VET3D: A Tool for Execution Trace Web 3D Visualization

Craig Anslow, Stuart Marshall, and James Noble
Victoria University of Wellington, New Zealand
{craig, stuart, kjx}@mcs.vuw.ac.nz

Robert Biddle
Carleton University, Ottawa, Canada
robert_biddle@carleton.ca

Abstract

We are interested in finding new ways to visualize our software execution traces. An issue in visualizing our execution traces is deploying and integrating them into users' environments. We have a tool called VET3D that transforms execution traces into visualizations over the web. Our tool will help developers to understand the structure and behaviour of software.

Categories and Subject Descriptors D.2.6 [*Programming Environments*]: Graphical Environments

General Terms Design

Keywords Software Visualization, Execution Traces, XSLT, X3D

1. Introduction

Software visualization is the use of the crafts of typography, graphic design, animation, and cinematography with modern human-computer interaction and computer graphics technology to facilitate both the human understanding and effective use of computer software [7]. However, we are more focused on program visualization which is the visualization of actual program code or data structures in either static or dynamic form.

The potential use of 3D graphics for program visualization is significant and mostly unexplored [7]. The use of 3D graphics is not to enhance the beauty of a program visualization; instead it provides additional and fundamental information. A great deal of experimentation is needed to better understand the strengths and weaknesses of using interactive 3D graphics for software visualization.

We are interested in visualizing the structure and behaviour of software over the web in 3D. We currently have a Visualization Architecture for REuse (VARE) [6] for deploying software components over the web. VARE is a client-server architecture for supporting the visualization of software in a distributed environment. The design supports components in multiple languages and configurations (e.g. complete programs or just code fragments), and provides user control for all parts of the visualization process.

VARE requires tools to develop and deliver visualizations from execution traces that are easy to learn and are intuitive to understand for users. Our XML execution traces contain static and dynamic information of software components including the events that happened during the execution of a component. We have previously created a tool that could produce 2D Scalable Vector Graphics (SVG) [2] visualizations from our execution traces. We now describe a tool for creating 3D visualizations from our execution traces using X3D [8] – the Web3D Consortium's X3D open standard for web based 3D graphics.

OOPSLA'06 October 22–26, 2006, Portland, Oregon, USA.
ACM 1-59593-491-X/06/0010.

2. VET3D

We are developing a web-based software visualization tool called VET3D (Visualizing Execution Traces in 3D) that transforms XML execution traces into X3D visualizations. Figure 1 shows the overall architecture of VET3D. Users make queries from a web browser. Users can test drive[1] remotely executing software components to produce an XML execution trace, transform XML execution traces using XSLT into X3D visualizations, and display X3D visualizations in a web browser.

Separating the test driving and creation of visualizations steps allows users to test drive components and then create visualizations in the future. Users can also view stored X3D visualizations without having to test drive remote software or transform XML execution traces. Previous work describes our XML execution traces for software visualization [5], test driving software components [4], and databases for storing and querying XML execution traces [1].

From the architecture in Figure 1, VET3D currently implements transforming our XML execution traces into X3D visualizations so that users can interact with the visualizations in a web browser. VET3D has been integrated with our database tools but has yet to work in conjunction with our test driving tools. Some design features of VET3D include plug-in features to create new types of visualizations and capabilities to add to existing visualization types.

X3D [8] is an XML language for 3D content delivery on the web. X3D is the successor to the Virtual Reality Modeling Language (VRML97). X3D combines both geometry and runtime behaviour into a single XML file. X3D can be displayed in a native X3D browser, a web browser that has a X3D plug-in or transformed and delivered to a VRML97 viewer. The X3D runtime environment is the scene graph which is a directed, acyclic graph containing the objects represented as nodes and object relationships in the 3D world. X3D content can be created using authoring tools such as X3D Edit, text editors, or transformed using XSLT [3]. X3D allows scripting languages to be embedded such as ECMAScript and Javascript.

For displaying X3D visualizations VET3D uses the Octaga Player [2] plug-in which is compatible with Mozilla Firefox. The Apache Xalan XSLT processor is used for transforming the XML

[1] Test driving is defined as specifying a sequence of method invocation and field access/modifications and then executing the sequence on a software component.

[2] http://www.octaga.com

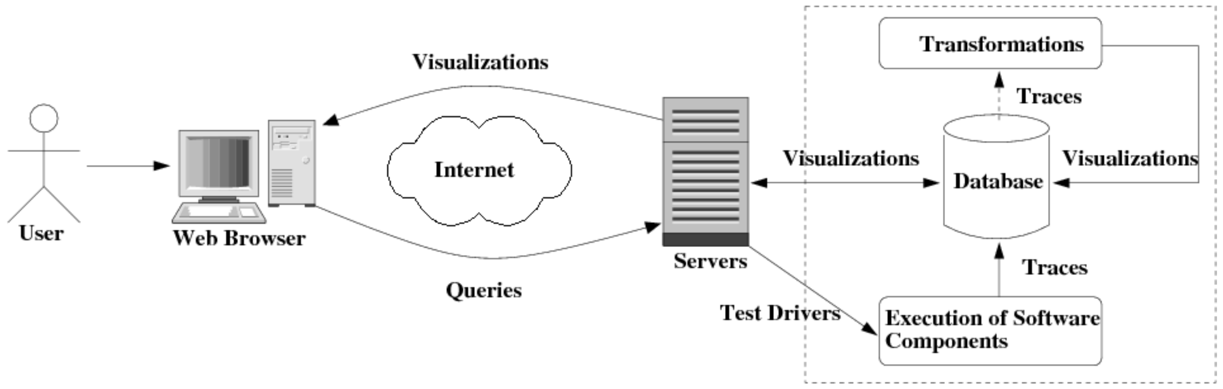
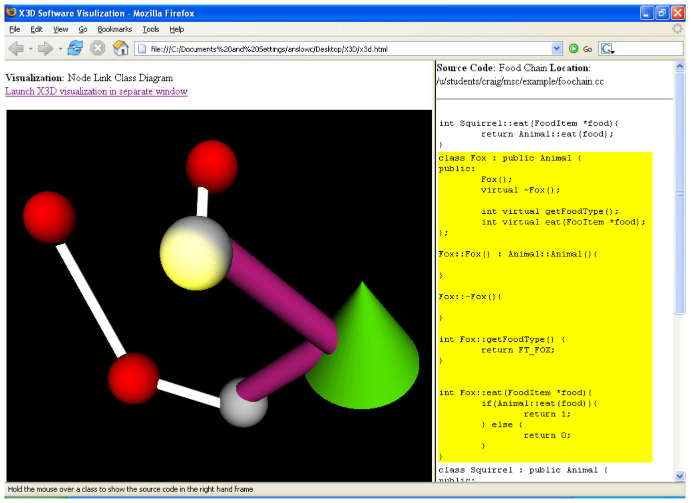


Figure 1. The VET3D architecture.

execution traces into X3D. Apache Tomcat is used to handle the Java Server Pages and Java Servlet web front end. Finally the eXist native XML database is used for storing and retrieving XML execution traces and X3D visualizations [1].

Figure 2 shows a node-link X3D visualization transformed from an execution trace of a C++ program. The visualization has two displays, the left display shows the X3D visualization and the right display shows the original source code of the program which was used to produce the visualization. The source codes is rendered in HTML. Classes are represented as spheres and the base class is represented as a cone. The two classes that inherit from the base class are coloured grey while the other classes are red. The cone is animated to change colours from blue to red to purple to green continuously every second. Larger purple links represent inherited relationships from the base class while smaller white links represent relationships amongst other classes.

When a user clicks on a class a light shines on the node in the visualization, and the associated class declaration in the source code is highlighted yellow in the right hand display. The user has selected the class in the middle of the visualization (grey sphere) which has highlighted the associated class declaration in the source code (the Fox class). The user can also rotate the visualization, zoom-in/zoom-out, pan from left to right and up and down, and move the classes around in the 3D world.


X3D Software Visulization - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
file:///C:/Documents%20and%20Settings/anslowc/Desktop/X3D/x3d.html
Go
Visualization: Node Link Class Diagram
Launch X3D visualization in separate window
Hold the mouse over a class to show the source code in the right hand frame

Source Code: Food Chain **Location:** /u/students/craig/msc/example/foochain.cc

```
int Squirrel::eat(FoodItem *food){
        return Animal::eat(food);
}
class Fox : public Animal {
public:
        Fox();
        virtual ~Fox();

        int virtual getFoodType();
        int virtual eat(FooItem *food);
};

Fox::Fox() : Animal::Animal(){

}

Fox::~Fox(){

}

int Fox::getFoodType() {
        return FT_FOX;
}


int Fox::eat(FoodItem *food){
        if(Animal::eat(food)){
                return 1;
        } else {
                return 0;
        }
}
class Squirrel : public Animal {
public:
```

Figure 2. X3D visualization created from an execution trace.

3. Summary and Future Work

VET3D is a prototype tool that transforms our XML execution trace into X3D visualizations. Users can select predefined visualization types to display information about a software component, such as node/link diagrams. New visualization types can be created by making new XML style-sheets for transforming execution traces and then plugging them into VET3D. Existing visualizations can be customized by adding new capabilities such as sliders to control the number of nodes displayed in a visualization.

We believe that the X3D visualizations our tool VET3D can produce will help assist developers to understand the structure and behaviour of software. However further extensive research on the performance, scalability, and usability of X3D visualizations is required to determine if the technology is applicable for use in software visualization. We also plan to integrate VET3D with other test driving tools to provide an end to end software visualization system for users.

References

[1] C. Anslow, S. Marshall, R. Biddle, K. Jackson, and J. Noble. XML database support for program trace visualisation. In N. Churcher and C. Churcher, editors, *Information Visualisation 2004*, *CRPIT Vol 35*. Australian Computer Society, 2004.

[2] M. Duignan, R. Biddle, and E. Tempero. Evaluating scalable vector graphics for use in software visualisation. In T. Pattison and B. Thomas, editors, *Information Visualisation 2003*, *CRPIT Vol 24*. Australian Computer Society, 2004.

[3] V. Geroimenko and C. Chen, editors. *Visualizing Information Using SVG and X3D, XML-based technologies for the XML-based Web*. Springer-Verlag, 2005.

[4] S. Marshall, R. Biddle, and J. Noble. Using software visualisation to enhance online component markets. In N. Churcher and C. Churcher, editors, *Information Visualisation 2004*, *CRPIT Vol 35*. Australian Computer Society, 2004.

[5] S. Marshall, K. Jackson, C. Anslow, and R. Biddle. Aspects to visualising reusable components. In T. Pattison and B. Thomas, editors, *Information Visualisation 2003*, *CRPIT Vol 24*. Australian Computer Society, 2004.

[6] S. Marshall, K. Jackson, R. Biddle, M. McGavin, E. Tempero, and M. Duignan. Visualising reusable software over the web. In P. Eades and T. Pattison, editors, *Information Visualisation 2001*, *CRPIT Vol 9*. Australian Computer Society, 2001.

[7] J. Stasko, M. Brown, and B. Price. *Software Visualization*. MIT Press, 1997.

[8] Web3D-Consortium. *X3D Specification*. `http://www.web3d.org/x3d/specifications/`, 2005.