SET09115 - Report

# Introduction

The aim of this project was to create a 3D scene using OpenGL.

# Scene Design and Inspiration

For my initial idea I envisioned the use of video as textures, I would’ve created objects to be used as “billboards” but with moving images. Early on as well the idea of using multiple cameras was prevalent as well as the idea of getting a camera render to render to texture, to have an object act as a “security camera” of sorts. Both these ideas didn’t make it into the final project due to time constraints but there are still fragments of code in the project where I attempted this.

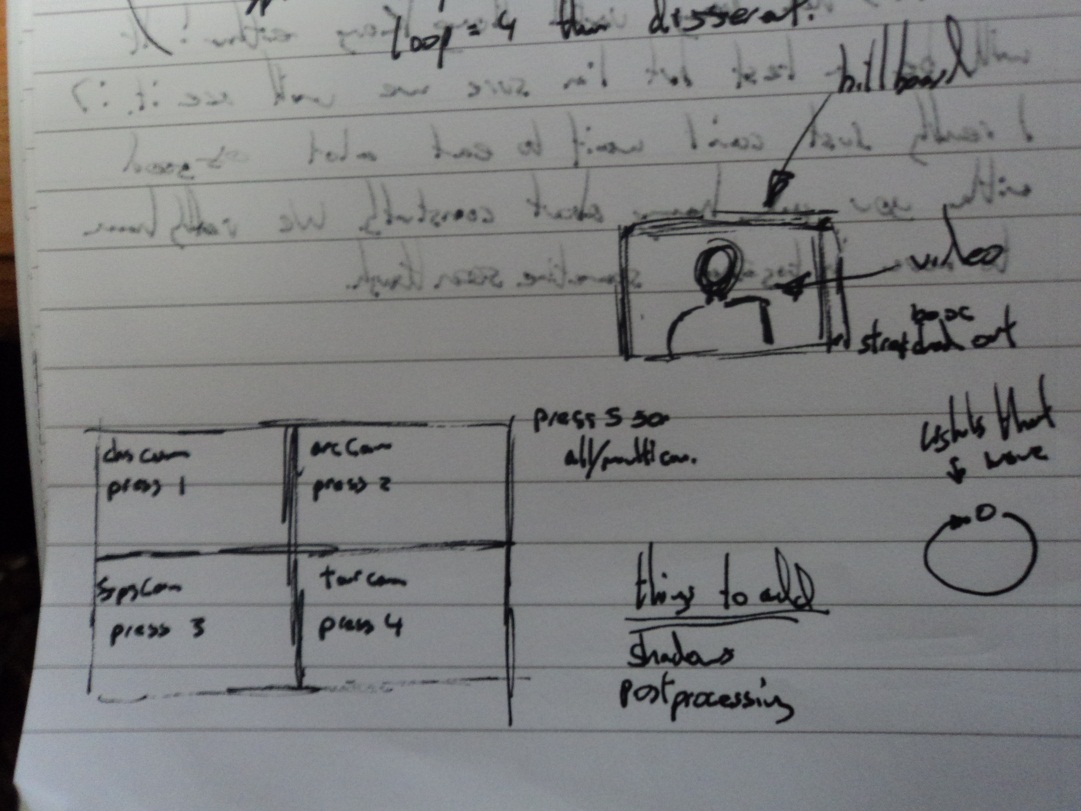
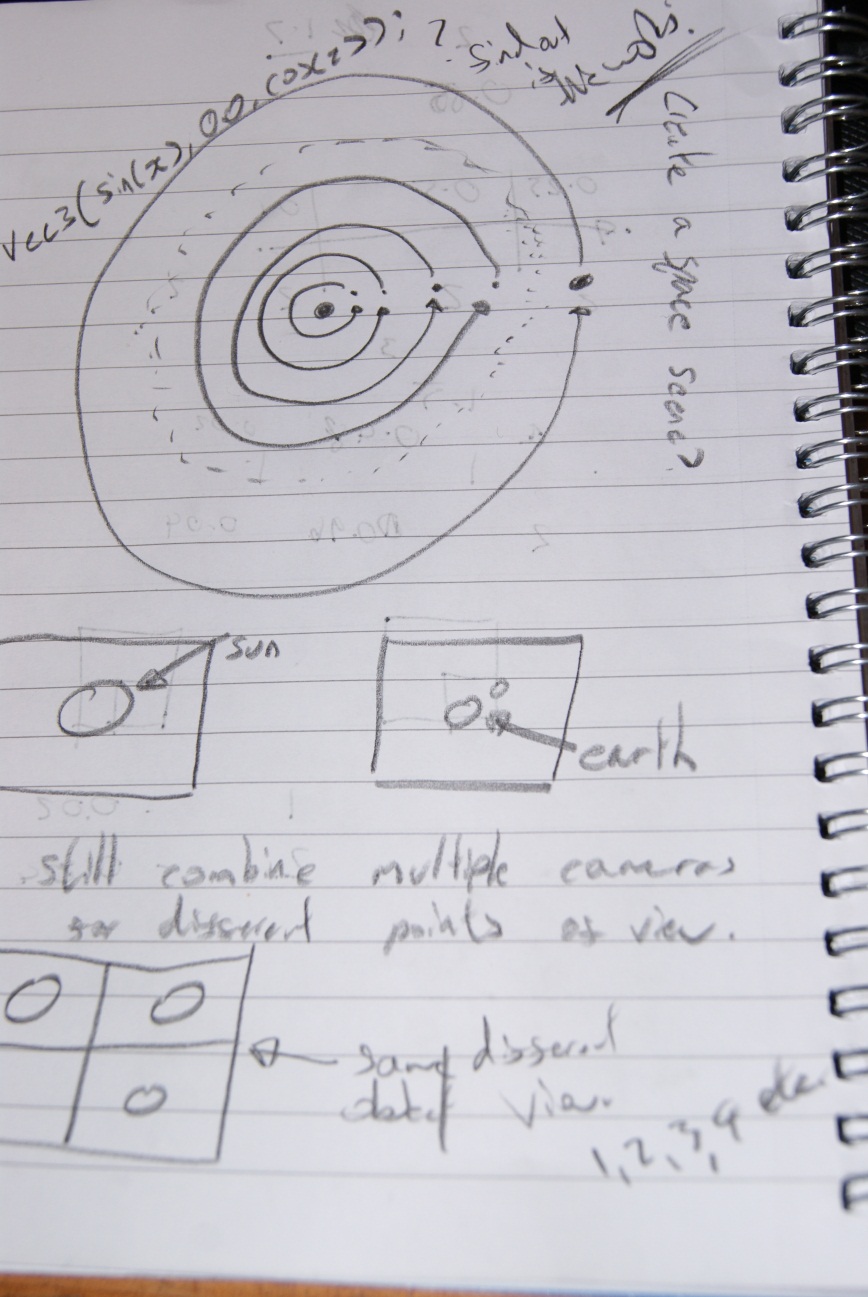


Figure 1. Initial plans

The contents of the scene gradually changed to suit certain needs and many ideas didn’t come about until later stages of the project. The idea for space, for example, didn’t come about until I changed the skybox from the given one to a space one I found online. Naturally from this point the inclusion of planets and creating an orbital system was natural and as such is how the project came to its current state.



When I realised I couldn’t achieve my primary goal I had to rethink and create something new and this is how I came to my current idea. Many of the final elements in the scene such as multiple cameras still managed to make it into the final program.

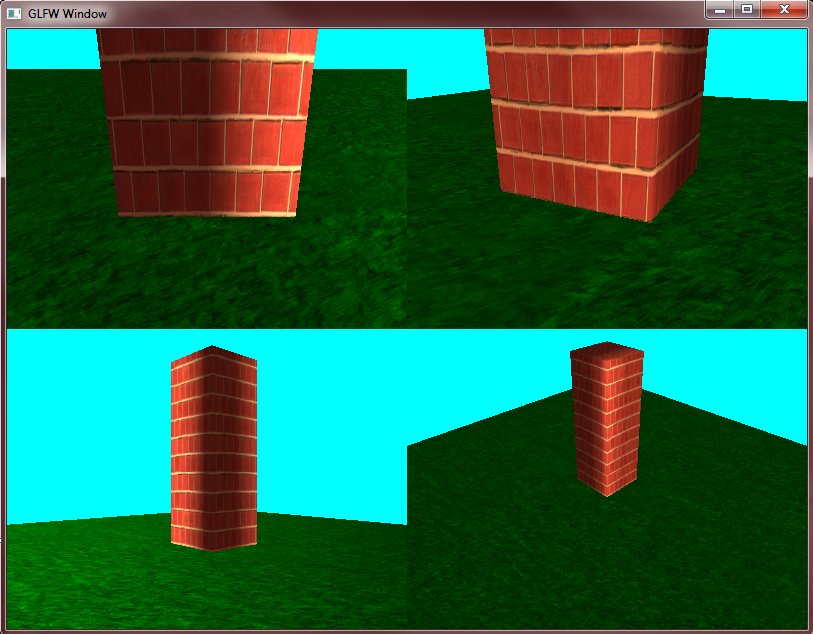


Figure 2. Early prototype Multi –Camera

# Background Research

For the project I had to investigate reading in AVI from files as well as other post-processing techniques. Whilst trying to do the AVI I referenced a tutorial by NeHe which can be found [here](http://nehe.gamedev.net/tutorial/playing_avi_files_in_opengl/23001/). This tutorial was very useful and I managed to get the project reading in the video to a texture in memory but couldn’t get it to apply correctly to any objects.

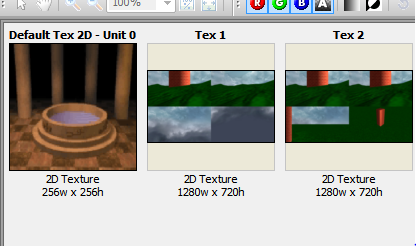


Figure 3. A frame of the video as seen in gDEBugger

Due to time constraints and the impending deadline I couldn’t spend much more time trying to get this to work and instead worked on a few other things. After creating my scene I did some research into post-processing techniques, after managing to implement the Sobel Operator in the edge detection shader I read about the Scharr Operator and instead implemented that

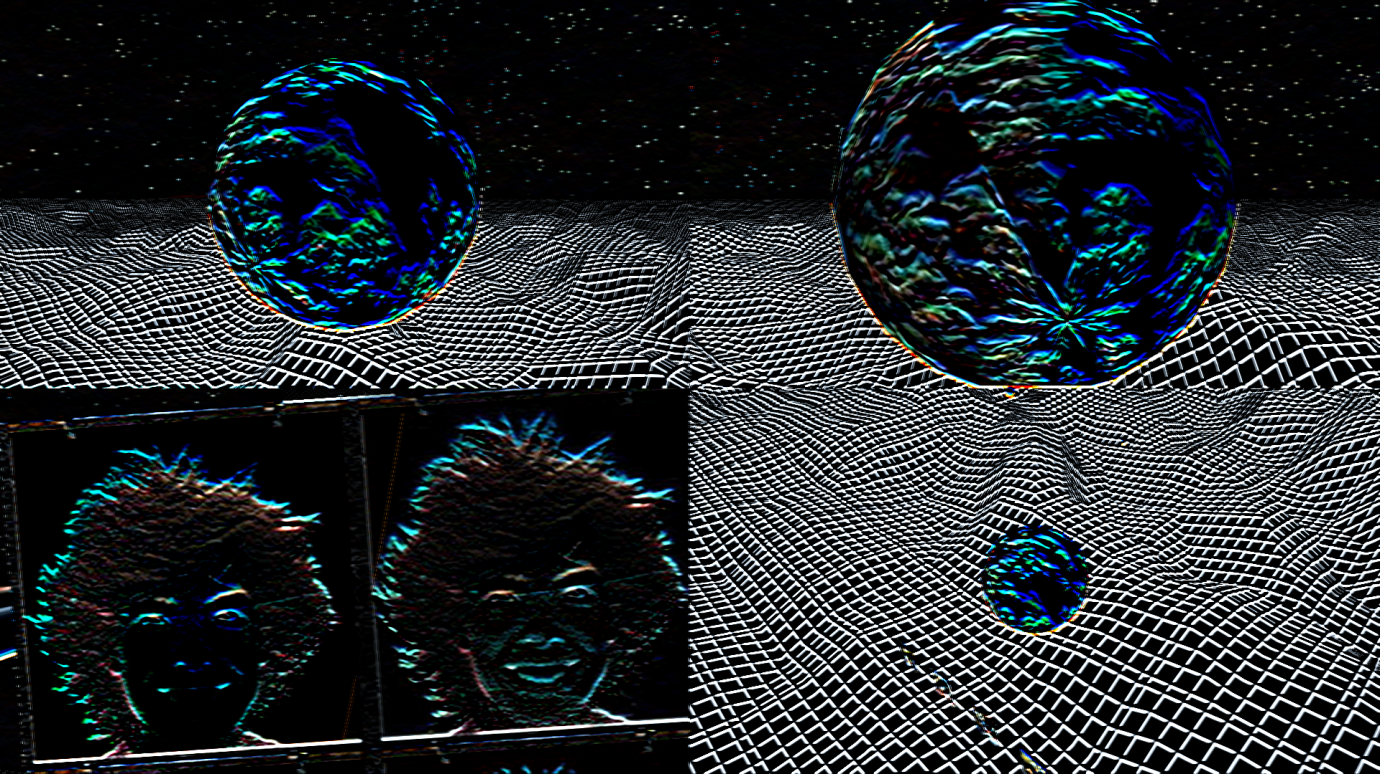
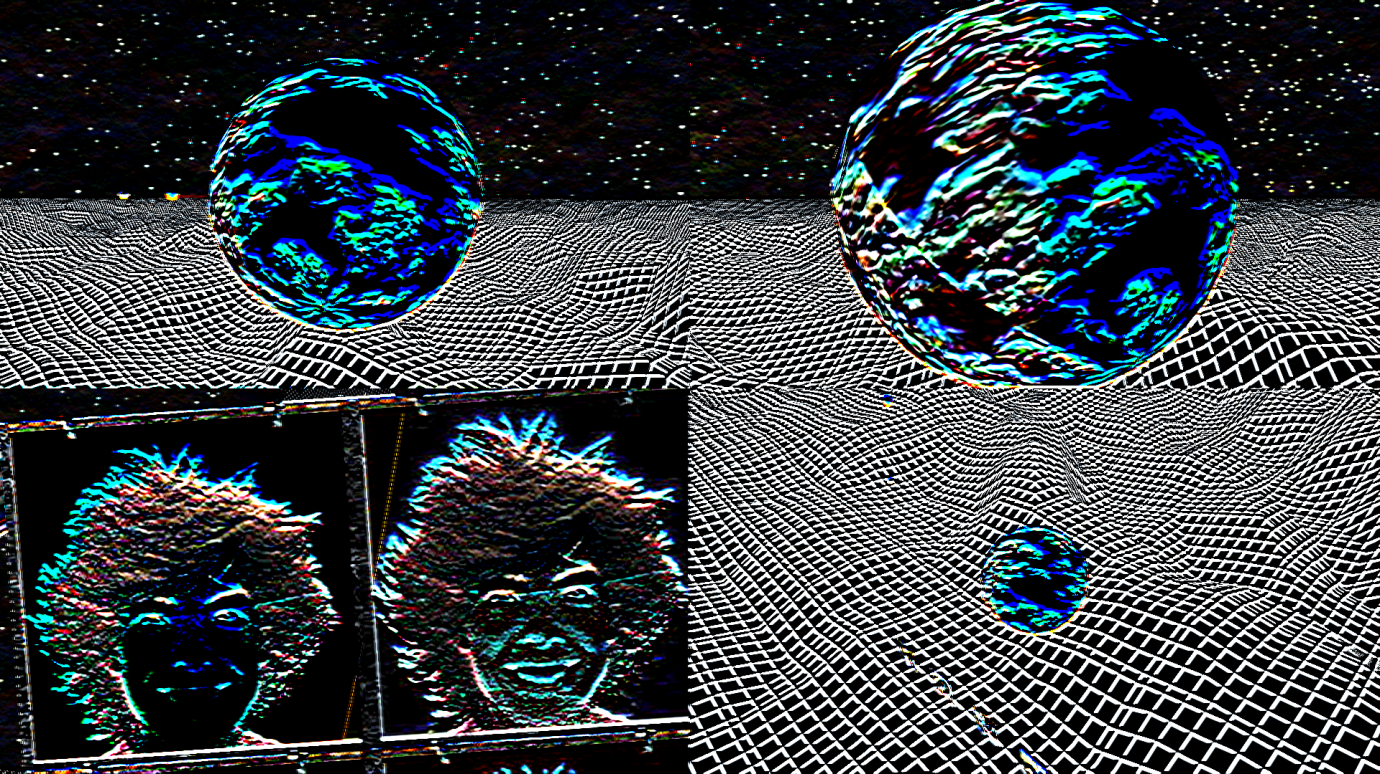
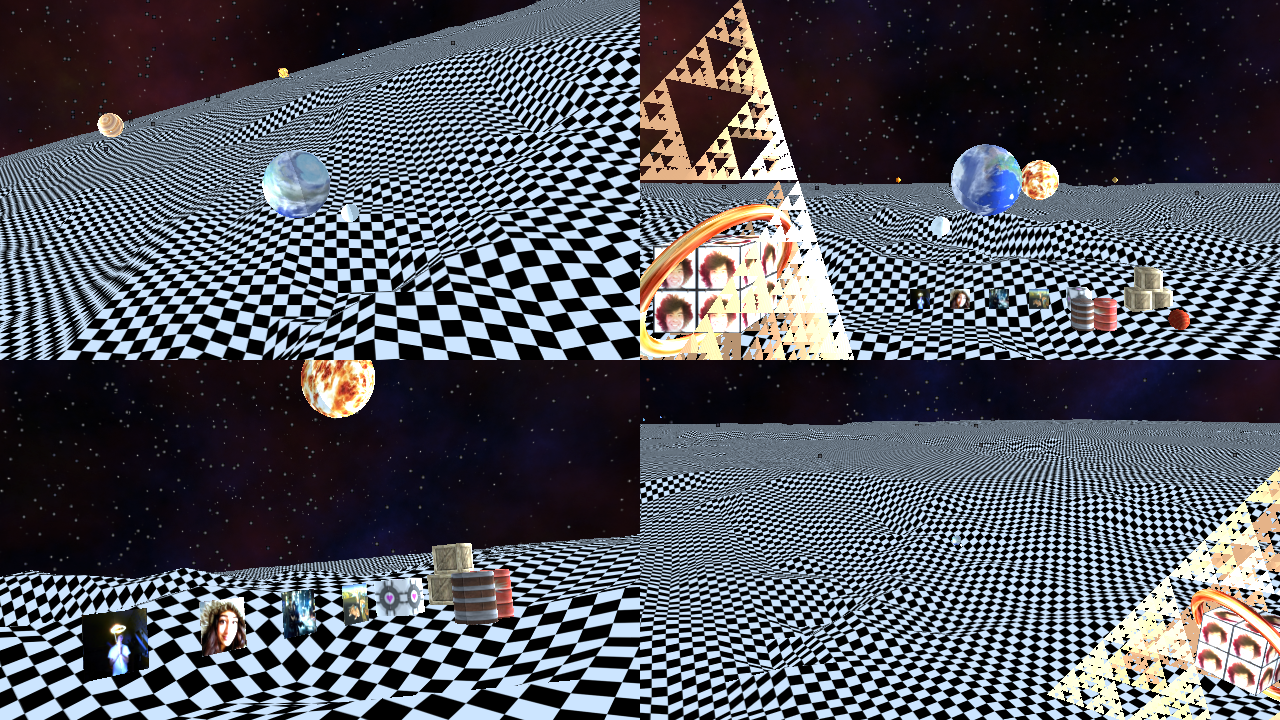


Figure - Scharr Operator on the left vs the original Sobel Operator on the right. (Note the more vivid stars as well as being able to see the terrain further away)

I also researched how to do multiple cameras which was made easy by the use of glViewport(). After doing this I also realised I wanted to add in the ability to set custom resolutions, set the program to fullscreen and to hide the mouse which was made possible with the documentation that comes with GLFW.

The final thing I researched for my project was general data on the solar system and the orbits of planets around others. I mainly referred to Wikipedia and other science websites for the information before implementing it within my project to give accurate movements. Most of the textures I have used have been found on Google Images.

# Implementation



For my final implementation I decided to create my space scene. I used the point sprite effect to model the stars shooting past. There is spotlight which is attached to whichever object the current focus of the camera is on. I created a piece of terrain using a heightmap which is the large panel on the floor. Also for this heightmap I used procedural texture generation to create a texture upon start up. I have also used the dissolve effect to dissolve the entire scene.

# Performance Evaluation

Using gDebugger I analysed my program at different settings. Each of the following images demonstrates the program at different resolutions with and without anti-aliasing enabled.

Incase it is illegible each column is represented by this, (FPS, OGLCalls/Frame, CPU Average Utilization, Texture Objects, Vertices/frame, Triangles/frame, Points/frame, Primitives/frame, texels, Used Virtual Memory in bytes)

Test System 1 – CPU – AMD Phenom X6 1055T at 2.8GHZ

GPU – GeForce GTX 460 1GB

RAM – 16GB – 1600MHZ

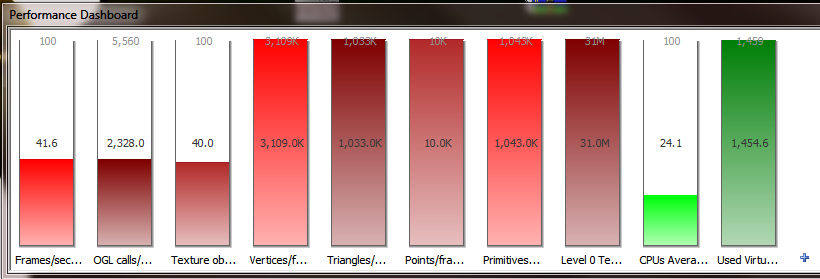


Figure – 1920\*1080 – No Anti-Aliasing



Figure - 1280\*720 - No Anti-aliasing

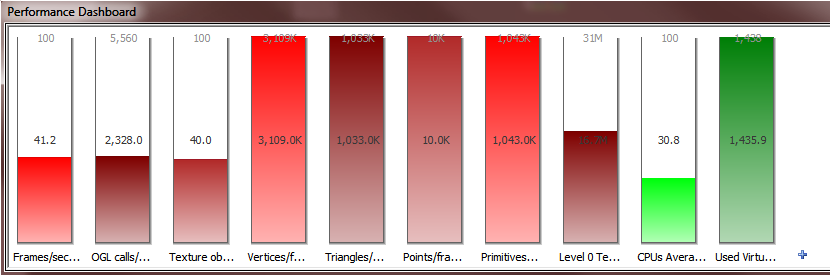


Figure - 800\*600 - No Anti-Aliasing

After looking at these results we can see that the majority of the statistics have stayed the same regardless of the resolution. The main consistant changes are in texels and Used Virtual Memory showing a 10byte difference between the resolutions.

Anti Aliasing didn’t function correctly below 16x and only at this level made a significant effect on the system dropping the framerate well below 20 on each resolution. In the interest of saving the planet the images have not been reprinted.

# Overall Evaluation/Conclusion

Despite missing my primary goal of including the AVI and camera texture effects I still managed to use many different effects including terrain generation, skyboxes, procedural texture generation amongst many others.

* You need to write a report. This should include the design and evaluation of your scene, and any other relevant information. More details are available in the marking schedule.

|  |  |
| --- | --- |
| Report and evaluation | Your report should include a design for your scene, and an evaluation on how well your scene meets the design. Good marks are given for showing where the inspiration for your scene came from. Performance evaluation using gDebugger and research background for any additional effects used will also be rewarded. The only code samples should be for researched effects. Submission of printed code for standard techniques is not considered. |