University of Queensland

COSC3000 Semester 1 2015

Model UQ
Computer Graphics Report

Thomas Cranny

42920368

Abstract

TODO

June 3, 2015

Contents

1 Introduction

The aim of the project was to construct a viable 3D visual representation of the University of Queensland's St Lucia campus. The rationale for undertaking this project was multifaceted; it is a thematic continuation of the previous data visualisation project, and it serves as a useful and relevant application of computer graphics techniques.

Following the initial proposal, the end goal of the project was to create a visually recognisable recreation of the UQ St Lucia campus. In order to accomplish this, multiple computer graphics techniques were utilised to produce the end result.

The previous data visualisation project aimed to analyse the relationships between the faculties, courses, majors, and programs offered by UQ. This project is a continuation of the UQ centric theme, however not much else is continued through into this project.

The rationale of visualising the St Lucia campus in a three dimensional model is to provide a better method of gaining a spatial awareness of the campus' layout and structures. This stems from recalling the unhelpfulness of most UQ maps when trying to navigate the campus for the first time. Due to the odd shapes of many buildings, the presence of many annexes and similarly labelled and named buildings, the ambiguity between roads, walkways, and viable pathways, and the dense arrangement of buildings, sufficiently learning the layout of the campus may even take weeks.

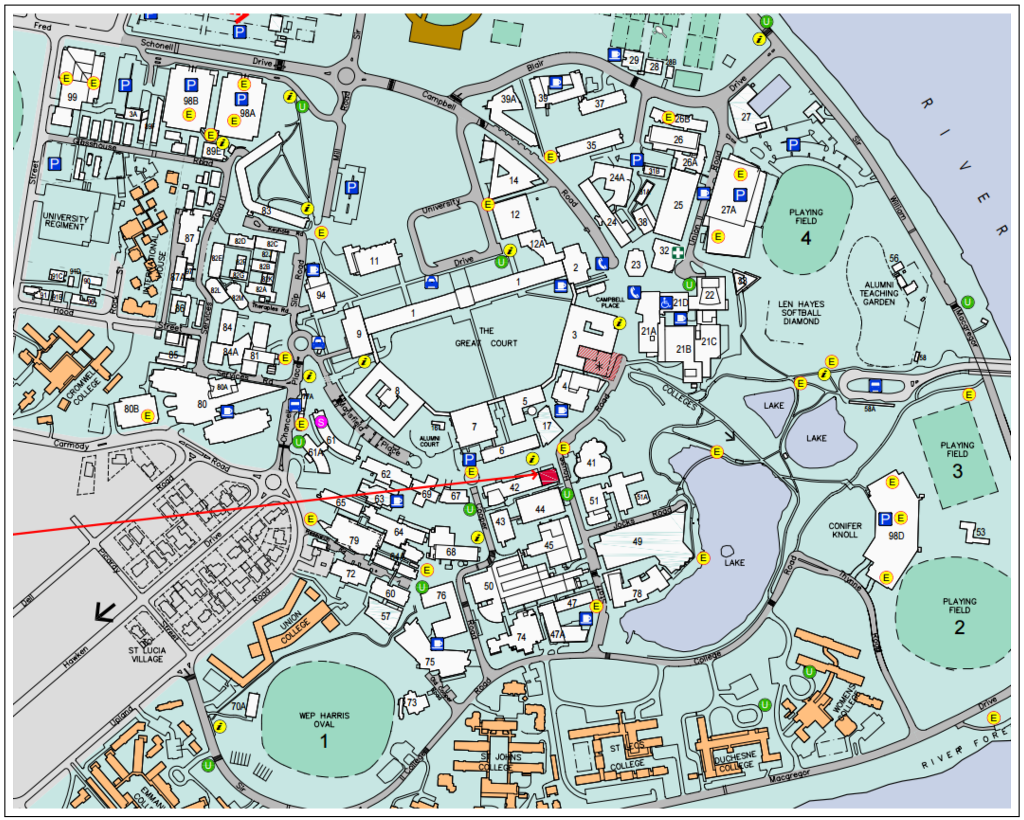

Figure 1: Official University of Queensland St Lucia map [2]. Note the densely packed, oddly shaped and often conjoined layout of the buildings.

Static maps, such as in Figure 1, can have high technical accuracy, but are often convoluted and typically don't convey a good physical model of an environment to a reader. Abstract two dimensional maps typically require a lot of cross checking between the map and the physical surroundings, in order to correlate real landmarks with what the map displays. If structures and landmarks could be better represented to correlate with their real life counterparts more realistically, users will be able to visualise and integrate a locale much more quickly and effectively than attempting to correlate an abstract diagram to their surroundings.

2 Methods

2.1 Technical Overview

The project was implemented using the graphics capabilities of modern browsers, primarily through the three.js JavaScript library, and GLSL shaders. The main features incorporated include:

- Mouse interactivity.
- Texture and bump mapping.
- Dynamic model creation.
- Lighting and shadow casting.
- Animation.
- Constructive geometry.
- GLSL shaders.

2.2 Graphics Environment: WebGL and Three.js

The project was done using WebGL, an implementation of the OpenGL specification used with in modern web browsers. WebGL is designed as a rendering context for the HTML Canvas element [5]. The rationale for using WebGL was based around the portability of web pages, and the ubiquity of web browsers. Equivalent OpenGL projects written in C have more complex and fragile requirements, such as a compilers, library headers, and platform specific binaries; and Python requires similar dependencies and suffers reduced performance in addition.

In order to access the browsers rendering capabilities, the *three.js* JavaScript library was used, as it is well documented, and has an extensive corpus of examples online. It provides a programming environment well suited for graphics programming in JavaScript. While JavaScript is more object orientated than the venerable C bindings for OpenGL, it maintains a close equivalence to native GL instructions, with a cleaned up API and astute use of flexible JavaScript objects[1].

2.3 Features

2.3.1 Mouse interactivity

A major usability feature of the application is the implementation of the so called 'Orbit' controls, which allow users to pan, zoom and rotate the camera around the scene naturally using a mouse. Mouse interaction is facilitated by registering event callbacks for the main canvas DOM[2] element within JavaScript to fire for different input methods. For example, a function can be registered for a mouse-drag event; should a mouse drag event occur, that function can determine how far the mouse's position has changed with respect to the x and y dimensions, and can manipulate the

[1]'*JSON*' is also often used, it stands for JavaScript Object Notation, and is a useful data serialisation format

[2]Document Object Model, the tree of HTML elements that comprise a webpage.

cameras position to simulate a panning motion.

Earlier versions of the project had a this implemented simply, however later mouse controls were delegated to the better performing and more natural ‘Orbit Controls’ library [12], which provides simple configuration for $y+$ (always vertical) camera controls, including panning, zooming, and rotating (Added in commit `37bc245`, see reference [1]).

2.3.2 Texture and bump mapping

Textures were used for most of the geometry for the scene in different ways. This includes the buildings, ground, water, and the skybox.

The Ground has the simplest texturing in the scene. It has a grass image applied to it, using repeat S and T wrapping to create a continuous “grassy” surface. A fairly low specular value was used, due to it’s lack of reflectivity in real life.

The Buildings in the scene have both a texture and a bump map applied. The bump map is of a simple tessellating brick pattern, and the texture is an image of varying sandstone patterns. The sandstone textures were created by assembling sections of photographs taken on campus specifically for this project.

The combination of sandstone and a brick bump map is obviously utilised to simulate the actual sandstone bricks that buildings at UQ are famously constructed from.

The Skybox utilises six textures of a cloudy sky, projected onto the interior of a large cube surrounding the scene in order to produce a realistic sky effect. A special purpose built-in shader is used to handle flipping the normals of the skybox cubes face, so the inside surface of the cube is visible instead of the outside. The shader also serves to illuminate the sky textures so they are still visible at great distances, and map the images to the correct location and rotation so that it appears seamless.

The Water of the UQ lakes and Brisbane River utilise a texture of caustic water map. The water however is a special case, as the texture is processed and manipulated via a GLSL shader, which provides a subtle rippling or wavy effect on the water surface. See 2.3.7 below for more information on the shaders behaviour.

2.3.3 Dynamic model creation

In order to specify and generate the buildings and water bodies of the model efficiently, a dynamic approach to model initialisation was taken. The process involves defining an arbitrary set of anticlockwise points for the “footprint” of each building and lake. This information is stored in the `data.js` JavaScript file, and also defines for each building the position, rotation, name, building number, and height.

In the case of buildings, each in the dataset is iterated, and a two-dimensional shape is constructed from the array of footprint points. Then, that polygon is extruded vertically by a factor of that buildings height.

This allowed for moderately fast input of the buildings and water bodies into the model, as only a position, height (for buildings), and small set of points was required to define each feature.

The initialisation of the water bodies is more complicated, as first geometry is subtracted away from the terrain to allow room for a water surface to appear below the level of the terrain, and then a shader is initialised to produce the rippling water effect on the water surface meshes. See 2.3.6 and 2.3.7 for further information on the geometry and shaders respectively.

2.3.4 Lighting and shadow casting

2.3.5 Animation

2.3.6 Constructive geometry

2.3.7 GLSL shaders

3 Results

Here be dragons!

4 Discussion

Here be dragons!

5 References

[1] Thomas Cranny *Computer Graphics Project - source code repository*
Accessed online at: https://github.com/kamikai/cosc_project2
On 3/May/2015

[2] The University of Queensland *St Lucia Map*
Accessed online at: https://www.uq.edu.au/maps/pdf/StLuciaMap.pdf
On 1/June/2015

5.1 Technical Resources

[3] Threejs.org *Online Documentation*
Accessed online at: http://threejs.org/docs/
On 1/June/2015

[4] Threejs.org *Online Examples*
Accessed online at: http://threejs.org/examples/
On 1/June/2015

[5] Khronos Group *WebGL Specification - Version 1.0.3*
Accessed online at: https://www.khronos.org/registry/webgl/specs/1.0/

[6] Jerome Etienne *Let's do a skybox* - August 2011
Accessed online at:
http://learningthreejs.com/blog/2011/08/15/lets-do-a-sky/
On 1/June/2015

[7] Roman Liutikov *Skybox and environment map in Three.js* - August 2013
Accessed online at:
http://blog.romanliutikov.com/post/58705840698/skybox-and-environment-map-in-three-js
On 1/June/2015

[8] Jerome Etienne *Casting Shadows* - January 2012
Accessed online at:
http://learningthreejs.com/blog/2012/01/20/casting-shadows/
On 1/June/2015

[9] Jerome Etienne *Constructive Solid Geometry with csg.js* - January 2012
Accessed online at:
http://learningthreejs.com/blog/2011/12/10/constructive-solid-geometry-with-csg-js/
On 1/June/2015

[10] Lee Stemkoski *Animated Shaders in Three.js* - May 2013
Accessed online at:
http://stemkoski.blogspot.com.au/2013/05/animated-shaders-in-threejs-part-2.html

[11] Paul Lewis *An Introduction to Shaders* - April 2012
Accessed online at:
https://aerotwist.com/tutorials/an-introduction-to-shaders-part-1/

5.2 External Libraries

[12] Threejs.org *Orbit Control Example*
Accessed online at:
http://threejs.org/examples/misc_controls_orbit.html
On 1/June/2015

[13] Evan Wallace *Constructive Solid Geometry*
Accessed online at: http://evanw.github.io/csg.js/
On 1/June/2015

[14] Ricardo Cabello *Performance Statistics Monitor*
Accessed online at: https://github.com/mrdoob/stats.js/
On 1/June/2015

5.3 Art Assets

[15] OpenGameArt.org - *Skybox Textures*
Accessed online at: http://opengameart.org
On 1/June/2015

[16] Threejs.org - *Grass Texture*
Accessed online at: http://threejs.org/examples
On 1/June/2015

[17] Thomas Cranny - *Sandstone texture & bump maps*
Source images taken May 2015