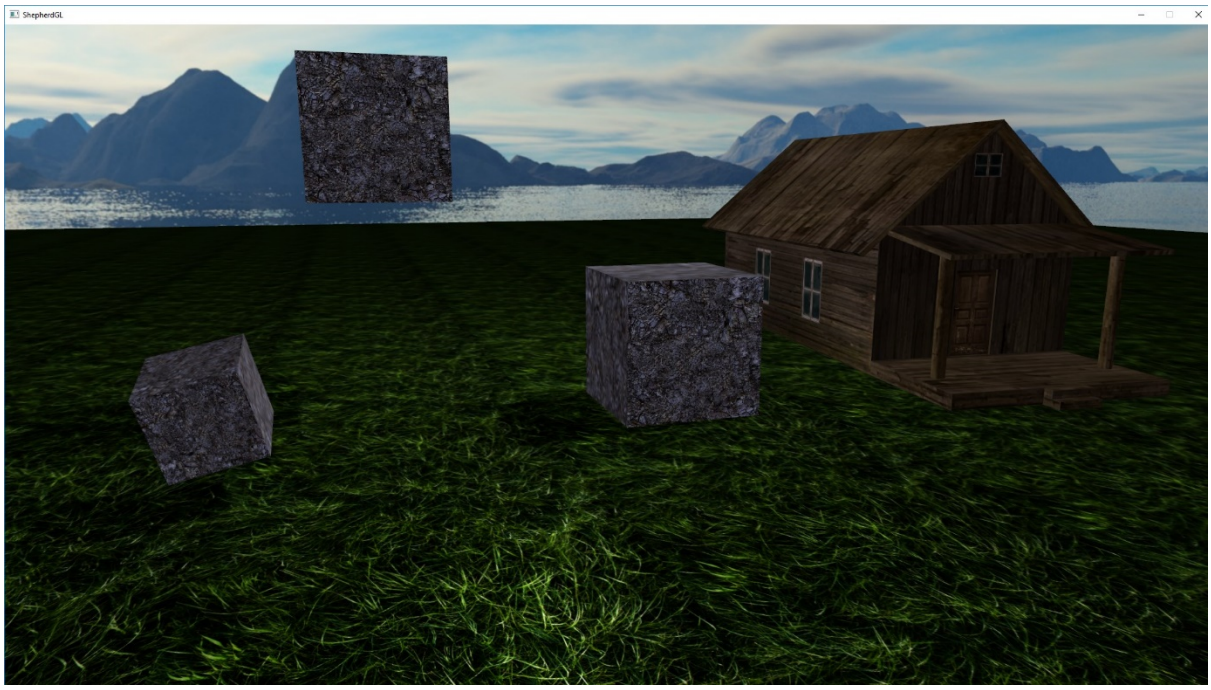
Programming for Entertainment Systems (Assignment 2)

By Jamie Shepherd (January 2015)

Introduction

For this assignment I have developed a scene with a moving light source that casts real-time calculated shadows on to a textured plane. There are a variety of ways in modern game programming to create shadows, and I have decided to use shadow mapping which is one of the simpler techniques that still looks great. The program can be executed from the Bin folder in the attached Zip (Windows 32bit).

The end product



A light source moves around the scene illuminating all of the different objects from different angles. You can freely move around the scene with the WASD keys to view the shadows being calculated in real-time. A large skybox also hosts the scene, which is a cube mapped object representing 6 individual textures. The scene renders very quickly and also includes multisampling by default.

How it works

The prototype has been developed with OpenGL and supports all versions down to 3.3 (generally considered “modern” OpenGL). With shadow mapping, the scene is actually rendered twice. First it is rendered from the perspective of the light source, with a depth map created to represent the shadows, then this is stored as a texture. The scene is then re-rendered to show the shadows on top (at certain levels of strength).

Two separate shaders are used for each render, with the depth shader just returning the positions of shadows.

In the main game loop, the time is incremented and a few mathematical calculations are used to move the light source over time. This gives the real-time dynamic shadow effect.

The application is object oriented and therefore broken into several classes, these are as follows:

- Main (the main entry point for the program)
    - Has a number of helper classes at the bottom, including those which run the input functionality
    - The main game loop is run from here
    - Rendering the scene’s 3 cubes and our custom model in RenderScene
- Model (based on a 3rd party class, which pulls a model file in with the open asset import library)
- Mesh (a representation of a mesh object OpenGL can use and understand)
- Shader (a helper class which loads our vertex and fragment shaders)
- Skybox (generates the cube, loads 6 images as a cubemapped texture)
- Camera (represents the camera object, and the view matrix which can be retrieved from the class)

3rd party libraries used (required if rebuilding)

- GLM (OpenGL Math Library) – A math library similar to DirectXMath
- GLEW (OpenGL Extension Wrangler) – Extension loading library which makes using OpenGL much easier
- GLFW – Cross platform library which allows creation of windows and contexts
- Assimp (Open Asset Import Library) – This library makes loading models from a variety of different formats much easier

Some caveats

Currently only runs on 32 bit Windows in Debug mode. It's possible and very easy to target other platforms, but new configuration files and versions of the 3rd party libraries will have to be built.

Project evaluation

Overall I think the project was very difficult. I had decided after the first assignment to use OpenGL after struggling with many compatibilities related issues with DirectX. While OpenGL definitely helped alleviate these issues, it meant learning an entirely new set of libraries, functions and classes.

Despite these concerns, I am pleased that I was able to generate real time shadows, and a first person environment that a player could potentially play in had the engine been more developed. I am particularly proud of the framework I have created, as this will allow me to develop upon the application in the future. It's clean, maintainable, and also surprisingly very performant considering it was my first foray into OpenGL. I have a Skybox implemented, which I think looks great. I also provide the ability to freely move around the scene in 1st person view, and load textures.

There are a few issues I would like to have corrected, including the ability to cap a players framerate. Right now, the application runs as fast as it can, which means I could get thousands of frames per second, but this also means the GPU is utilising a lot of its available capacity.

I would also have liked to have generated some terrain for my objects to live in, rather than a flat plane of grass. I had actually started working on this towards the end of the project, but was not able to get it working in time for the deadline.

If I was to re-take the project, I would have completed the first part of the assignment in OpenGL, giving me more time to work on new features for part 2 rather than starting a new engine framework from scratch.

Despite the shortcomings, I think learning both OpenGL and DirectX during the course of the module is a great feat in itself, and I've created great looking real time shadows which can be adapted for use in future projects.