such as when the viewpoint is raised above roof level, several dozen new textures may be required immediately. We limit loading to one texture per rendered frame to avoid I/O congestion. Common textures are always shared.

Frustum culling. While overall visibility is determined as an offline process, still most of the potentially visible objects lie outside our view space, the *view frustum*. We apply temporal coherence and perform a two-level hierarchical frustum test for only a part of the objects per a rendered frame.

Implementation. The client was programmed using the standard OpenGL ES 3D API with the C language, applying several rendering optimizations. Implementing m-LOMA on Symbian required substantial reprogramming.

Engine performance. Our reference model of Helsinki contains 14 statues and almost 300 buildings, of which about 200 are textured. The model contains a total of 475 individual textures, packed to 2-9MB depending on the allowed maximum resolution. The complete model consists of only 12610 triangles, totaling 714kB of binary meshes. Visibility lists require 2.6-6.3MB. Table 1 presents performance for a 1.7GHz Pentium M laptop with NVIDIA Quadro FX700 graphics, a 604MHz PDA (Dell X30) and a smart phone (Nokia 6630). 10fps is commonly reached. The laptop demonstrates the effect of hardware acceleration with over 100fps in all situations. The configurable memory management limits the use of memory. In smart phones it can be even less than 4MB, sacrificing cache efficiency.

Platform	Dell M60	Dell X30	Nokia 6630
Resolution	1024x768	240x320	176x208
View distance	800m	500m	300m
Street	300-600fps	20-25fps	10-15fps
Sky	150-250fps	8-15fps	7-12fps
Top-down view	200-300fps	10-16fps	8-14fps
Memory usage	12-50MB	$<$ 16MB	4-8MB

Table 1: m-LOMA frame rates and memory usage.

3. M-LOMA SYSTEM

The overall system block diagram is presented in figure 2.

Preprocess. The m-LOMA preprocess accepts a VRML model with JPG or PNG textures. For a city, it is advisable to model with at least two levels of hierarchy, where entire buildings are grouped from separate `Transform` nodes consisting of roofs and walls. The hierarcy, with building names, is stored into a separate file. The preprocess outputs visibility clusters, binary meshes (walls and roofs, the leafs in the scene graph) and LOD versions of textures. Meshes are given a default color based on the dominant color of their texture. Everything is packed to cache files that can be preinstalled into mobile devices.

Server. A server provides model data and location-based information to clients via a possibly wireless internet connection. It also pushes dynamic data, such as annotations created by other users, to clients. The server supports tracking of GPS enabled users, at their discretion.

Clients. Clients browse the current visibility lists (from local cache or server), load meshes and textures, and render them. Loading, be it from local cache or a server via wireless networks, doesn't stall rendering. Location-based information, represented by billboard icons, is rendered last.

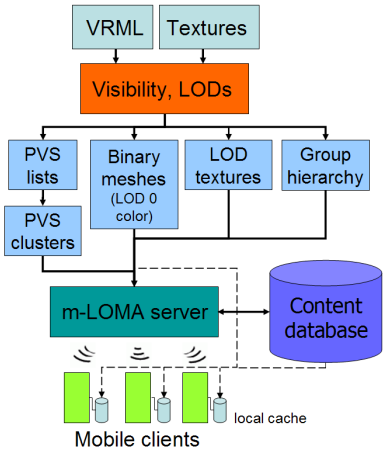

Figure 2: m-LOMA overall system.

4. SYSTEM FEATURES

The m-LOMA system provides several features, which are accessible via configuration files, function calls, or state variables for which more advanced functionality can be programmed. Figure 4 presents some of the described features.

3D rendering. The basic service of the system is efficient management and rendering of a textured 3D model from any viewpoint. Models can cover large areas and contain hundreds of textures, and be renderable, given lightweight geometry. Texture LODs can be changed for individual buildings in *administrator mode*. The client executables can be installed on desktop machines, PDA devices running MobileWindows, or Symbian Series 60 smart phones.

Widgets. The system provides a set of lightweight widgets such as buttons, menus, lists, tickboxes, scrollboxes, a virtual keyboard etc. These can be used to create common user interface components. Figure 3 presents an interface for routing.

Collision avoidance. Collision avoidance has been implemented efficiently, and the effect on rendering speed is negligible. It can be turned on or off with one function call. When on, the viewpoint slides along the colliding surface.

Annotation. The model can be annotated at run time by billboard icons, that can represent cues, location-based information, user messages or bookmarks. These icons should be collected into one aggregate texture, along with a configuration file. Administrators can integrate external content databases, such as restaurant or tourist databases. Each annotation also contains a bulletinboard, where users can have discussions.

Picking. Anything in the scene can be selected or queried. The pick coordinates and object pointers are returned, which can be used to launch menus, retrieve further information or for example set route start or end points.

Routing and street maps. Routing functions return a path between two points. Routing topology can be defined in two ways. One can draw connected node points in *roadmapping mode*. Or, a separate topological road set can be constructed from map data, yielding a scalable 2D street map and facilitating address queries. The system can also view traditional 2D raster maps, given they are divided to suitable pieces, and the coordinate system calibrated via a configuration file.

Landmarks. To facilitate visualization of *major landmarks* that lie far beyond the view frustum but are probably visible to a user, a set of icon files can be configured to be rendered whenever the related model is not visible. In addition, *minor landmarks* such as statues or other identifiable cues can be tagged. At run time, positions of local visible landmarks can be queried.

Markers. The system provides marker arrows that point to an appointed target, displaying the distance or the target's name. Markers can be assigned automatically, for example for route start and end points, or used for bookmarking locations, or attached to other users. Markers can be picked, for example to animate the viewpoint to a target.

Dynamic data. Dynamic data, such as GPS enabled users, or public transportation vehicles, can be tracked via server, and visualized at client side.

Navigation functions. The navigation state contains a set of variables that affect movement. Virtual or physical buttons can be mapped to affect these variables to create motion. A set of functions for automatic viewpoint movement are provided to smoothly interpolate both position and orientation between given points. Navigation can be restricted to follow the streets or a route.

GPS. GPS devices are supported via BlueTooth. The current position is available as a navigation variable. For example, the viewpoint can be attached to it, or a marker can be assigned to point towards it.

Figure 3: Lightweight widgets: setting up a route.

5. FIELD EXPERIMENT AND FEEDBACK

The first version of the m-LOMA engine and the city model, installed on a Dell X30 PDA, was put to a test in a field study to research users' orientation strategies [7]. The experiment was not intended to test platform robustness or model veridicality, but certain observations are justified. There were no software crashes during the 1-2 hour experiments. The frame rate was sufficient for conducting the tests, even though some subjects would have preferred a bit faster rendering. Subjects found it generally easy to recognize properly textured targets when the viewpoint was close to street level. When one subject had chosen a real façade as an orientation cue that was not textured in the model, he became severely disoriented. Similar problems were encountered if texture colors were unbalanced. On the other hand, ground level textures were usually incorrect due to occluders such as cars and people in digital photographs, but that did not seem to affect overall recognizability. It would appear that the subjects were able to adapt to consistent flaws in

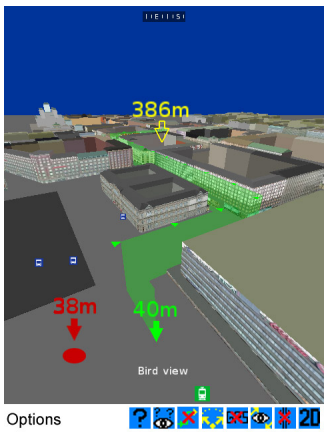

Figure 4: A 3D view presenting a route, markers, content icons and hotbuttons.

the model, but unexpected ones could easily disrupt their orientation strategies.

As the purpose of the first field experiment was to provide insight for developing a 3D interface, not to test one, it was performed using a very simple egocentric navigation with only the up vector constrained (*drive mode*, applying the flying saucer metaphor [4]). As anticipated, maneuvering was considered difficult, controls requiring attention frequently, causing heavy cognitive load. No visual aids such as markers or GPS were provided. If a target was lost from the view during maneuvering, it was difficult to relocate. Similarly, the point of origin was easily lost, frustrating the subjects.

Based on feedback, the set of features described earlier were developed, and most flaws in the 3D model corrected. In addition, the first real attempt at creating a 3D navigation interface was made. Several hotkeys and modes were developed. For example, a hotbutton provides a fast, automated viewpoint change between a bird view, top-down view and street level. To resolve cases where the user is disoriented, the *view landmarks* button causes the viewpoint to be animated to a position where both the user's current position (GPS position or latest viewpoint location) and closest landmark visible to the user are shown. *Tracks* can restrict the viewpoint along roads, keeping users from getting too close to buildings. The current street address can be constantly shown on the display. Analog 2D controls for stylus let the user move more freely, and more accurately. A field experiment will be conducted to evaluate this new interface.

6. CONCLUSIONS

An efficient mobile 3D city map engine has been developed, providing several features for developing navigation interfaces. The platform demonstrates that a realistic mobile 3D map can be implemented on current mobile devices without hardware support. When such hardware becomes commonplace (supporting the *Common Profile* of the OpenGL ES used in m-LOMA), a major increase in rendering speed is to be expected. The current system will provide scalability, memory management and features for navigation and 3D map development along with good performance in contrast to static scene graph renderers such as the JSR-184.

The platform will be used and modified to conduct a series of field studies, each with a different focus, such as the effect of local cues and model veridicality to recognizability and navigation. Traditional usability studies will be performed on the current navigation system. In near future, we will attempt to further modularize the system and release it for academic use.

7. ACKNOWLEDGMENTS

This work was supported by Interreg IIIA. Thanks to lead programmers Ville Helin and Nikolaj Tatti, and to Hybrid Graphics for the OpenGL ES implementation.

8. REFERENCES

[1] J. M. Airey. *Increasing Update Rates in the Building Walkthrough System with Automatic Model-Space Subdivision and Potentially Visible Set Calculations.* PhD thesis, UNC Chapel Hill, 1990.

[2] M. Bessa, A. Coelho, and A. Chalmers. Alternate feature location for rapid navigation using a 3d map on a mobile device. In *MUM '04: Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, pages 5–9, New York, NY, USA, 2004. ACM Press.

[3] S. Burigat and L. Chittaro. Location-aware visualization of vrml models in gps-based mobile guides. In *Web3D '05: Proceedings of the tenth international conference on 3D Web technology*, pages 57–64, New York, NY, USA, 2005. ACM Press.

[4] S. Fuhrmann and A. M. MacEachren. Navigation in desktop geovirtual environments: Usability assessment. In *Proceedings of the 20th ICA/ACI International Cartographic Conference*, pages 2444–2453, 2001.

[5] S. Hughes and M. Lewis. Attentive camera navigation in virtual environments. In *IEEE International Conference on Systems, Man & Cybernetics*, 2000.

[6] A. Nurminen. m-loma - a mobile 3d city map. In *Web3D '06: Proceedings of the 11th international conference on 3D Web technology*, pages 7–18. ACM, 2006.

[7] A. Oulasvirta, A.-M. Nivala, V. Tikka, L. Liikkanen, and A. Nurminen. Understanding users' strategies with mobile maps. In *Mobile Maps 2005 - Interactivity and Usability of Map-based Mobile Services, a workshop.* Mobile HCI, 2005.

[8] M. Przybilski, S. Campadello, and T. Saridakis. Mobile, on demand access of service-annotated 3d maps. In *Proceedings of the 23rd IASTED International Conference on SOFTWARE ENGINEERING*, pages 448–452. IASTED, 2005.

[9] I. Rakkolainen, J. Timmerheid, and T. Vainio. A 3d city info for mobile users. *Computers and Graphics*, 25(4):619–625, 2001.

[10] T. Vainio, O. Kotala, I. Rakkolainen, and H. Kupila. Towards scalable user interfaces in 3d city information systems. In *Mobile HCI '02: Proceedings of the 4th International Symposium on Mobile Human-Computer Interaction*, pages 354–358, London, UK, 2002. Mobile HCI, Springer-Verlag.