AN OPEN-SOURCE C SHARP LIBRARY BASED ON OPENGL FOR STEREOSCOPIC GRAPHIC APPLICATIONS DEVELOPMENT

Santiago Martín[1], Liudmila Pupo[2], Yoander Cabrera[2] and Ramón Rubio[1]
[1]*Department of Manufacturing Engineering and Construction, University of Oviedo, Gijón, Spain*
[2]*Department of Visualization and Virtual Reality, University of Informatics Sciences, Havana, Cuba*

Keywords: Computer graphics, Virtual reality, Stereoscopic applications.

Abstract: The Graphics Library Stereo Vision engine (GLSVe) is a freely available open-source C♯ library based on OpenGL. It has been designed to facilitate the creation, mainly by researchers or students, of graphic and virtual reality prototypes incorporating stereoscopic representation. Its design will allow a stereoscopic graphic application in an easy way and without previous theory knowledge. This is demonstrated to be a good training for student's motivation in order to learn the theoretical bases by means of experimentation. The observer, the 3D pointer, the screen, the 3D sound and the graphics primitives are managed through different classes. This allows easy implementation of virtual reality scenarios if a tracking system is available (including multi-screen environments). Graphic primitives could have different appearance as they are seen by each observer eye, allowing the development of software for optometry research. Different stereoscopic modes have been implemented: side by side, cross eye, anaglyph, interlaced, alternated pages and dual stream. The article describes the GLSVe architecture and main capabilities, as well as different application scenarios (virtual reality environments; ophthalmology research; and visualization and compilation of geological photo pairs). The GLSVe is distributed under the terms of the GNU Library General Public License agreement.

1 INTRODUCTION

Learning methods have evolved from master classes to practical training due to industrial competitive requirements (Clausen, 2005). Software packages and virtual laboratories are especially useful in this new way to learn. Nowadays it is common to include them in teaching classes, especially in engineering degrees, where the engineer expected to have intuitive knowledge of complex and very different subjects(Palanichamy et al., 2004; Patryn and Pietruszko, 2005). Stereoscopic visualization and virtual reality systems are interesting subjects to include in an engineering degree (syllabus), in the senior year course, as this topic uses previous and important mathematic and geometric concepts. Historically, the first step in the development of computer vision theory has been the understanding of human stereoscopic vision. The difficulty of this subject makes advisable a suitable computer tool. A close example of this is a package oriented to obtain disparity maps and three-dimensional models based on stereo images (Castejón et al., 2009).

On the other hand, scientists need software packages as an easy development prototypes tool to validate their research. Regrettably, scientific software is too often a closed source, leaving the user with a black box performing magical operations. The open-source revolution that has taken place in the world of software development, perceived as being counterproductive for the overall scientific progress, faces these limitations (Steven, 1999). Stereoscopic visualization and virtual reality systems are subjects of research by computer graphics scientists. Their results are applied in different areas, like medical image, surgical planning and training (Kockro et al., 2007), ophthalmology research (Waddingham et al., 2006), psychology, scientific visualization, industrial design, chemistry education (Limniou et al., 2008), geology or cartography (Billen et al., 2008) . An open source package oriented to easily develop stereoscopic graphic applications will be very well received by researchers of these areas.

The creation of graphic prototypes, incorporating stereoscopic representation, is possible using an API (Application Programming Interface) as aid, like OpenGL or DirectX. However, any programmer who wishes to develop these contents has to fully un-

Martín S., Pupo L., Cabrera Y. and Rubio R..
AN OPEN-SOURCE C SHARP LIBRARY BASED ON OPENGL FOR STEREOSCOPIC GRAPHIC APPLICATIONS DEVELOPMENT.
DOI: 10.5220/0003646902050210
In *Proceedings of 1st International Conference on Simulation and Modeling Methodologies, Technologies and Applications* (SIMULTECH-2011), pages 205-210
ISBN: 978-989-8425-78-2

derstand the rules governing stereoscopy which, besides entailing a considerable additional effort, is a possible source of errors. Some graphics card and display device manufacturers have developed drivers with stereo functions. These drivers generate two or even more cameras from a single scene. The result is a stereogram that can be displayed in devices that support the driver in question (shutter glasses, auto-stereo monitors, etc.). Obviously, offering these drivers with stereo functions represents a first competitive advantage for hardware manufacturers.

The problem is that these drivers are not multiplatform and the programmer does not have total control of parameters associated with the stereo nature of the scene. From this we can deduce the advantage of defining a library that eliminates the need for the programmer to make any calculations requiring mastery of the theory of stereo vision.

With this in mind, a first library was proposed by our research team, named GLSV library (Graphics Library Stereo Vision) (Martín et al., 2009). It was a set of auxiliary functions similar to the ones used in OpenGL: gluLookAt was substituted by glsvLookAt; gluPerspective by glsvPerspective; and so on. Thus the use of GLSV does not require changing the way a programmer develops applications development with OpenGL.

Despite its interest, GLSV is not an object-oriented library but a set of functions. It is not suitable for beginner programmers, although expertise computer graphics programmers would appreciate it. The creation of graphic prototypes incorporating stereoscopic representation is feasible with GLSV and easier than with OpenGL, but it is not at all an easy or quick task. GLSV does not seem to be the solution a medical researcher is looking for. It is not what students would like to use in an intensive course about virtual reality systems.

Therefore, in this paper a new Graphics Library Stereo Vision engine (GLSVe) is proposed. GLSVe has been written in C♯ and based on OpenGL, and it is distributed under the terms of the GNU Library General Public License agreement.

The rest of this paper is organized as follows. Section 2 describes functional and non functional requirements of the GLSVe library; Section 3 presents a description of the libraryt's structure. The result of using the GLSVe library in different applications is presented in Section 4. Finally, Section 5 is devoted to concluding remarks.

2 FUNCTIONAL AND NON FUNCTIONAL REQUIREMENTS

2.1 Functional Requirements

First of all, the observer, the 3D pointer, the screen, the 3D audio and the graphics primitives are managed through different classes. This functional requirement allows implementation of virtual reality scenarios if a tracking system is available. Viewport transformations are also implemented on its own class. This allows having under control the conversion from pixel to user unit.

Graphic primitives can have different appearance as seen by each observer's eye. This requirement allows the development of software for optometry research and it can be also used when a stereoscopic photo pair is included in the virtual scene. Related with this, once the library is compiled, the programmer can still define its own new graphic primitives.

Monoscopic mode and several stereoscopic modes have been implemented: side by side, cross eye, anaglyph (red-cyan, red-blue, red-green), interlaced, alternated pages and dual stream (for Quadro graphic cards) (Lipton, 1997).

Finally, as GLSVe evolves from GLSV, all the set of GLSV functions are still accessible to the programmer.

2.2 Non-functional Requirements

GLSVe has been written in C♯ because this programming language has been designed to be easily learned and used by beginner programmers. C♯ was developed by Microsoft within the .NET initiative. Even though, other considered simple and general-purpose programming languages, like Visual Basic, an object-oriented language. Although C♯ applications are intended to be economical with regard to memory and processing power requirements, the language was not intended to compete directly on performance and size with C. For example, managed memory cannot be explicitly freed; instead, it is automatically de-allocated. Moreover memory address pointers can only be used within blocks specifically marked as unsafe. The reference C♯ compiler is Microsoft Visual C♯ but there are also other compilers, including open source ones.

GLSVe uses OpenGL as low level graphic programming library. OpenGL is the pervasive standard for the development of multiplatform applications. It was developed by Silicon Graphics Inc. (SGI) in 1992 and it is still the reference for graphic application developers. Generally, programmers use OpenGL in

combination with other supplementary APIs in order to generalize common functions. Thus, for example, the GLU (OpenGL Utility) library provides some functions to create complex objects and to easily manage the camera.

Currently there is no managed code library widely known for working with OpenGL. For that reason, it is necessary to use a wrapper for OpenGL or directly import the relevant methods into the assembly. The solution selected in this project has been the Tao Framework. It implements a OpenGL wrapper allowing to use it in the C♯ language. It has only been necessary a small modification to support dual stream stereo mode for Quadro graphic cards. Tao Framework also gives GLSVe access to SDL (Simple DirectMedia Layer) and OpenAL (Open Audio Library). Non-functional requirements are graphically summarized in Figure 1.

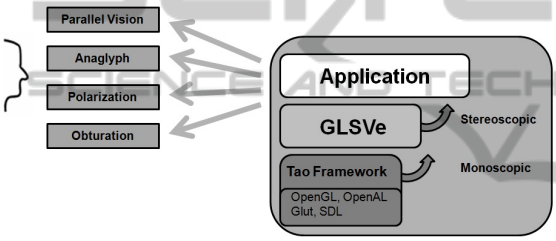

Figure 1: GLSVe context.

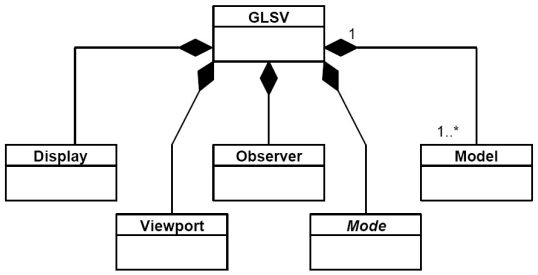

Figure 2: GLSVe core classes.

ing devices work in pixel, it is necessary to implement a transformation between the Display's dimensions in the virtual space and its dimensions in pixels. Viewport class does that.

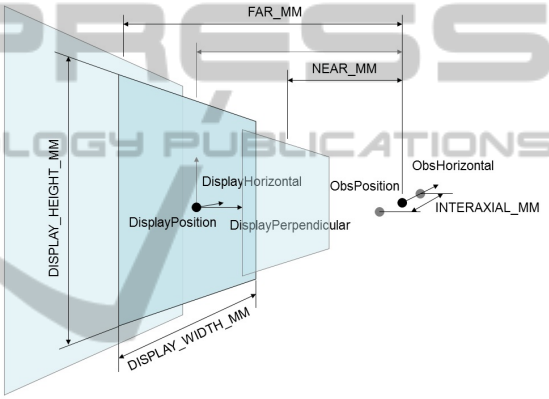

Figure 3: Observer and Display cases.

3 DESCRIPTION OF THE LIBRARY'S STRUCTURE

3.1 Core Classes

GLSVe core classes are represented in Figure 2. The main class is called like the old library, GLSV. This is because all the functions defined in the old library are still accessible through it. Internally, it declares objects of the rest of the classes: Observer, Display, Viewport, Mode, and a list of Model. It invokes the drawing of the scene using the currently active stereoscopic viewing mode.

The Observer class is defined by the 3D positions of its two eyes (Figure 3). Near and Far planes, which represent in OpenGL the top and the base of the pyramid frustum to be rendered, are also properties of the Observer class. That distances are measured from each eye to the perpendicular direction to the Display. The Display class is defined with 6 degrees of freedom and two dimensions. Both Observer and Display belong to a virtual space which dimensions can be made equal to the ones in the real space. As viewport class does that.

In a virtual reality prototype this would be enough. But in other applications would be desirable another transformations, like translation (pan), scaling (zoom) and parallax. These transformations are properties measured in pixels in the Viewport and in virtual units in the Display. We must take into account that these transformations break the virtual reality paradigm, as they introduce geometric distortions in the perceived 3D space. A translation (pan) entails moving the Display (or the Viewport) laterally, it differs from a lateral movement of the whole scene, but the observer would tolerate it. A scaling (zoom) transformation entails making the Display (or the Viewport) bigger. If we use the property Viewport zoom , we will be constrained by OpenGL limitations (see GL_MAX_VIEWPORT_DIMS). Zoom is not the same as making the scene bigger, but the observer would tolerate it also. Finally, a parallax transformation entails moving the left and right projections in opposite directions. It is not the same as getting closer to the scene, but again the observer would tolerate it. Figure 4 explains these transformations in the

Viewport implementation.

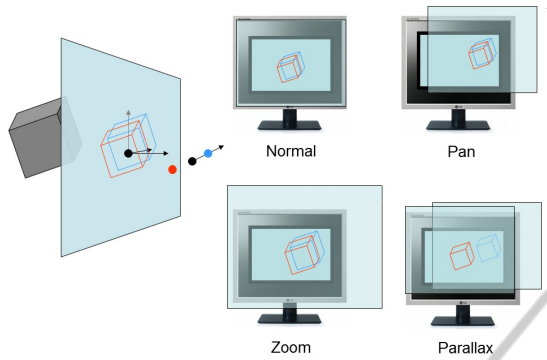

Figure 4: Viewport pan, zoom and parallax transformations.

The available visualization modes are defined in an abstract class, named Mode. We have a monoscopic visualization mode and several stereoscopic modes: SideBySideMode, CrossEyeMode, AnaglyphMode, InterlacedMode, MonoChannelStereoMode (alternated pages) and QuadroStereoMode (dual stream).

Finally, we have the graphics primitive. Each graphic primitive is defined on its own class, which inherit attributes and behaviors from a parent class, named Model. Each graphic primitive has three different drawing methods: left eye, right eye and head. Moreover, they have also the properties VisibleLeftEye and VisibleRightEye. In this way, graphic primitives must have different appearance as they are seen by each observer eye. OpenGL's depth and blend tests can be switched through the main class GLSV. Once the blend test is activated, we can define graphics primitives with a different alpha value for each eye. This can be used in optometric software to study the visual perception of a patient.

Anyway, once the library is compiled, the programmer can still define its own new graphic primitives. Graphic primitives are organized in a Scene Graph with Nodes. Parts of the 3D scene can be rotated, translated and scaled thanks to the Nodes.

3.2 The 3D Pointer and Mouse Interaction

M_Pointer3D is a singular subclass of Model. It represents the user pointer in the 3D virtual space. It can have different appearances (Type property). Its left and right projections can have constant size on the screen if needed (ConstantSize property): as the M_Pointer3D goes farther away, its size will be increased to compensate the effect of the perspective.

Two new classes, MouseState and MouseEvents, allow managing the pan/zoom/parallax transformations with the mouse. They also manage the XY and Z (depth) movements of the M_Pointer3D. MouseState stores the mouse changes on its properties. The public MouseEvent function (of the class MouseEvents) receives an object MouseState and interprets it. M_Pointer3D projections will move to the center of the screen due to the perspective effect. If we want M_Pointer3D projections not to move up or down, MouseEvents will compensate these changes for us again. Finally, MouseEvents guarantees also the coherence between the windows coordinates of the mouse and the projected coordinates of the M_Pointer3D.

3.3 Audio3D Class

A class named Audio3D has been defined using OpenAL, a free software cross-platform audio API. It is designed for efficient rendering of multichannel three dimensional positional audio. The basic OpenAL objects are a Listener, a Source, and a Buffer. Each buffer can be attached to one or more Sources, which represent points in 3D space which are emitting audio. There is always one Listener object (per audio context), which represents the position where the sources are heard rendering is done from the perspective of the Listener.

Each Audio3D object can be played, stopped or paused; it has a volume, a file path and a 3D position. Moreover, it receives a copy of GLSV instance. Thus, the Audio3D instance knows where the observer (listener) at any moment.

3.4 File Input/Output

X3D is the ISO standard XML-based file format for representing 3D computer graphics, the successor to the Virtual Reality Modelling Language (VRML). It has been adopted as the main CAD format in this project. The structure of the graphic primitives tree, which uses the class M_Node : Model as said before, is a direct interpretation of the X3D file format.

Although, at present, this module is not completely functional, its development is not a difficult task thanks to the System.Xml class available in .Net C♯. As example, the M_Mesh : Model has a public function to read the data directly from a X3D file.

3.5 Interaction with Input Devices

The programmer can use other libraries in conjunction with GLSVe to face the development of more so-

phisticated software using VR devices. It is especially interesting to implement the tracking option to control the Observer properties and the Pointer3D position. Simple examples of use have been developed using the WiimoteLib (for the Nintendo Wiimote) and the Touchless SDK (which permits to track color marks with a web cam), both written in C♯.

4 APPLICATION RESULTS

The library is being used in different application scenarios, demonstrating its utility and versatility.

4.1 VISAGE: A Stereo Viewer and Compilation Software Tool for Geologic Works

The aim of this project is to define a suite of applications to solve the problem of seeing a stereo pair and manually recover a 3D model. It was originally oriented to geologic works, but it can be used in other applications, like building restitution, medical image, reverse engineering, etc (Martín et al., 2007).

VISAGE is written in .Net C♯ using the GLSVe and the TAO Framework. The user can see pairs of stereo photos (implemented as objects of PhotoPair : Model class) with total control of all the variables involved: intraocular distance, camera focal, camera size, screen size, observer position. Translation (pan), scaling (zoom) and parallax transformations have been implemented using the Display class. Vectorial entities can be drawn using the mouse as a 3D pointer thanks to MouseState and MouseEvents classes. The DXF format is used to export and import those vectorial entities to or from a CAD program. The reference system can be defined by the user using the terrain coordinates module. VISAGE it's able to work with different stereoscopic modes.

The stereo pair used in VISAGE has been previously idealized, it is supposed. This means: to solve the problems of intrinsic and extrinsic calibration problems of the stereo pair. Then a new pair of photos can be generated without geometric distortion and a common projection plane. Nowadays, this has been implemented in a different tool, called STEREO NORMALIZER, which uses the OpenCV library.

4.2 AMBLY: Medical Optometry

Treatment of amblyopia consists of forcing use of the amblyopic eye, usually by patching the good eye. It results in a better accuracy of the lazy eye after the treatment, but the stereopsis is not improved (Simons, 2005). Virtual Reality and Stereoscopic Vision technologies can be used to design computer games where each eye receives different signals of the virtual world (Waddingham et al., 2006; Achtman et al., 2008; Kozulin et al., 2009). This project consists in the development of computer activities to improve the vision of the child, not only the visual acuity of the lazy eye but also the stereoscopic vision.

Although the project is in its initial stages, there have been designed four different activities that should be clinically validated. The first activity measures the stereopsis. As in the classic TNO Test, a dot stereogram is showed. The hidden shape must be identified by the patient. The second activity trains the lazy eye. The patient must look for a hidden object in a flat scene. Although the lazy eye sees perfectly the object, the healthy one is penalized. This can achieved thanks to different drawing methods implemented in the Models for each eye. The third and fourth activities train the stereopsis ability. In the third one the patient must look for a hidden object in a 3D scene and in the fourth activity he/she must draw a 3D shape joining a set of 3D points. The mouse is used again, as a 3D pointer thanks to MouseState and MouseEvents classes.

The viewing format selected has been side by side. The conversion among real units, virtual units and pixels are well known thanks to the Viewport class. The results of visual acuity can be presented to the optometrist in seconds of arc.

4.3 Virtual Reality Room

The aim of this project is to build a multiwall virtual reality stereo system using low cost hardware and open-source libraries (CruzNeira et al., 1993; Johnson et al., 2006).

The tracking system is implemented using the Nintendo Wii Remote infrared cameras. Intrinsic and extrinsic calibration camera is implemented using Intel OpenCV image processing library. Emgu CV is used as .Net C♯ wrapper to the OpenCV library. In the basic implementation, two wiimote cameras are needed to track the Observer and the Pointer3D. The observer is marked with at least two IR leds, as GLSVe Observer class needs the positions of the two eyes. The pointer is marked with at least one led, as Pointer3D class needs only a 3D position.

In addition, a virtual device server has been developed. Different physical devices (data gloves, inertial tracking, gaming controllers, etc.) can be connected with the virtual server using XML messages. Finally, a GLSVe based application connects to the server, re-

ceiving the data of the virtual device as XML messages. This architecture makes easier the interaction with devices whose libraries are written in other languages. Moreover, the virtual server can be configured as a tracking system. Several computers, each one with its own screen and all running the same GLSVe based application, are connected to the virtual server, receiving the tracking data in a multiwall configuration.

5 CONCLUSIONS

We presented some details of a new open-source library designed to facilitate the creation of graphic and virtual reality prototypes incorporating stereoscopic representation. Different application scenarios have been described to demonstrate the utility of the library and to justify several aspects of its design. It is especially remarkable that graphic primitives can have different appearance as they are seen by each observer eye, allowing the development of software for optometry research.

The GLSVe is freely available under the terms of the GNU General Public License. Nowadays two Universities are collaborating on its development, but the project is open to other contributions. You can download GLSVe with several examples of use from the SourceForge website.

Besides its proven usability in research and production of quality scientific software, the library has also become a valuable tool for teaching virtual reality concepts. Until now the library has been successfully used in four university courses for computer engineering students.

ACKNOWLEDGEMENTS

This research has been possible thanks to the existence of the collaboration agreement between the University of Oviedo and the University of Informatics Sciences of Havana.

REFERENCES

Achtman, R., Green, C., and Bavelier, D. (2008). Video games as a tool to train visual skills. *Restorative Neurology and Neuroscience*, 26:435–446.

Billen, M. I., Kreylos, O., Hamann, B., Jadamec, M. A., Kellogg, L. H., Staadt, O., and Dawn, Y. (2008). Geoscience perspective on immersive 3d gridded data visualization. *Computers & Geosciences*, 34:1056–1072.

Castejón, C., Blanco, D., and Moreno, L. (2009). Friendly interface to learn stereovision theory. *Computer Applications in Engineering Education*, 17(2):180–186.

Clausen, T. (2005). Guest editorial undergraduate engineering education challenged by the bologna declaration. *IEEE Trans Educ*, 48(2):213–215.

CruzNeira, C., Sandin, D. J., and DeFanti, T. A. (1993). Surround-screen projection-based virtual reality: The design and implementation of the cave. pages 135–142. 20th Annual Conference on Computer Graphics and Interactive Techniques.

Johnson, A., Leigh, J., Morin, P., and Keken, P. (2006). Geowall: Stereoscopic visualization for geoscience research and education. *Computer Graphics and Applications, IEEE 26 (6)*, pages 10–14.

Kockro, R. A., Stadie, A., Schwandt, E., Reisch, R., Charalampaki, C., Ng, I., Yeo, T. T., Hwang, P., Serra, L., and Perneczky (2007). Collaborative virtual reality environment for neurosurgical planning and training. *Neurosurgery*, 61:379–391.

Kozulin, P., Ames, S. L., and McBrien, N. A. (2009). Effects of a head-mounted display on the oculomotor system of children. *Optom Vis Sci*, 86:845–856.

Limniou, M., Roberts, D., and Papadopoulos, N. (2008). Full immersive virtual environment cave in chemistry education. *Computers & Education*, 51:584–593.

Lipton, L. (1997). Stereo-vision formats for video and computer graphics. *Proc. SPIE*, 3012(239).

Martín, S., García, S., Suárez, J., and Rubio, R. (2007). Visage: A new stereo viewer and compilation software tool for geologic works. 23rd International Applied Geochemistry Symposium.

Martín, S., Suárez, J., Orea, R., Rubio, R., and Gallego, R. (2009). Glsv: Graphics library stereo vision for opengl. *Virtual Reality*, 13(1):51–57.

Palanichamy, C., Kumar, C. A., and Babu, S. (2004). An educational package for environmentally friendly, economic operation of power systems. *International Journal of Electrical Engineering Education*, 40(2):130–143.

Patryn, A. and Pietruszko, S. M. (2005). Didactic software for solar cells and materials parameters analysis. *Solar Energy Materials and Solar Cells*, 87(1-4):271 – 285. International Conference on Physics, Chemistry and Engineering.

Simons, K. (2005). Amblyopia characterization, treatment and prophylaxis. *Surv Ophthalmol*, 50:123–166.

Steven, E. (1999). The cathedral and the bazaar. *Knowledge, Technology &; Policy*, 12(3):23–49. 10.1007/s12130-999-1026-0.

Waddingham, P., Cobb, S., Eastgate, R., and Gregson, R. (2006). Virtual reality for interactive binocular treatment of amblyopia. volume 20, pages 370–374. 6th Intl Conf. Disability, Virtual Reality & Assoc.