An OpenGl Desert Scene

Sam Serrels
40082367@napier.ac.uk
Edinburgh Napier University
Computer Graphics (SET08116)

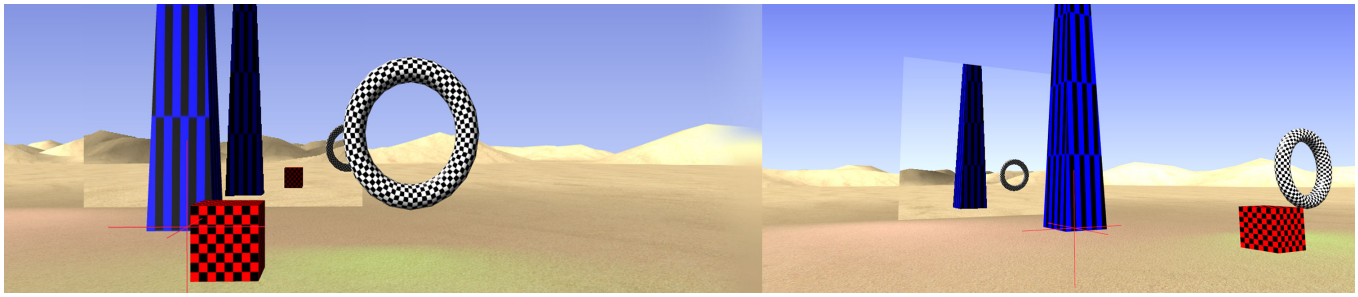

Figure 1: *Project screenshot*

Abstract

This project aims to develop a Real-time 3D graphics scene, with an emphasis on high aesthetic quality. The scene will use a pre-written OpenGL render framework to save development time for producing graphical effects. This project aims to also look into the cutting edge features available in the newest OpenGL standards and investigate the performance benefits of such features.

Keywords: OpenGL, GLSL, Graphics, SSBO, Reflections

1 Introduction

Figure 2: ***Project Inspiration - The Witness*** *- [Thekla, Inc. 2015]*

Project Aims The setting for the project scene is a Desert. Although not a visually busy scene, a desert provides vibrant and harsh lighting, interesting landscape and plenty of opportunity to squeeze more visual fidelity out of a small amount of scene elements. Using a multitude of texture effects, such as bump and parallax mapping, blend maps and level-of-detail, this project plans to bring life to even the most basic of objects, like sand on the ground.

Sky With the wide open vita of a desert plain, the sky takes center stage and is the most important visual cue to selling the realism of the scene. This project aims to have a fully dynamic and procedural sky system, without relying on static textures. This will allow for a time-of day system and full control of elements such as clouds and sun position.

Water The centrepiece of the scene will be a water feature of some kind, providing a contrast against the dry desert and visually interesting element in it's own right. Reflections, waves using distortion maps, refraction, and particle effects will be used to provide realistic looking water.

2 Background / Related Work

Figure 3: ***Spec Ops: The Line*** *- [Yager Development 2012]*

Desert scenes are not uncommon to video games, the simplicity of the landscape was a large bonus to performance limited software. Older games (pre DirectX 9), could get away with a simple ground mesh, a single sand texture and an interesting skybox, and this would be considered a sufficiently detailed scene for the time. See Figure: 4
As the graphical power of computer increased, the challenges to make a realistic desert increased dramatically, the standard of games required more than just a barren wasteland. Buildings, foliage, human characters and interesting landscape were needed and now the scene is just as complex as any other environment. Some modern games have taken up the challenge to model sand behaviour, as a very simple fluid dynamics system to add life to a scene. Sandstorms are a popular feature as they offer the valuable benefit of obscuring level

elements, these elements now do not have to be rendered, therefore increasing performance.

Figure 4: ***Project Inspiration - Guild Wars*** *- [ArenaNet 2005]*

Sky Procedural skys have been used in games for many years, the choice to use them depends on the needs of the game. Games that have aim to have a more living environment are the primary uses for procedural skies, in other games if a static image is sufficient, then a simple texture is used.

Figure 5: ***Procedural Sky - Fallout 3*** *- [Bethesda Game Studios 2008]*

3 Overview

4 Implementation

5 Evaluation

Scope The results of the project have been inconclusive in determining the performance benefits of physics optimisation across multiple platforms. The main shortcoming has been the Playstation 3 portion of the project. Due to unforeseen technical problems and the time constraints of the project, the support for the Playstion 3 could not be completed to a level which would result in usable results. Issues with the rendering framework and subtle differences in the maths libraries across the two platforms caused substantially more time spent on fixing code issues than developing features.

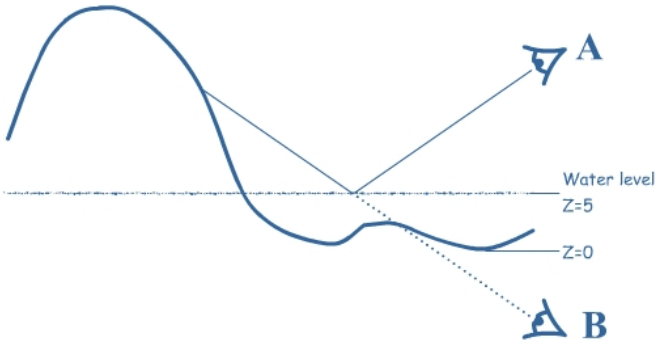

Figure 6: ***Refection Camera position*** *- [Riemer Grootjans 20011]*

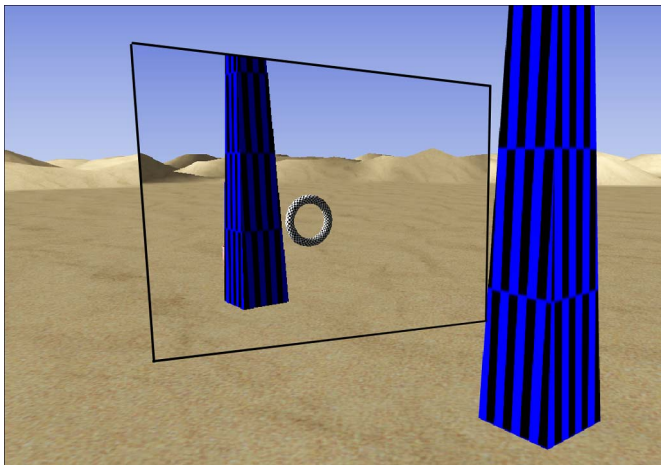

Figure 7: ***Rendered Scene*** *- Mirror Outline overlay for clarity*

Playstation 3 results Unfortunately, optimising the code for the strengths of the Playstation 3, such as subdividing the work to the separate SPU processors was not achieved. Rendering issues caused the final output of the project to be generally unviewable on the Console.

PC results The results of the simulation were more positive on the PC architecture. The simulation runs well and without any major issues, major features detailed in the original design have been omitted due to time constraints. Visually the program is basic, without any graphics embellishments to bring the output closer to the original design vision.

6 Future Work

The primary focus for future work would be to fix and complete the Playstation code to bring it upto par with the PC code, then other features can be worked on. Firstly, dynamic control of simulation variables at runtime, allowing the user to interact with the simulation while it running. The User should be able to adjust parameters such as the size of the world, the attributes of the spacial partitioning, and the ability to create and remove balls at will.
More complex collision shapes would be the next area of work, followed by rendering improvements.

References

ARENANET. 2005. Guild wars.

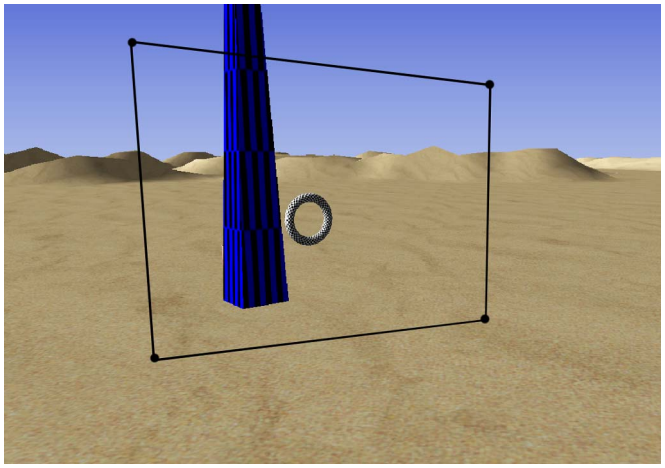

Figure 8: ***Refection Camera View*** *- UV coordinates shown, this is the area that will be rendered on the mirror quad*

BETHESDA GAME STUDIOS. 2008. Fallout 3.

RIEMER GROOTJANS. 20011. Reflections. *Accessed: Feb 2015*. www.riemers.net.

THEKLA, INC. 2015. The witness.

YAGER DEVELOPMENT. 2012. Spec ops: The line.