3D Output Devices for the GRAPECluster Project
Independent Study Report
By Andrew Piotr Bak

Sponsored by:
Hans-Peter Bischof
Department of Computer Science
Table of contents

I. Introduction

II. What was done

III. Conclusion

IV. Appendix I: modified Makefile

V. Appendix II: stereo 3D movie Output chart

VI. Appendix III: list of modified files
I. Introduction:

An important aspect of the GRAPEcluster project is the display of the output. In the winter quarter 2004-2005 we hoped to setup and get the grape cluster project to work with a few 3-Dimensional output devices like 3d goggles and a DTI 3D’s 3D lcd monitor. This would give the user a more intuitive perspective for the 3D space.

The monitor used was DTI 3D’s model 2015XLS which supported all popular 3D modes: Side by side, frame sequential, and field sequential. More information on this monitor is available at [www.dti3d.com](http://www.dti3d.com). The monitor uses a lattice of light columns to send all the odd columns to the left eye, and the even columns to the right eye. DTI calls this Parallax Illumination. This means that each eye sees half of the 1024x768 pixels. More information on the hardware is available at [http://www.dti3d.com/About/dti_technology.htm](http://www.dti3d.com/About/dti_technology.htm). Despite its promising look the DTI website was less then helpful in setting up the 3D monitor. The monitor also had glitches with it. When it had a substantial amount of 2D material, and sometimes in 3D mode, it would lose track of what mode it was in and freeze for a few seconds. It would also show artifacts. Due to these issues we only used the monitor for 3D display and used an old CRT to work on.
The i-glasses 3D goggles that we used were successfully tested in frame sequential 3D mode. The documentation on the i-glasses was weak and not specific to which 3D modes they supported. The goggles were notably heavy and annoying to wear. Their resolution for each screen was a mere 800x600.

The following is a synopsis of the things which I have worked on over the course of the quarter. Even though there were hurdles to overcome I want to give special thanks to Sam Waters and Hans-Peter Bischof for getting the team what it needed.

II. What was done:

Upon starting this project I wanted to see if I could get an OpenGL program, like Half-Life to run in Stereo 3D. Since these devices advertise to work with such programs. This came from the assumption that if one OpenGL program worked in stereo 3D, then Java3D with OpenGL would also work. Working on a computer in Break Out room 2, we quickly realized that the stereo driver wasn’t working properly. Since the computer we were working on was used for other projects, I thought it would be best to get a clean machine to work with. This would eliminate many factors which may have been causing the malfunction.

The team got access to a Sony Vaio desktop to work on the project thanks to professor Bischof. This machine needed to be re-formatted, and a video card with 3d acceleration hardware was needed. Sam Waters hooked us up with an ATI Radeon
Bak 5

9700 pro. This seemed to be a good choice at the time. After installing a clean OS, and the Radeon, I did some research.

Apparently ATI has no stereo drivers for their cards. And most stereoscopic web sites only support nVidia products and the native nVidia stereo drivers. Going back to Sam I obtained a nVidia GeForce 5900 ultra. This card was a much better choice and the driver seemed to work properly. The Sony Vaio had no com ports so Sam provided an USB serial com port. This was used for the DTI 3D Monitor, and the Flock of Birds. Overall Sam Waters and his office were extremely helpful in making progress from the hardware side.

Since the cost of the DTI 3D monitor was so high, we setup a programming environment in Hans-Peter’s old office. This was a good working area for the team since it had only 3 people. Any more and the place would have been crowded.

With the monitor running and the stereo graphics producing 3D images in Half-Life in OpenGL mode. I figured we were only a step away from making this thing work with Java3D. I was wrong. After reading the Java3D manual it was clear that setting a Java3D flag “–Dj3d.stereo=true” didn’t solve any problems. Then after reviewing countless forums it was clear that this stereo 3D technology peaked about 5 years ago and there were few new products or technologies with it. After coding various examples of programs that worked for other people I learned how to use advanced features of Java Swing among other things.

The main sample program I worked with was HelloUniverse. HelloUniverse is a Java3D sample program which shows a multi-colored cube spinning. When running HelloUniverse in full screen mode and enabling nVidia stereo driver the image would
flicker and disappear. Un-toggling it would cause the cube to re-appear. So the
driver did recognize that this program was running OpenGL, but failed to bring it into
the stereo realm. The only posted success stories on getting Java3D working in
windows were on Windows 98, with select outdated video cards. This was
discouraging. Not wanting to revert to old systems, this seemed to lead to an endless
search which could end up at a dead end.

Realizing that the pre-made solutions wouldn’t work I started to investigate
programming around the problem. Knowing that Spiegel could show multiple
camera’s by using the built in language I figured I could make two cameras and move
each camera to a different eye position. By putting the camera windows side by side
the 3d monitor could engage the side by side 3D mode and produce a 3D image. I ran
the following command to show a proof of concept:

Camera new 1 1
Camera new 2 2
Camera 1 moveto 5 0.2 0.1
Camera 2 moveto 5 -.02 0.1
These commands would make two camera views, which produced two
perspectives on the same scene. Knowing this was possible I started to solve the
math problem which would properly show a stereo 3D image.

Knowing eyes are about 2.5 inches apart, and focusing on a monitor which was
about 3 feet away the calculated angle is 0.034708 radians off center for each eye.
Knowing that a constant focus on position <0,0,0> was necessary I converted the
current position into spherical coordinates centered at <0,0,0>. By modifying theta
for each eye by 0.034708 the produced image would appear in stereo.
At first I decided to hardcode this equation into the Flock-of-Birds plug-in and the Joystick plug-in in order to produce a rapid prototype. The outcome of this was two cameras that seemingly moved in a “stereo mode”. It was hard to see a 3d image. The cube in the middle of the display would appear more 3D then any of the stars. This approach caused the each camera to render its own world and would show very poor performance when running the simulation through time.

The next step was embedding the stereo code into the camera view portion of the project. By coding it in the inner level of Spiegel there could be performance gains and easier usability.

After reading over countless lines of Spiegel code I found the classes that I needed to edit to make stereo output work. The following classes were key: Flyer, Display, CapturingCanvas3D, and Spiegel. The display was modified to work in full screen move at 1024 by 768, which was required by the DTI 3D monitor. The Display class was also modified to make two scenes in the same world so that there could be two angles with only one world. Each canvas was sized to 512x768. The display also created the two Flyer objects, and associated them with each other.

The Flyer objects were modified in the moveto method. When in stereo mode the Flyer would know which eye it was and adjusted x,y,z coordinates with the spherical coordinates algorithm described above. After the master moved its flyer to the left location it would inform the slave to move to the right eye location. With this setup working the images were much more consistent. Before, when moving in time the left eye or the right eye would black out between frames, but with this new setup there would either be an image for both eyes, or neither. This is consistent with a
single camera moving forward through time. Due to this shortcoming in the Spiegel program I decided to make a movie, this would guarantee all frames rendered properly.

Trying to make a stereo movie lead me to modify the CapturingCanvas3D, and Display. I modified CapturingCanvas3D by adding an eye variable which would specify which eye the image was coming from. The display, which makes the CapturingCanvas3D objects, was modified to specify the eye the object was capturing from. This way when the images were captured the were saved to files starting with a “L” or a “R” symbolizing left and right respectably.

The Spiegel class was also modified to add the command line argument “3DStereo”. If set to true the camera used is a 3D stereoscopic camera instead of the regular camera. With this option set to false (the default), Spiegel runs as it did before any stereo code was added to it, with the exception of the 3DOutput plug-in.

The 3DOutput plug-in was a plug-in that I added to the Spiegel GUI in order to create a 3D display without using any command line type things. Using this plug-in overrides the default false setting on stereoscopic 3D camera to true, and creates a 3D stereoscopic camera.

The next stage in the movie making process was combining left and right images into one image. I wrote a combiner program to do this. This program would open a set of images starting with the first image in the left set. Then the user would pick the first image in the right set. Upon opening the first image in the right set the program would open each set create a single image and output to a new file name. Then the
output images would be sent to professor Bischof for movie creation using QuickTime Professional.

The movie, although created seemingly properly, didn’t work as well as expected. There seem to be performance issues with running a 1024x768 movie, or there was frames missing. In either case it’s a bit jumpy at parts. Also I have notice that it is difficult to see the output for just one eye out of the DTI 3D monitor. With the black background of space the x-axis for both stereo outputs can be seen from most locations. It would appear that without goggles it is hard to keep images isolated from each other. Another issue was that the i-glasses, goggles, my group used didn’t support the side-by-side 3d which is used in the stereo movie. One possible solution would be to make a left eye, right eye, frame alternate movie at 120 frames per second, since the goggles support 120 Hz alternate frame mode stereo 3D. The images would also need to be modified to fit in an 800 by 600 environment.

III. Conclusion:

The team this quarter had a better dynamic than last quarter. This quarter’s team consisted of Jin Zhang, Trevor Blanarik, and myself. Having meetings twice a week assured that we all stayed in close contact, but also assured enough apart-time to make individual progress.

Since Trevor is a freshman he played mostly an observer role in how things are run. He was helpful in testing, and got the goggles working with the Half-Life in OpenGL with nVidia stereo drivers. I believe that he learned a lot of thing in programming and other areas of Computer Science, and has high potential for pushing the GRAPEcluster project to completion in his next 4 years.
Jin performed exceptionally well especially with a full, CS heavy, course load. She was initially tasked to port the Joystick (JInput) code to work on the Mac OS X platform. Each week she would come in with new ideas on how to get it working showing progress every time. Eventually the JInput did recognize the controller on Mac OS X, unfortunately the code had not been written by the JInput program to poll data from the controller rendering her efforts into a big education lesson. She was also helpful in helping me to code faster by using extreme programming. She caught a few bugs before they became bugs. Also she found out that the newer version of Java actually caused JInput to crash so it only works up to version 1.4.2_03.

When looking back at the equation that I used to see a stereo image I realized something wasn’t right. There was a stereo image at some angles but not so much at others. Upon further review I realized that the equation will only work if Z is near zero. An orthonormal basis transformation to align the xy axis with the current position, and <0,0,0> would create the proper scenario for modifying theta. I recommend using a math major or professor to ensure the math is done right. It is easy to mess these things up. This is one reason why at some angles a 3D image could be seen, while not at others.

Besides the slightly right equation used, the monitor didn’t produce good 3D on black background with white axes. The Parallax Illumination didn’t isolate each image well enough so only one eye would see it. This made it harder for the brain to create the 3D image.
Throughout the course of this independent study there were many things learned.

The first thing being that simple looking things aren’t always simple. Specific things learned include, but are not limited to:

- Intricacies of the Spiegel program
- Greater understand in Java3D
- Greater understanding of stereo 3D technologies
- Advanced knowledge of Java Swing components
- Some incites into other technologies researched: JOGL, JInput, JXInput, Muppets, ESLA 3D, DTI 3D technology, …
- Having multiple meetings a week helps keep the team motivated even in hard time.
- A greater knowledge of Java Image classes.
- Math that goes into stereoscopic images.
- JNI for mapping between Java and C++ as well as proper Makefile making were taught to all the grape students by professor Bischof.

Even though there is more work left in this area, I believe my team and I have taken a considerable chunk out. With the high potential of true 3D possible, I believe that someday GRAPECluster will have a good stereo output system.
IV. Appendix I: modified Makefile

The following target was added to my Makefile in order to create stereo 3D movies:

```
movie3d:
    rm -rf Gif_Movie
    mkdir Gif_Movie
    $(JAVA) $(JFLAGS) spiegel.Spiegel
        --feeder-port=1000
        --debug=false
        --batch=true
        --dirName=Gif_Movie
        --inputFileName=../theSimulation.sim14003.color.compact.binary.Z
        --deltaT=0.01
        --flyPath="DarthVader"
        --createMovie=true
        --peek=false
        --blackHoles=3
        --3DStereo=true
```

* Note the only different line is the highlighted line
V. Appendix II: Stereo 3D movie Output chart
VI. Appendix III: list of modified files

**Movie:**
- Spiegel/viewcontrol/CameraView/CaptureingCanvas3D
- Spiegel/viewcontrol/CameraView/Display

**Stereo 3D:**
- Spiegel/viewcontrol/mapview/plugins/Outputs3d/*
- Spiegel/viewcontrol/CameraView/Display
- Spiegel/viewcontrol/Flyer
- Spiegel/Spiegel

**Modified for prototyping, but not changed in the end:**
- **Joystick plug-in**
  - Spiegel/viewcontrol/mapview/plugins/Joystick3d/*

- **Flock of birds plug-in:**
  - Spiegel/viewcontrol/mapview/plugins/FOB/*