# Introduction

The aim of this project is to render a procedurally generated cityscape. Cities are an important part of our modern world, and visualizing cities can provide valuable insights into fields such as transportation modelling and urban planning.

# Technology

This project was developed in OpenGL. I chose to develop the project in OpenGL over a raytracing approach mainly for performance – a cityscape would be significantly more useful if it can be rendered and explored in real-time.

The programming language of choice for this project was Python, using PyOpenGL and glfw for bindings to OpenGL. I chose to use Python due to my past experiences with this, which would (hopefully) allow for easier development. While Python is not quite as efficient as C or C++, most of the heavy-lifting in this project is done in the shaders. If there was additional aspects, such as simulating traffic, Python would not work as well – but this is outside the scope for the project.

For some 2D rendering, I chose to use the fantastic PIL/Pillow library. This library is used to load textures into the shaders, and also for some basic image pre-processing. For some UI controls, the imgui library proved to be invaluable in the early stages of development.

For debugging, I used RenderDoc when times got tough. RenderDoc was extremely useful for stepping through each frame and figuring out what was going wrong – this was invaluable for the beginning of the project, when nothing was displaying at all.

# Development

## Initial Setup

Before jumping into constructing a city, a lot of work has to be undertaken to setup a basic OpenGL environment. To help further my understanding of the topics covered in the course, I chose to re-implement a lot of the code instead of simply copying code used from the practicals. Unfortunately, this took significantly longer than I anticipated.

Debugging was especially a challenge – it’s very difficult to figure out what’s going on when nothing is showing up at all. At this point, I used RenderDoc to investigate what was going on, and it helped me identify several issues. RenderDoc proved invaluable to getting the basic code working.

For loading and rendering objects, I wrote a ObjLoader class based off the ObjModel code in the labs. This helped me gain familiarity with the obj file format. My approach to this helped separate loading the model and rendering it – this will prove useful when reusing the same building model many times in a large city.

## Lighting & Basic Shaders

The next step once basic rendering was done was to setup a (relatively) simple lighting system for the city. My goal with this was to setup a basic daylight cycle, allowing for the sun position and colour to change over time.

Lighting shading is implemented with a simple phong model – ambient colour, diffuse colour, and specular reflections.

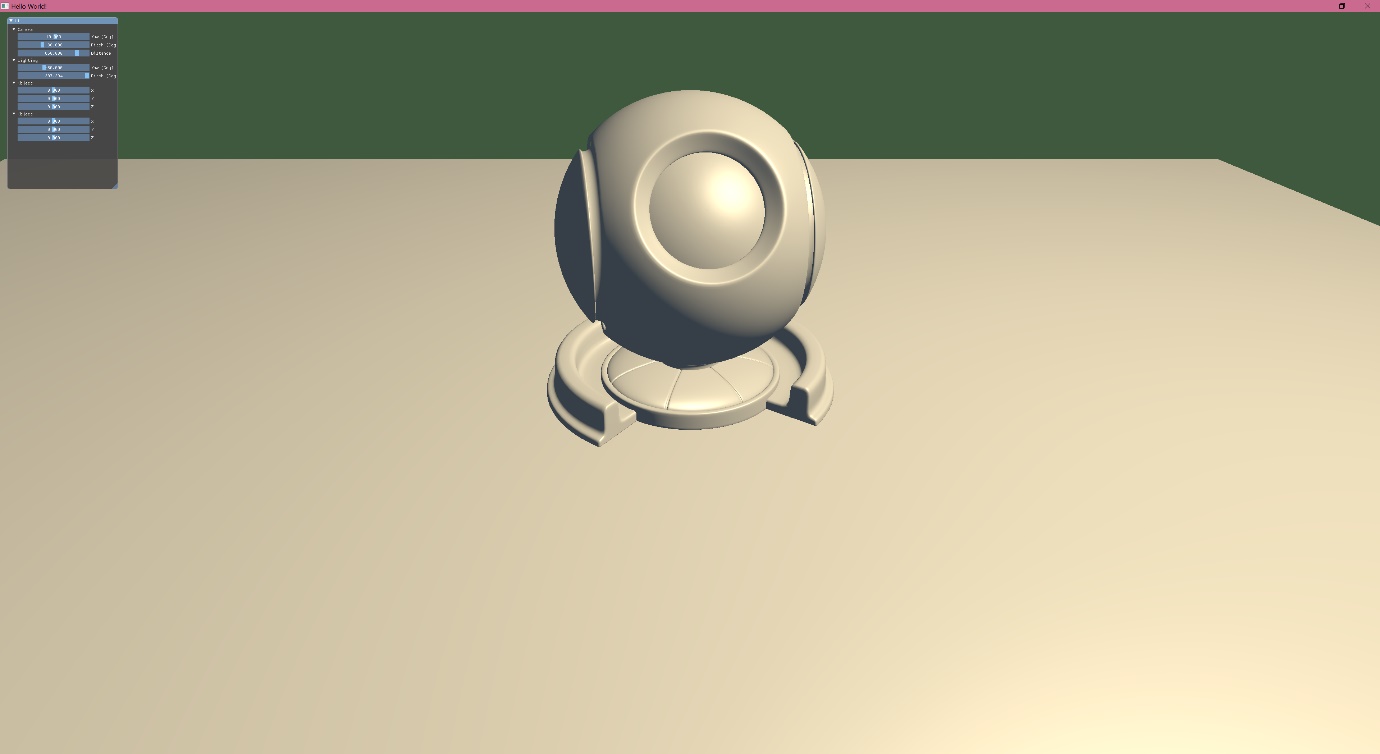


Figure 1: Example implementation of phong lighting model

For a simple daylight cycle, the pitch of the sun has to change over time, along with some different colouring based on the pitch.

I made the following simple gradient to approximate a daylight cycle:

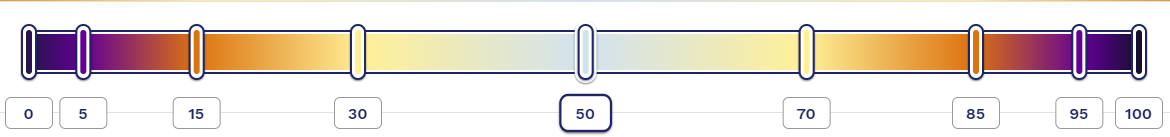


Figure 2: Example of the daylight cycle gradient

Similar gradients were implemented for ambient colour and strength to produce a decent looking daylight cycle:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Relying only on ambient light at night doesn’t work too well, so a similar lighting system was introduced with significantly less strength to simulate a moon. This isn’t a realistic effect, but it’s still interesting enough to look at while providing adequate lighting to the night environment.

|  |  |
| --- | --- |
|  |  |

## Construction Begins

The original plan for this project was for building meshes to be procedurally generated. After some investigation, I eventually decided to create the buildings by hand instead. The main reason for this was difficulty – handling everything such as face normals and UVs was too time-consuming to implement procedurally. I decided that having a small handful of individual building meshes would help populate a basic city and demonstrate the graphical capabilities of the program.

All meshes were created using Blender 2.8. I had past experience with Blender, so it was relatively easy to create the simple buildings. The main area I had trouble with was UV mapping the objects – since I’m using a single texture on all faces, I ran into some minor troubles here.

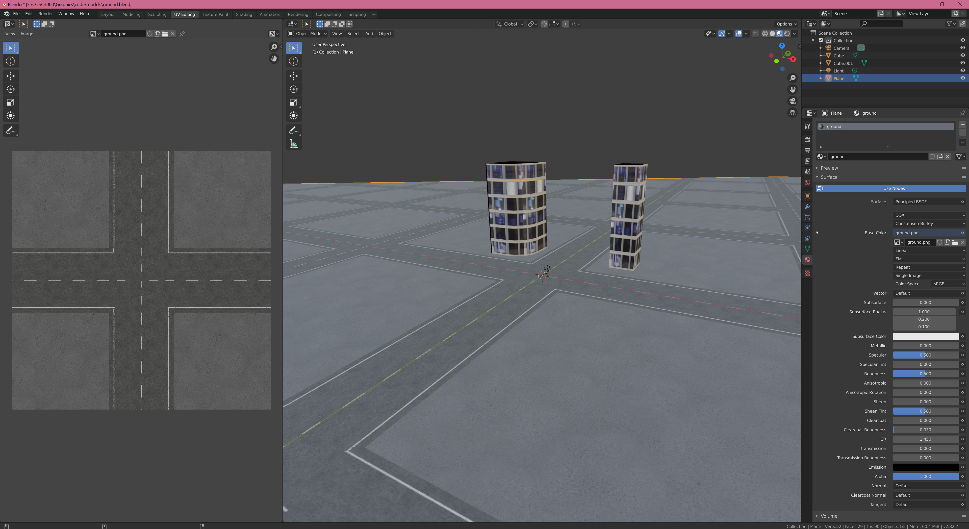


Figure : Development of the ground mesh in Blender. UV scales were carefully picked to work well with the building meshes (included for reference)

At this stage, some basic cube mapped reflections were also implemented. While the texture quality is not fantastic, the reflections add a subtle extra layer to the shading model. These cubemaps have been tied into the lighting system, and the texture changes depending on whether it is day or night. This could be expanded with further textures for more times of day.

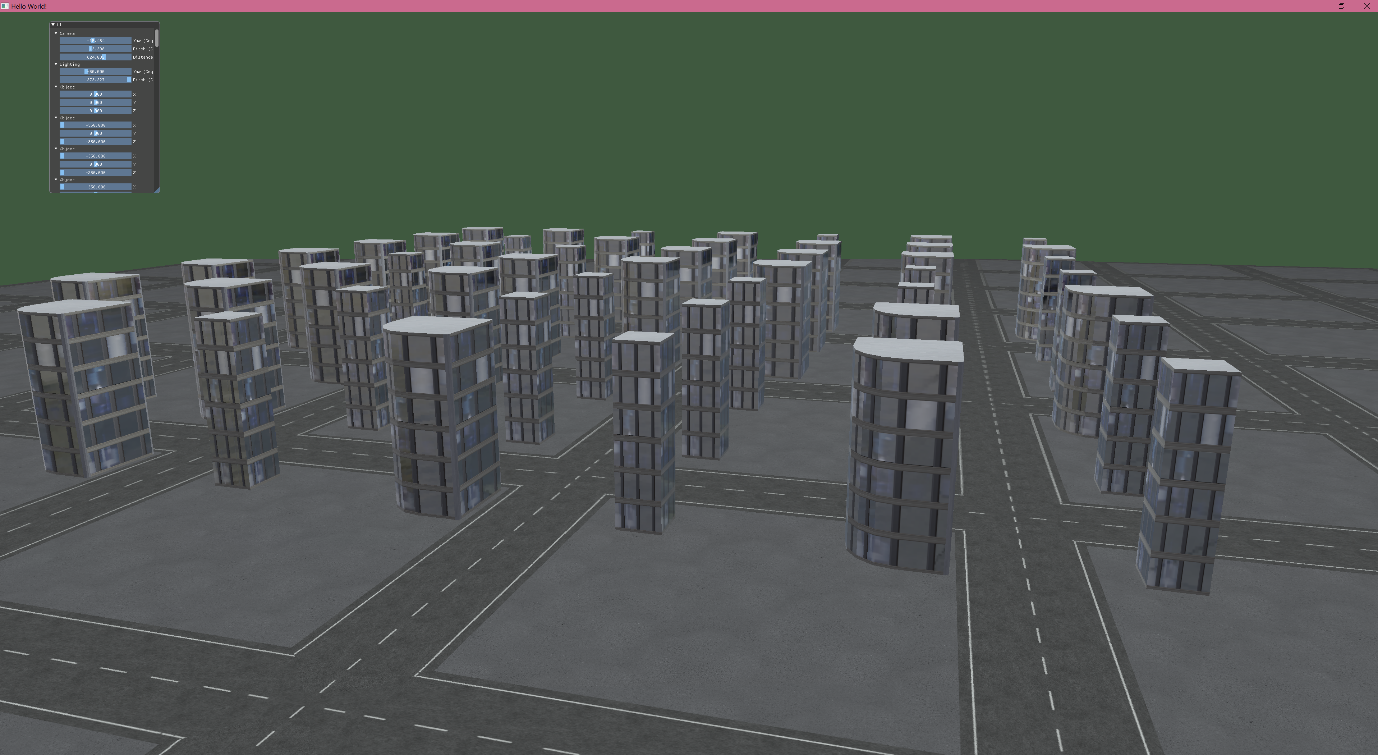


Figure : First view of a city with multiple buildings

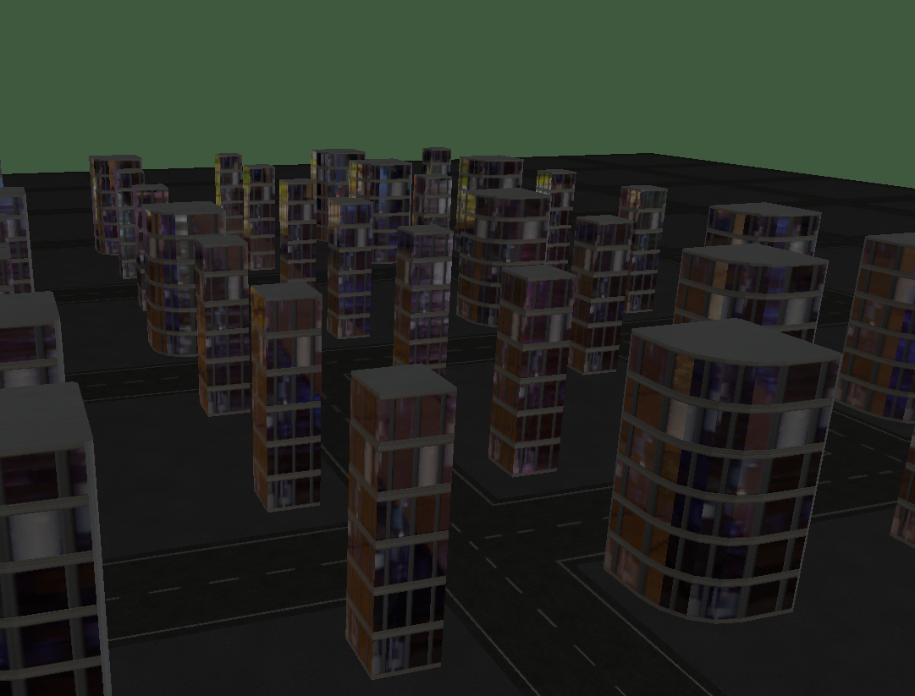


Figure : City at night, demonstrating a different cubemap texture

The city so far is very rudimentary, but this is a good showcase of the city so far. The buildings use two different shader models: a simple diffuse shader for the rooftop and other edges, and the phong shader model for the windows. Having two separate shaders for the mesh will allow for some more complex effects later as well.

## Interior Mapping

# Reflection

I’m pretty happy with how the day-night cycle turned out. Despite using a simplistic shading model, the smooth gradients for adjusting the lighting colours have a lot of impact on the environment, and the end result feels dynamic. One possibility to improve this system would be to add a dynamic skybox – having stars in the night sky would make the cityscape feel magical.

There is plenty of additional work that can be done on this project. The main step would be improving the randomness of the city – reworking the mesh system to be procedural would be a major milestone for the project. More work could also be put into the environment, such as having randomly generated terrain and a randomly generated road network.

If I was to start this project again, I would use a language such as C++ instead of using Python. I think that a project of this sort would benefit from stronger type handling, and there are more resources online for C++ implementations. I spent a large portion of this project working on the fundamentals instead of unique functionality, which has ultimately lead to a lacklustre product.