CP-X3D:
A CROSS-PLATFORM 3DVR E-COMMERCE ENVIRONMENT

Tzu-Min Chang
Dept. of Computer Science and Information Engineering,

Bing-Yu Chen
Dept. of Information Management, Graduate Inst. of Networking and Multimedia,

National Taiwan University
No. 1, Roosevelt Rd. Sec. 4, Taipei 106, Taiwan
minming@cmlab.csie.ntu.edu.tw *robin@ntu.edu.tw*

ABSTRACT

X3D (Extensible 3D) is a standard description language for describing 3D scenes on the web environment and can be used to establish 3DVR e-commerce systems, such as virtual shopping malls. However, due to the CPU and memory limitations of mobile devices, the 3DVR e-commerce systems described by using X3D scene language cannot be used on mobile devices. This limitation delays the growth of 3DVR e-commerce systems, since the population and influence of mobile devices are more powerful than PC environments. To make X3D to be really cross platform, we provide a cross-platform X3D (CP-X3D) environment in this paper, which including a modified X3D scene language suitable for mobile devices and a cross-platform CP-X3D engine for both of mobile devices and PC environments. Through the CP-X3D environment, the 3DVR e-commerce systems described in CP-X3D scene language or developed with CP-X3D engine can be used on both of PCs and mobile devices.

Keywords: X3D, mobile device, cross-platform, 3DVR e-commerce environment

INTRODUCTION

Thanks to the progress of mobile devices, powerful mobile devices become more and more wide-spreading. People can use their powerful mobile devices to obtain the information they need or just for entertainments no matter where they are. Mobile devices will play more important roles in the near future. Not only using mobile devices to get information but also using mobile devices to play a game for entertainments will become common behaviors. However, existing applications on mobile devices are almost 2D graphics because rendering 3D graphics on mobile devices is still considered a difficult task, although 3D graphics can provide fancier interface and show more information than 2D graphics. Mobile devices are indeed characterized by some serious limitations with respect to PCs such as limited CPU and memory, lack or limited

performance of graphics accelerators, and lack of powerful development and debugging environments. Nevertheless, the capabilities of mobile devices are now on the increase, and this makes the research of 3D graphics on mobile devices come earlier. Hence, besides to use the 3D capability of mobile devices for entertainments, it is also possible to provide a 3DVR (3D Virtual Reality) e-commerce (electronic commerce) environment on mobile devices.

On the other hand, X3D (Extensible 3D) [2006] applications are in a broad range of application areas such as engineering, scientific, and information visualization, multimedia presentations, entertainments, web pages, shared virtual worlds, and 3DVR e-commerce systems. Furthermore, X3D applications probably can express more information than general 2D applications. Although there are so many types of browsers on computers, the utility of X3D is restricted by the demand people must have one computer beside them if they want to use any applications with X3D. That is because X3D applications are almost for PCs. In other words, X3D files can not be played anywhere and anytime because these X3D browsers of nowadays are made almost for PCs. However, there are many situations that people need to use X3D applications but they do not have any PC at that moment. As electronic maps based on X3D, people more often need these applications when they are in the way to a strange place than at home or office. Here is another example. Try to imagine a situation, somebody has a powerful mobile device and he or she wants to spend some money on a virtual shopping mall. This person is interested in two malls whose content are the same, and their prices are also identical. However, among these two virtual shopping malls, one is only for PCs, but the other can be played on both of PCs and mobile devices. If this person chooses the later virtual shopping mall, he or she can enjoy this virtual shopping mall even though he or she is waiting for a bus. In the same way, if X3D files or applications can be played both on PCs and mobile devices, they will get higher value than before.

To make mobile devices be able to play X3D files and applications not only raises the visibility of X3D but also make these applications be able to use in more appropriate situations. Hence, we provide a cross-platform X3D (CP-X3D) environment in this paper, which including a modified X3D scene language suitable for both of mobile devices and PCs and a cross-platform CP-X3D engine also for both of mobile devices and PC environments.

In the rest of this paper, we first introduce X3D briefly. Then, the CP-X3D engine and CP-X3D scene language are described. Finally, some results, conclusion, and future work are discussed in the final sections.

EXTENSIBLE 3D

Extensible 3D (X3D) is an open standard file format for 3D content delivery and run-time architecture which can represent and communicate 3D scenes and objects. It is an ISO (International Organization for Standardization) ratified standard that provides the storage, retrieval and playback of real time graphics content embedded in applications, within an open architecture to support a wide array of domains and user scenarios. X3D is not a programming API (Application Program Interface) and is not just a file format for geometry interchange either. X3D is intended for use on a variety of hardware devices and in a broad range of application areas. X3D allows components to be added to extend functionality for vertical market applications and services. Furthermore, X3D is the successor to the Virtual Reality Modeling Language (VRML) [2003] which is the original ISO standard for web-based 3D graphics.

The architecture of X3D provides increased functionality for immersive environments and enhanced interactivity. Moreover, the architecture of X3D also provides focused data interchange formats for vertical market applications within a small downloadable footprint. A component-based architecture supports creation of different "profiles", and components can be individually extended or modified through adding new "levels". However, the original X3D is designed for PC environments, since it depends on powerful graphics hardware which is only provided for PCs or above hardware. To provide CP-X3D, a subset of X3D should be defined suitable for both of PCs and mobile devices.

ARCHITECTURE OF CP-X3D ENGINE

Before describing the CP-X3D scene language, we have to first give an overview of the CP-X3D engine. The CP-X3D engine can be divided into two parts. One is the original X3D engine running on PC environments which can read the show the full version of the X3D nodes. The other is the X3D engine running on mobile devices, which is called X3D ES engine. The architecture of the X3D ES engine is similar with that of X3D engine, so the two engines can be treated as only one CP-X3D engine.

Figure 1 shows the architecture of the CP-X3D engine. The left side is the original X3D engine. For the original X3D architecture, when an X3D browser received X3D files or streams, the X3D files or streams will be parsed and displayed by the X3D engine. For the CP-X3D architecture, the behavior is similar, but the X3D browser will be a common CP-X3D browser, which can be used on PC and mobile devices. When the CP-X3D browser works on PCs, it will use the X3D engine. When it works on mobile

devices, it will use the X3D ES engine. If the CP-X3D browser received some nodes which are not defined in the CP-X3D scene language, the nodes will be ignored when the X3D file or stream are shown on mobile device, but they will be shown on the PC version.

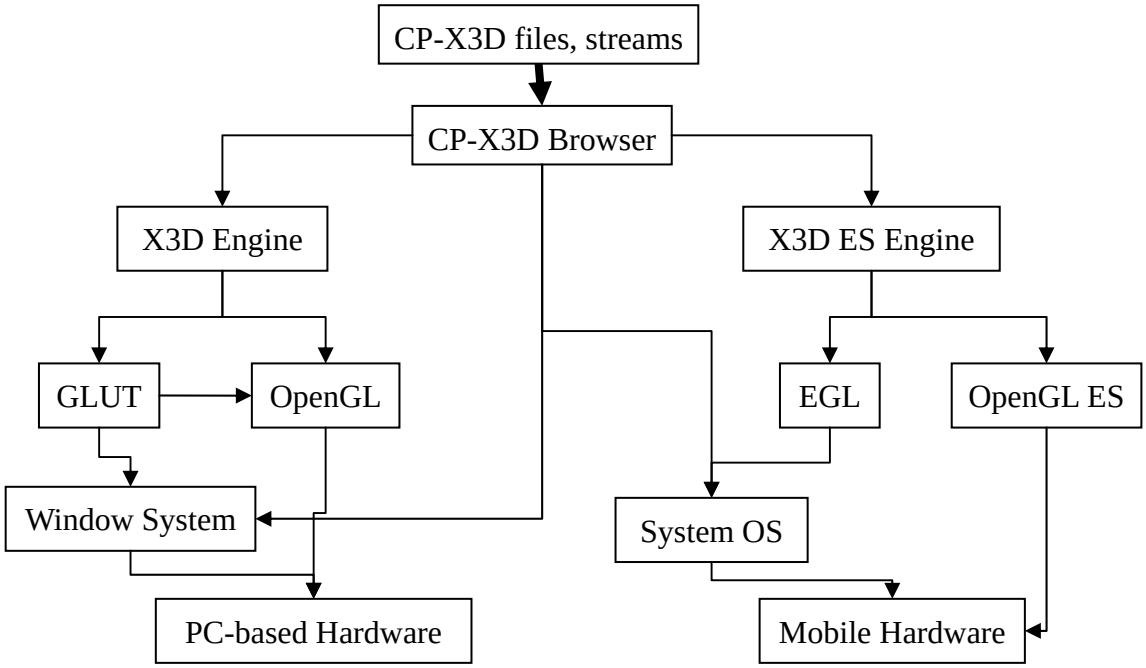

Figure 1. The architecture of CP-X3D engine.

The X3D ES architecture is similar with the original X3D architecture. The X3D engine is built by using GLUT and OpenGL and the X3D ES engine is built by EGL and OpenGL ES. OpenGL ES [2006] is a low-level and lightweight API for advanced embedded graphics using well-defined subset profiles of OpenGL. It provides a low-level API between software applications and hardware or software graphics engines. As for the reason to choose OpenGL and OpenGL ES as the rendering engines of the X3D and X3D ES engines, that is because OpenGL and OpenGL ES has the property of cross platform. Moreover, OpenGL and OpenGL ES are industry standards and royalty free. Moreover, OpenGL and OpenGL ES are easy to use because they are well structured with an intuitive design and logical commands.

Figure 2 shows the flow diagram of the X3D ES engine which is the same as that of the X3D engine. The X3D and X3D ES engines can be separated into three major modules, *Parser*, *Renderer*, and *Event Handler* whose functionality is as follows.

- *Parser*: this module parses the X3D files or streams. It supports full version of the X3D nodes and can then parse the X3D core concepts. Its input is X3D files or

streams and its output is a set of X3D nodes in a tree structure which defines the 3D scene. It includes *x3dio*, *x3dxml*, and *x3ddataset* for parsing different data-type of input data streams.

When users want to play X3D files, *x3dio* starts to receive the X3D files one by one. Moreover, *x3dio* is an interface in order to handle any type of input and output, and it has an implementation called *filestream* to work with the files while *x3dxml* has an implementation called *xmlreader* used to read a stream which has processed by *x3dio* and convert the stream to a tree of the X3D nodes. Furthermore, *x3dxml* has a subordinate called *x3ddataset* which handles the nodes defined in the X3D specification and helps *x3dxml* to generate a tree of X3D nodes according to the X3D specification. After the parsing process, a set of X3D nodes in a tree structure has been generated. These X3D nodes are sent to the second module, *Renderer*. Since the CP-X3D scene language is actually a subset of the original X3D scene language, the *Parser* of X3D ES engine is the same as the *Parser* of X3D engine.

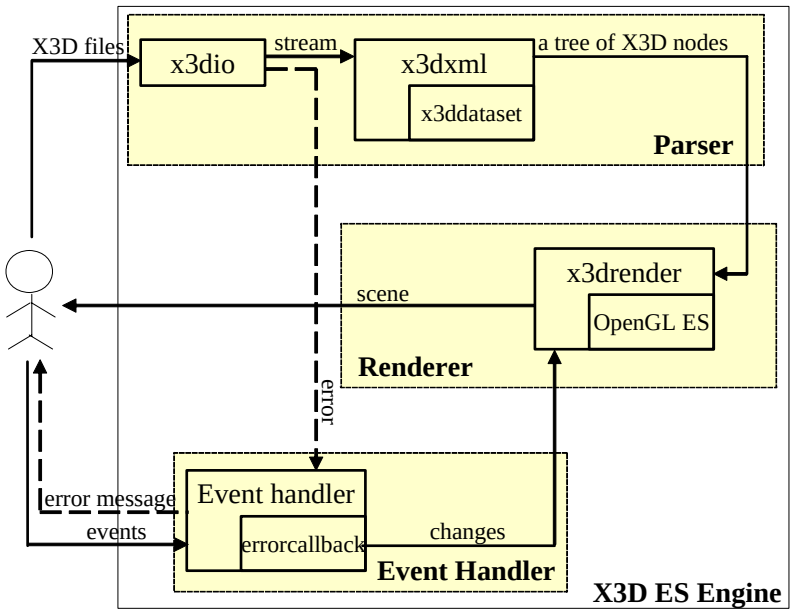

Figure 2. The flow diagram of X3D ES engine.

- *Renderer*: this module deals with all the process about rendering. The input is a tree of the X3D nodes generated by the *Parser* of the CP-X3D Engine. After the rendering processing, the *Renderer* outputs the 3D scene defined by the X3D nodes. Moreover, the *Renderer* also receives a set of changes in the attributes of X3D nodes such as the movement of the viewport and models. The main sub-module of the *Render* is called *x3drender*, which parse the X3D scene tree and call the

OpenGL or OpenGL ES engine to render the 3D scene. Since the rendering performance is not the same for mobile devices and PCs, although the X3D ES and X3D engines have the same *Parser*, their *Renders* are different.

- *Event Handler*: this module manages all kinds of events. The input of *Event Handler* is a set of events and the output is a set of changes in the attributes of X3D nodes that define the scene. The events include the interaction of users such as to move the viewport, to make some objects to rotate, etc. When some exceptions happen such as out of memory, *x3dio* which has an implementation called *filestream* to work with files, will send an error notification to the *Event Handler*. The *Errorcallback*, a sub-module of *Event Handler*, will gather all of the exceptions and generate a message to notify the users. The *Event Handler* in the X3D ES engine is also the same as that in the X3D engine.

CP-X3D SCENE LANGUAGE

According to the X3D specification [2006], each X3D component may designate a level of service by using a numbering scheme in which higher-numbered levels denote increasing qualities of service (QoS). In other words, X3D specification may be supported at varying levels or QoS. Table 1 lists the level supported by the *Parser* and *Renderer* of X3D and X3D ES engines. The first column is the components of X3D specification in alphabetical order. Gray background is to give clear indication of full support. For the module has a level of support set to zero means the specific feature may not need to be supported by that module. For example, the *Renderer* never interpolates anything, and does not need to support *Networking*, *KeyDeviceSensor*, *PointingDeviceSensor*, *Time*, and *Sound* components according to the X3D specification.

Obviously, the *Parser* of the X3D ES engine fully supports all of the X3D components. With this *Parser* for mobile devices, the system can provide the ability to parse every component and node defined in the X3D specification. Hence, even the user uses the X3D ES engine to read an X3D file; the X3D file can also be pared normally, although some nodes may not be shown in the screen. As for the *Renderer*, there are ten components full support in the X3D ES engine, which are *Core*, *Geometry2D*, *Geometry3D*, *Geospatial*, *Grouping*, *Lighting*, *NURBS*, *Rendering*, *Shape*, and *Texturing*. Hence, the *Renderer* can organize X3D nodes into groups to establish a transformation hierarchy for the X3D scene graph. The *Renderer* also handles fundamental rendering primitives such as *TriangleSets* and *PointSets*, and geometric

properties nodes that define the coordinate indices, colors, normals and texture coordinates. The CP-X3D nodes are listed in エラー! 参照元が見つかりません。.

	Maximum	X3D ES Engine		X3D Engine	
		Parser	Renderer	Parser	Renderer
Core	2	2	2	2	2
DIS	1	1	0	1	0
EnvironmentalEffects	3	3	3	3	3
EnvironmentalSensor	2	2	0	2	0
EventUtilities	1	1	0	1	0
Geometry2D	2	2	2	2	2
Geometry3D	4	4	4	4	4
Geospatial	1	1	0	1	0
Grouping	3	3	3	3	3
H-Anim	1	1	0	1	0
Interpolation	3	3	0	3	0
KeyDeviceSensor	2	2	0	2	0
Lighting	3	3	3	3	3
Navigation	2	2	0	2	2
Networking	3	3	0	3	0
NURBS	4	4	4	4	4
PointingDeviceSensor	1	1	0	1	0
Rendering	4	4	3	4	3
Scripting	1	1	0	1	0
Shape	3	3	3	3	3
Sound	1	1	0	1	0
Text	1	1	0	1	1
Texturing	3	3	1	3	1
Time	2	2	0	2	0

Table 1. The comparisons of support levels between X3D and X3D ES engines

On the other hand, the X3D engine for the PC environment can play the X3D files which have more complex scenes because the X3D engine is based on OpenGL 2.1 [Segal & Akeley 2006] and designed for running on the PC environments. Some limitations of X3D ES engine are caused by hardware and OpenGL ES 1.1 [2006] while some limitations are due to these components may be redundant and decrease the performance of X3D ES engine. For example, the X3D ES engine does not fully

support *Navigation* and *Text* components. That is because these two components are frustrating on mobile devices. Besides, the X3D ES engine does not support *H-Anim* and *DIS* components. That is because these two components require too large computation power which is hard to be realized on mobile devices. Hence, the nodes supported by the X3D ES engine can be treated as CP-X3D nodes, since the nodes can be supported both on mobile devices and PC environments. For those X3D nodes which are not CP-X3D nodes, although the X3D ES engine can also parse the nodes, but it can not render them.

RESULT

Figure 3 shows a simple result of the X3D ES engine which shows several spheres with their wireframe, flat shading, and smooth shading. Figure 4 shows the comparisons of X3D and X3D ES engines which use the same X3D file as shown in Figure 3. Since this X3D file is written by only using the CP-X3D nodes, the results on PCs and mobile devices are the same. Figure 5 shows four complicated results, which show an ocean scene, a grasslands scene, an ocean scene at sunrise, and a simple city map. The users can move the viewport in these static scenes.

Figure 3. The results of three classic shading and lighting algorithms on a Pocket PC 2003 emulator.

Besides, the X3D ES engine is able to show some special effects such as fog. In X3D specification, the *Fog* node provides a way to simulate atmospheric effects by blending objects with the color specified by the *color* field based on the distances of the various objects from the viewer. Figure 6 shows a scene with the fog effects in different viewports. In Figure 7, there is a scene with several simple buildings. This scene has around 5000 triangles. The performance of this complicated result is 5fps (frames per second) on a real mobile device with a 520MHz processor and 64MB main memory. Of-course, the performance on a desktop or laptop PC is real-time.

Component	Node	Component	node
Core	MetadataDouble MetadataFloat MetadataInteger MetadataSet MetadataString	Lighting	DirectionalLight PointLight SpotLight
EnvironmentalEffects	Background Fog TectureBackground	Navigation	Billboard Collision ViewPoint
Geometry2D	Arc2D ArcClose2D Circle2D Disk2D Polyline2D Polypoint2D Rectangle2D TriangleSet2D	Geometry3D	Box Cone Cylinder ElevationGrid Extrusion IndexedFaceSet Sphere
Networking	Anchor Inline LoadSensor	Rendering	Coordinate IndexedLineSet LineSet PointSet Normal
Grouping	Group StaticGroup Switch Transform WorldInfo	Shape	Appearance FillProperties LineProperties Material Shape
KeyDeviceSensor	KeySensor	Texturing	ImageTexture TextureCoordinate

Table 2. CP-X3D nodes

Figure 8 shows the comparison between the results of X3D and X3D ES engines. The tree structure which defines the 3D scene of this X3D file has already used a lot of memory. Because of limited memory, the X3D ES engine can not play the X3D files which are as complex as the result of the X3D engine. However, the user using the

mobile device can still use the X3D file and see the partial result. Although the result is partial, the information may be enough to fulfill the user's needs.

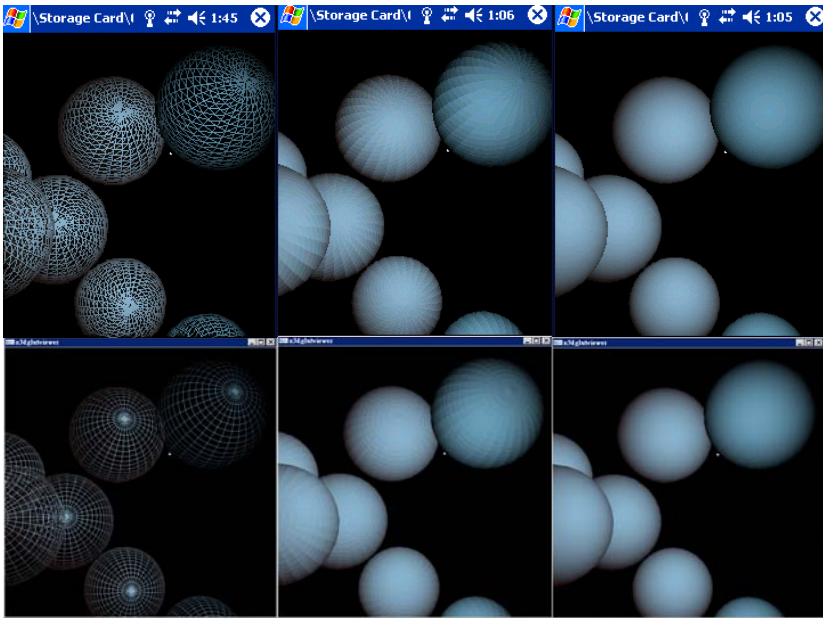

Figure 4. The comparisons between the X3D (lower) and X3D ES (upper) engines.

Figure 5. X3D scenes played with X3D ES engine on a Pocket PC 2003 emulator.

CONCLUSION AND FUTURE WORK

Because of the CP-X3D engine, it is feasible explicitly to develop cross platform X3D applications on both of mobile devices and PCs. The CP-X3D engine proves X3D is able to be implemented on both of mobile devices and PCs without any replacement though some X3D components may be redundant for some X3D applications. Moreover, the CP-X3D engine brings great contribution not only to X3D development but also to the applications on mobile devices. The X3D ES engine provides a full support parser which can parse X3D files and easy to make a browser be able to play X3D files, and it

also provides a partial support renderer for basic rendering. In other words, the X3D ES engine makes other applications on mobile devices support X3D easily. Thus, the X3D ES engine or the CP-X3D engine makes the cross platform 3D games and virtual shopping malls much simpler to develop than before.

Figure 6. The results of X3D scene with fog in different viewports on a Pocket PC 2003 emulator.

Figure 7. A simple 3D city map played on a Pocket PC 2003 emulator.

Although the X3D ES engine has made X3D applications on mobile devices easy to develop, the X3D ES engine still has its own limitation caused by mobile devices and OpenGL ES 1.1. For example, the X3D ES engine can not play several complex models at the same time because of the limited memory and CPU of mobile devices. This limitation makes the 3D scenes have to be as simple as possible. In fact, some limitations of the X3D ES engine are caused by the limitations of the hardware. Although the performance of the X3D ES engine is really restricted by these hardware limitations, the X3D ES engine can also get improvement with the progress on the mobile devices.

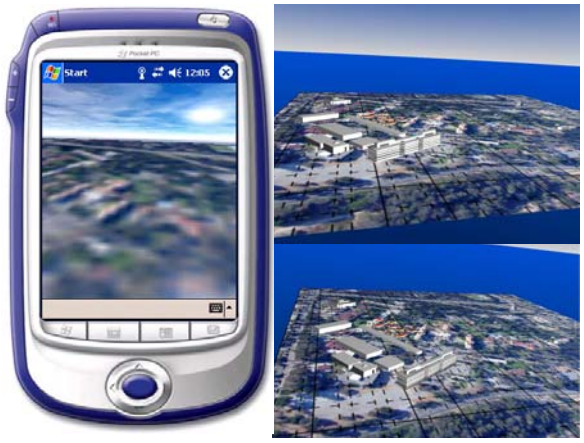

Figure 8. The comparison between the results of the X3D (right) and X3D ES (left) engines.

ACKNOWLEDGMENT

This work is partially supported by the National Science Council of Taiwan under the numbers: NSC93-2622-E-002-033 and NSC94-2622-E-002-024. The authors would like to thank Nein-Hsien Lin for his great support on X3D engine implementation.

REFERENCE

Extensible 3D (X3D) -- Part 1: Architecture and base components, ISO/IEC 19775-1:2004. 2006. Web3D Consortium, Inc.

Lin, N.-H., Huang, T.-H., & Chen, B.-Y. 2007. 3D model streaming based on a jpeg 2000 image, In *Proceedings of IEEE 2007 International Conference on Consumer Electronics*.

OpenGL® ES Common/Common-Lite Profile Specification Version 1.1.07. 2006. The Khronos Group Inc.

Pouderoux, J., & Marvie, J.-E. 2005. *GLUT|ES - The OpenGL|ES Utility Toolkit* http://glutes.sourceforge.net/.

Segal, M., & Akeley, K. 2003. *The OpenGL® Graphics System: A Specification (Version 1.5)*. Silicon Graphics, Inc.

Segal, M., & Akeley, K. 2006. *The OpenGL® Graphics System: A Specification (Version 2.1)*. Silicon Graphics, Inc.

The Virtual Reality Modeling Language -- Part 1: Functional specification and UTF-8 encoding, ISO/IEC 14772-1:1997. 2003. Web3D Consortium, Inc.

Will, H.-M. 2004. *Vincent Mobile 3D Rendering Library*. http://sourceforge.net/projects/ogl-es/.