Examination of the possibility to use OpenSceneGraph for real-time graphics using Immersive Projection Technology

Linus Valtersson, M.Sc
CKK
Chalmers University of Technology
SE-412 96 Göteborg, Sweden
d98linva@dtek.chalmers.se

Abstract

This paper describes the examination of whether OpenSceneGraph (OSG) [1] can be used to render real-time graphics with Immersive Projection Technology (IPT). It starts by describing the systems used for graphics today and motivates why it is desirable to use OSG instead. After this, the examination itself is described. The result from this examination is that OSG can be used to render real-time graphics in a IPT-environment, but the task is not trivial. OSG alone is not enough to render graphics satisfactory, it requires an extra layer dealing with all VR-features included in the environment. To solve this, VR Juggler (VRJ) [2] was used on top of OSG. VRJ is not the only option to use together with OSG, but it was the one that proved to work most satisfactory. The combination of OSG and VRJ was used to implement an application displayed at the International Science Festival in Gothenburg 2003.

1 Introduction

In the field of Virtual Reality (VR) many new and interesting environments have been developed, this includes Head Mounted Displays (HMD:s) and different IPT-environments, which are the focus in this paper. An IPT-environment usually consists of four to six walls displaying stereoscopic graphics and some kind of tracking-device (FASTRAK, Flock-of-birds etc.) that keeps track of the user's and possible interaction devices' positions in space. An IPT-environment can be used for a number of different purposes, it can display for instance buildings, molecules and landscapes and it can also be used to create different kinds of learning aids or even games. Whatever it is used for, real-time graphics is essential.

Early applications mostly displayed a static scene like a building or a molecule. The user could possibly rotate and translate this scene but the scene itself stayed static. When the environment was developed further the ability to create dynamic scenes arose. For dynamic scenes some kind of programming is required to update the scene according to user-interaction.

The IPT-environment that will be the focus in this paper is located at Chalmers University of Technology in Sweden. At present there are two different systems that are used to create applications for this environment and these are DIVISION Mockup [3] and CAVELib [4].

DIVISION Mockup is a graphical user-interface used to create interactive scenes. The scenes are usually created by using some kind of modeling-tool, for instance Alias, and the scenes are later exported to DIVISION Mockup. In DIVISION Mockup the finished scene is composed and it is also possible to add some amount of user-interaction.

CAVELib is an OpenGL-based programming-interface. When using CAVELib the scene is created by using OpenGL-commands to draw the different parts of the scene. It is also possible to update the scene according to user-interaction for each frame.

These two systems differ in a number of ways. With DIVISION Mockup most of the work is modeling and than some programming might be required, whereas with CAVELib the entire work is programming. In DIVISION Mockup the ability to add dynamics is very limited, but in CAVELib on the other hand this is not a problem. The problem with CAVELib is that it uses OpenGL and standard C, which often requires an extreme amount of code (and time) in order to create a scene. As mentioned earlier, the ability to create dynamic scenes is very desirable but the only means to do so at present is to use CAVELib, which often takes to much time to use. To solve this problem a new system is needed. With the new system it should be possible to create interactive dynamic scenes without having to write an extreme amount of code.

OpenSceneGraph (OSG) is a newly developed graphics library that encapsulates OpenGL. As the name suggests it uses scene-graph-technique [5], which is much more smooth to use than pure OpenGL. OSG is an open-source project still under development but it already has a lot of users both in research and industry. Due to its popularity it seems like a good choice for application development, that is, *if* it can be used with IPT. The rest of this paper describes the examination of if OSG can be used in an IPT-environment.

2 Technical description of the environment

As mentioned in the introduction, the IPT-environment, which is the focus in this paper, is located at Chalmers University of Technology in Sweden. The IPT-environment at hand is a 3x3x3 meter TAN VR Cube with stereo projection on five sides (the four walls and the floor). The VR Cube is run by a Silicon Graphics Onyx2 Infinite Reality with 22 MIPS R10000 processors at 250 MHz, 2GB RAM and three graphics pipes. Users of the Cube wear Crystal Eyes shutter glasses, which are synchronized with the graphics. A Polhemus FASTRAK system tracks the position of the user's head and the position of the interaction device used. The interaction device depends on the system. With DIVISION Mockup a DIVISION 3D-mouse is used and with CAVELib a wand is used. The wand consists of three buttons and a joystick. It is also equipped with a sensor, in order for the tracking system to be able to locate it in space. There are also a Polhemus Stylus and 3BALL available.

The main environment for the examination was the IPT-environment described above. But for some early tests a ordinary desktop computer was used. This was a Silicon Graphics o2 with one MIPS R5000 processor at 200MHz and 256MB RAM.

The version of OSG used was OSG 0.93, which was the most recent release at hand. It should be noted that some facts about OSG mentioned in this paper are not true for newer releases.

3 The examination

OSG was first installed on a regular desktop computer. It was shortly discovered that to use OSG on a desktop computer was not very difficult. The distribution comes with a number of example programs, which are all well documented. To use OSG in the VR Cube on the other hand proved to be much more difficult. The OSG has classes that can be used to create a simple window to display a scene or the scene can be viewed in full screen. The stereo-graphics is not a problem since OSG has support for the stereo-technique used - quad-buffered stereo. The five projection walls on the other hand were a big problem. It was soon evident that OSG alone was not sufficient; some kind of extra layer was needed to handle the complex environment.

There exist a number of different libraries that can be used to deal with multi display systems and/or different interaction and tracking devices. Most of these have support for a number of different graphic libraries, such as OpenGL and OSG. Since OSG is very new there are not many libraries that support it, but two of them where found and these where OpenProducer [6] and VR Juggler (VRJ).

OpenProducer is also an open-source project and like OSG it is still under development. It was not very surprising that OpenProducer would have support for OSG, because it is developed by Don Burns, who has also developed OSG together with Robert Osfield. OpenProducer can handle multiple displays but one disadvantage is that it has no support for the tracking system used.

VR Juggler is also an open-source project, but unlike OSG and OpenProducer it has been around for a while and already exists in a 1.0 version and a 2.0 is on the way. VR Juggler is designed to be able to handle graphics and interaction in a number of different VR-environments. It can handle both the hardware and tracking system used.

The first combination to be tested was VRJ together with OSG because VRJ seemed a little more promising than OpenProducer. The idea behind VRJ is that it should be possible to run the same application in different environments without having to re-compile the application. This is possible due to the use of configuration-files. When an application is started the VRJ kernel gets the current configuration-file(s) and sets up VRJ accordingly. The VRJ distribution comes with a number of sample configuration-files. These include one file to run an application in "simulator mode" on a regular desktop computer and one for a four-sided Cube. The first goal was to create a simple program and get it running in simulator mode and then try to get it running in the Cube. To create a simple program was not very difficult; VRJ comes with a detailed step-by-step programmer's guide [7]. The first program just draw a cube in the VR Cube by using OSG and to run this in simulator mode worked just fine. To get the program up and running in the Cube, on the other hand, was not as easy. The sample configuration for a Cube that was supplied with the distribution was for a four-sided Cube using two graphics pipes and a Flock-of-birds tracking system. This differs quite dramatically from the system at Chalmers that has five walls, three graphics pipes and a Polhemus FASTRAK tracking system. A new configuration-file had to be created.

In order to be able to create a new configuration-file a deep investigation of the system was conducted. When the test-program was run it turned out that something was wrong in the configuration. The drawing did not work at all as expected and it did not follow the main shutter glasses, which is totally necessary in order to get a satisfying result.

Since the initial testing of VRJ was not exactly satisfactory, some time was spent testing OpenProducer. OpenProducer has been developed with a movie producer in mind. The idea is that a scene is created, by using for instance OSG, and then the number of cameras (viewports) desirable are set up (position, direction etc.). Whatever is caught on "film" is seen in the different viewports corresponding to the cameras. To test the combination of OpenProducer and OSG, the same approach as with VRJ was used. The first problem was known from the start, that OpenProducer had no support for input-devices or tracking-systems. Separate libraries would have to be created for this. Another problem was that there appeared to be a bug in OpenProducer regarding stereo-graphics, at least regarding quad-buffered stereo, which is used in the VR Cube. Due to all these drawbacks with OpenProducer, VRJ was given all the attention.

The initial testing of VRJ was not at all satisfactory. After some further tests it seemed like VRJ had problems with the tracking system used, Polhemus FASTRAK. To solve this, an alternative approach was used. When CAVELib is used, the values from the tracking system are read from a tracking-daemon called trackd. VRJ has support to use this as well and the configuration-file was changed accordingly. When this was tested the result was much better than the first time, but it was not satisfactory. This time, the drawing reacted when the user moved, but it was still not possible to see what was being drawn. The coordinate system appeared to the rotated and possibly translated.

After getting the drawing to react to user movements, it was time to make sure that the drawing got right. Something was obviously wrong with the coordinates, so the first step was to print out the coordinates for different positions in the Cube and compare these with the expected values. The expected value-ranges for the x-, y- and z-coordinates are –5…5, 0…10 and –5…5 respectively. Unfortunately, these were not the ranges that were read. The width of the ranges appeared to be correct, that is, all coordinates had ranges with a width of about 10, but the ranges themselves where not the expected. The x-coordinate seemed correct, but the y- and z-coordinates seemed to be shifted by about 7.8 and 1.6 respectively. After a deeper investigation of the configuration for the environment, these values turned out to be the transmitter offset, which was set to (-0.01,8.0,1.65).

The trackd daemon uses shared memory keys to store its data and these keys have to be known in order for VRJ to be able to find the correct data. These keys where easy to find in the old configurations and when they had been entered in the configuration for VRJ it was no problem to get the correct data.

When the data for the wand could be read correctly, the only problem that remained was the translated coordinate system. In the VRJ configuration files it is possible to add offset values for the different sensors, so it appeared as if the problem would be solved easily. However, the VRJ kernel did not interpret the values at all as expected and therefore the offsets were removed and the problem remained. The solution was to let the coordinates have the ranges they had and set the coordinates for the walls of the Cube accordingly. In other words, instead of adjusting the coordinate system after the Cube, the Cube was adjusted to fit the coordinate system. When the coordinates for the walls of the Cube had been set to fit the coordinate system the drawing appeared to work satisfactory.

After a lot more tests and a thorough investigation of the VRJ source-code the cause of most remaining problems where found. The cause was a bug in the VRJ 1.0 source-code that causes the matrix representing the sensor-data to be created in the wrong way. VRJ creates the matrix by applying the rotation angles in XYZ-order whereas it is supposed to be in ZXY-order to fit Polhemus FASTRAK. After correcting this, a re-compile of VRJ was attempted. But as it turned out, none of the machines available could understand the makefiles supplied with the VRJ distribution. Instead the solution became to create a new class that handles all contact with the trackd daemon and use this instead of the VRJ-classes. After this fix, everything appeared to work satisfactory.

4 Result

The goal was to examine if OSG could be used to render real-time graphics in an IPT-environment. As stated in the last part of the examination, a working configuration could be created in which OSG is used to render graphics. The task was not trivial however. Since OSG is “merely” a graphics-library and has no support for multiple displays, it alone can not be used with IPT. This is due to the fact that an IPT-environment uses multiple displays and furthermore the environment has a tracking system and several input-devises that OSG has no support for. The solution became to use an extra layer on top of OSG that handles the advanced features of the environment. This layer is at present VRJ, which can handle all advanced VR-features needed. Together they form a working platform, which has been used to create several sample applications and a larger application displayed at the International Science Festival in Gothenburg 2003 in collaboration with Swedish Television (SVT).

5 Future work

Even though a working platform has been configured, the examination is not complete. There are still questions that need to be answered. The current platform uses the wand for user-interaction. But as mentioned in the technical description, several other interaction devices exist. One question that needs to be answered is if it is possible to use any of these other devises.

When the basic examinations have been completed, it is time to look into how the platform can be improved to simplify application-development and fit the needs of the users of the system. Most users are not programmers but are more used to modeling. This implies that some kind of platform where the programming is minimized might be desirable. On the other hand, users who want to program their application from the bottom up should of course have the possibility to do so. With this in mind, a platform where the users can choose the level of abstraction depending on their needs and knowledge could be a good solution.

To develop applications using the current platform one needs to gain knowledge about both OSG and VRJ even though VRJ is mostly used just for getting tracking-information and to set the drawing-routine. Having to gain knowledge about both of these seems unnecessary even for the skilled programmer. To gain knowledge about OSG should be enough. This can be accomplished by developing a basic platform, which hides VRJ from the user. A skilled programmer can then use this basic platform to develop applications, while this platform can be developed further to get a higher level of abstraction for the not so advanced users. For the users who do not wish to program at all this can be further developed to create a graphical-user-interface similar to DIVISION Mockup.

Another approach is to perform further testing of the OSG+OpenProducer combination when OpenProducer becomes more stable. When the original testing was performed OpenProducer was in early alpha stage but now it has evolved to beta and the ambition is of course to be able to release a 1.0 version in a near future. It still has no support for tracking devices, but the data for these can be obtained by using the class that had to be written to deal with the bug in VRJ, and the displays can then be updated manually. This might improve performance because since OSG 0.94 OSG and OpenProducer are tightly integrated. The disadvantage is that the applications will be system-dependent since the code that handles the readout from trackd is system-dependent.

The same approach as for the OSG+VRJ combination could then be used. That is, to develop a platform with increasing steps of abstraction up to a GUI-interface.

So far the discussion has only been about graphics. But in DIVISION Mockup it is possible to use sound in the applications by using the Lake-system [8] available at Chalmers. With the Lake-system it is possible to get 3D-sound which can be "positioned" in space. To be able to use sound is a powerful tool, which makes applications more "alive" and therefore it is desirable to be able to use sound when using the new platform as well. No investigation at all has been made in this area so this is a big step to take.

Acknowledgements

This examination was carried out at CKK during the spring of 2003 and it would not have been possible for me to get this useful experience if Sven Andersson and Josef Wideström at CKK had not believed in me.

Finally I would like to thank Josef Wideström for his support during this examination and valuable help in writing this paper.

References

[1] OSFIELD, R. Introduction to the OpenSceneGraph; http://openscenegraph.sourceforge.net/introduction/index.html. 2003-09-14

[2] About the Juggler Suite of Tools. http://www.vrjuggler.org/about.php; 2003-09-14

[3] Division MOCKUP. http://www.division.com. 2003-09-14

[4] CAVELib overview; http://www.vrco.com/products/cavelib/cavelib.html. 2003-09-14

[5] BAR-ZEEV, A. Scenegraphs: Past, Present and Future. http://www.realityprime.com/scenegraph-php. 2003-09-14

[6] Introducing OpenProducer. http://www.andesengineering.com/Producer/index.html. 2003-09-14

[7] VR Juggler: The Programmer's Guide. http://www.vrjuggler.org/vrjuggler/docs.php. 2003-09-14

[8] Lake Technology Limited. http://www.lake.com.au. 2003-09-14