

6-16-2016

Explorable 3D Model of SCU Campus

Benjamin Giglione
Santa Clara University

Follow this and additional works at: http://scholarcommons.scu.edu/cseng_senior



Part of the [Computer Engineering Commons](#)

Recommended Citation

Giglione, Benjamin, "Explorable 3D Model of SCU Campus" (2016). *Computer Science and Engineering Senior Theses*. Paper 57.

This Thesis is brought to you for free and open access by the Student Scholarship at Scholar Commons. It has been accepted for inclusion in Computer Science and Engineering Senior Theses by an authorized administrator of Scholar Commons. For more information, please contact rscroggin@scu.edu.

SANTA CLARA UNIVERSITY

Department of Computer Engineering

I HEREBY RECOMMEND THAT THE THESIS PREPARED
UNDER MY SUPERVISION BY

Benjamin Giglione

ENTITLED

EXPLORABLE 3D MODEL OF SCU CAMPUS

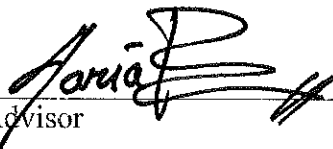
BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

BACHELOR OF SCIENCE

IN

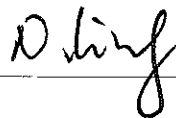
COMPUTER SCIENCE AND ENGINEERING

Thesis Advisor



6/14/2016
date

Department Chair



6/15/2016
date

Explorable 3D Model of SCU Campus

By

Benjamin Giglione

SENIOR DESIGN PROJECT REPORT

Submitted to
the Department of Computer Sciences and Engineering

of

SANTA CLARA UNIVERSITY

In Partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Science and Engineering
School of Engineering

Santa Clara, California

June 16, 2016

Table of Contents

Abstract	1
Chapter 1 – Introduction	
1.1 Problem Statement	2
1.2 Requirements	4
1.3 Related Works	5
Chapter 2 – Software	
2.1 Use Cases	7
2.2 Technologies Used	7
2.3 Design Rationale	10
Chapter 3 – Analysis	
3.1 Risk Assessment	12
3.2 Aesthetics Analysis	13
3.3 Ethics Analysis	16
Chapter 4 – Development and Testing	
4.1 Application Development	19
4.2 Demonstration of Final Project	20
4.3 Testing and Bugfixes	20
4.4 Procedure	21
4.5 Conclusions	28
Chapter 5 – References	
5.1 Literature Review	30
Chapter 6 – Appendix	
6.1 Screenshot of the campus model in JOSM, overlaying the Bing map	32
6.2 UE4 Project setting to run the SCU Level as a VR Preview App	32
6.3 Oculus Desktop App set to Enable VR in Unknown Sources like UE4	33
6.4 The SCU Landscape Texture’s Nodes in the Material Graph Editor	33
6.5 Header file generated from the project’s First Person Blueprint	34

List of Figures

1 Google Maps Street View Screenshot from the perspective of an Oculus	3
2 Use Cases Diagram	7
3 Early Work in Progress screenshot, from OpenStreetMap	9
4 Senior Design Conference Work in Progress screenshot of Swig Hall in Blender . .	9
5 Final Project screenshot of Swig Hall in Unreal Engine 4.11.2	10
6 Bing Aerial Map of SCU Campus used to texture heightmap	20
7 Heightmap of Campus generated from OSM2World OBJ	21
6.1 Screenshot of the campus model in JOSM, overlaying the Bing map	32
6.2 UE4 Project setting to run the SCU Level as a VR Preview App	32
6.3 Oculus Desktop App set to Enable VR in Unknown Sources like UE4	33
6.4 The SCU Landscape Texture's Nodes in the Material Graph Editor	33

List of Tables

1 Risk Table	12
2 Aesthetics Analysis Table	13
3 Ethics Rationale Table	16

Abstract

My project is an interactive 3D model of SCU campus, which prospective students and their parents can explore without actually having to make the journey to campus. The architecture of the university is made traversable by running under Unreal Engine 4 which is a 3D game development framework that supports the Oculus Rift. The Oculus Rift is a virtual reality headset, which enhances the immersive experience for users of the SCU campus application. It accomplishes this by displaying the rendered images in immersive 3D right in front of their faces, and tracking their head motion and moving the viewpoint in the virtual world accordingly, so it will be as if they were actually there. Oculus Rift compatibility is fully integrated into the Unreal Engine, so it's only natural to take advantage of the technology for this project.

Chapter 1 – Introduction

1.1 Problem Statement

Prospective incoming freshman and their parents need to get a good look at a college's campus before they decide whether it's a sounder investment than the other universities they've been accepted into. Entire families often go miles out of their way to take a tour of the school primarily for this purpose. It could save a significant amount of time and money, and show them the true beauty of the campus, if they had the experience of touring the university before actually going there. Unfortunately, SCU's current options on that front are limited.

Anyone can use a labeled map to take a look at SCU's campus, but it can't give them the same kind of feeling as actually being there. Currently, the only technology that simulates a first-hand exploration of the school's grounds is Google Street View, a feature of Google Maps that allows the user to browse select areas from a grounded first-person perspective. However, even with Street View, you only get a series of stationary panoramas, which is far from the same experience of exploring the university on foot. There are even some websites that will take Google's 3D imaging data and output a 3D Street View on the Oculus Rift (Figure 1), a developing technology that can project 3D data to lenses right in front of your eyes as if you were actually there, and internal sensors that know when you're tilting your head so it can tilt the 3D worldview in kind. Although that's a step in the right direction, the underlying service still does not offer the kind of mobility one should expect from an accurate emulation of an area.

I propose we remedy this lack of a decent campus simulation by building a full 3D model of the campus and deploying it in a game engine to enable interactivity and cross-platform support. The interactive model will include most of the common structures of SCU's campus, and cover as much ground as possible with time constraints. My project will make use of the Unreal Engine, and will be constructed using Unreal Engine 4 (citation 3). With a video game-type application in the Unreal Engine, users will be able to use their keyboards to walk around the campus in a fully realized three-

dimensional environment, which will allow them to view the school at every possible angle and position. The Unreal Engine also natively supports the Oculus Rift VR, which will do one better than the standard computer screen by allowing the user to look around the campus in 3D as if they were inside the school. With this leap in virtual reality services, parents and students alike will be impressed by such a convenient and well-executed simulation of the area, as well as by the architectural beauty of Santa Clara University.

Figure 1. Google Maps Street View, from the viewpoint of an Oculus:



1.2 Requirements

Functional Requirements

Users must be able to do the following:

- Travel around in an accurate model of SCU campus that encompasses all of its buildings to scale
- Wear the Oculus VR to view the scene in 3D and look around as if they're actually there

Non-Functional Requirements

The final project must be:

- Portable on multiple platforms
- User-friendly
- Straight-forward
- Easy enough for prospective students and their parents to understand
- Well-designed

1.3 Related Works

Qing Wang; Xijuan Zhu, "The implementation of campus 3D electronic map based on SketchUp and ArcGIS," *Audio Language and Image Processing (ICALIP), 2010 International Conference on* , vol., no., pp.1031,1034, 23-25 Nov. 2010

Yangtze University students created a basic 3D campus with the ArcGIS Engine and Google SketchUp. Although the authors emphasized the speed and efficiency of their method, the model they pictured looks wooden and low-quality.

Bin Chen; Fengru Huang; Hui Lin; Mingyuan Hu, "VCUHK: Integrating the Real into a 3D Campus in Networked Virtual Worlds," *Cyberworlds (CW), 2010 International Conference on* , vol., no., pp.302,308, 20-22 Oct. 2010

The CUHK students modeled over 100 buildings of CUHK campus in Second Life. They also gave the example of NTU, who have done the same thing. The Second Life rendering reminds me of old CGI from the 90s like ReBoot and Myst, but the geometric accuracy of the VCUHK campus compared to the real CUHK campus is impressive.

Wu Fenghua; Chen Guangzhao, "Virtual 3D Campus Design and Implementation," *Intelligent Computation Technology and Automation (ICICTA), 2010 International Conference on* , vol.3, no., pp.1136,1139, 11-12 May 2010

Hebei Polytechnic University students used 3DS MAX 8 to model the campus, in conjunction with GIS and VR-Platform. Their campus model looks fairly decent, but the resolution of the sample images provided is pretty low, so it's hard to make out the finer details. Nonetheless, the models provided are a good example of the balance of quality and accuracy I'm aiming for in my project.

Zhang Shuai; Tan Guoxin; Liang Bo; Hu Fanggang, "Design and Implementation of Real-Time 3D Campus Scene Simulation Management System Based on Vega," *Computer Science and Software Engineering, 2008 International Conference on* , vol.2, no., pp.1162,1165, 12-14 Dec. 2008

This project denotes the results of a Chinese Computer Engineering team's efforts to design and implement a virtual campus using Vega. The team made the 3D models for the campus in Maya and Creator, and rendered them in the Vega API. The final images produced seemed a tad flat but overall impressive and indicative of a real college campus.

Ke Yu; Zhuo shi; Hong-yan Yang, "3D Virtual campus based on VR-Platform," *Consumer Electronics, Communications and Networks (CECNet), 2011 International Conference on* , vol., no., pp.2848,2851, 16-18 April 2011

High-echelon computer engineers of Guilin University of Electronic Technology described how they incorporated a 3D campus based on Google Earth images and modeled in 3DS MAX into a Virtual Reality Platform. Grated, I couldn't understand most of it; almost all of the text was in Chinese besides the abstract, but the renders of the campus they created looked sound. The framing device they used for those 3D views felt quite hokey to me, but perhaps that's just the fashion in the country.

Chapter 2 – Software

2.1 Use Cases

A user must be able to perform the following actions within the application:

- Move the character throughout the virtual environment
- Adjust the character's camera view to any 3D angle (Figure 2)
- Switch between the 2D and 3D modes if the user has an Oculus Rift

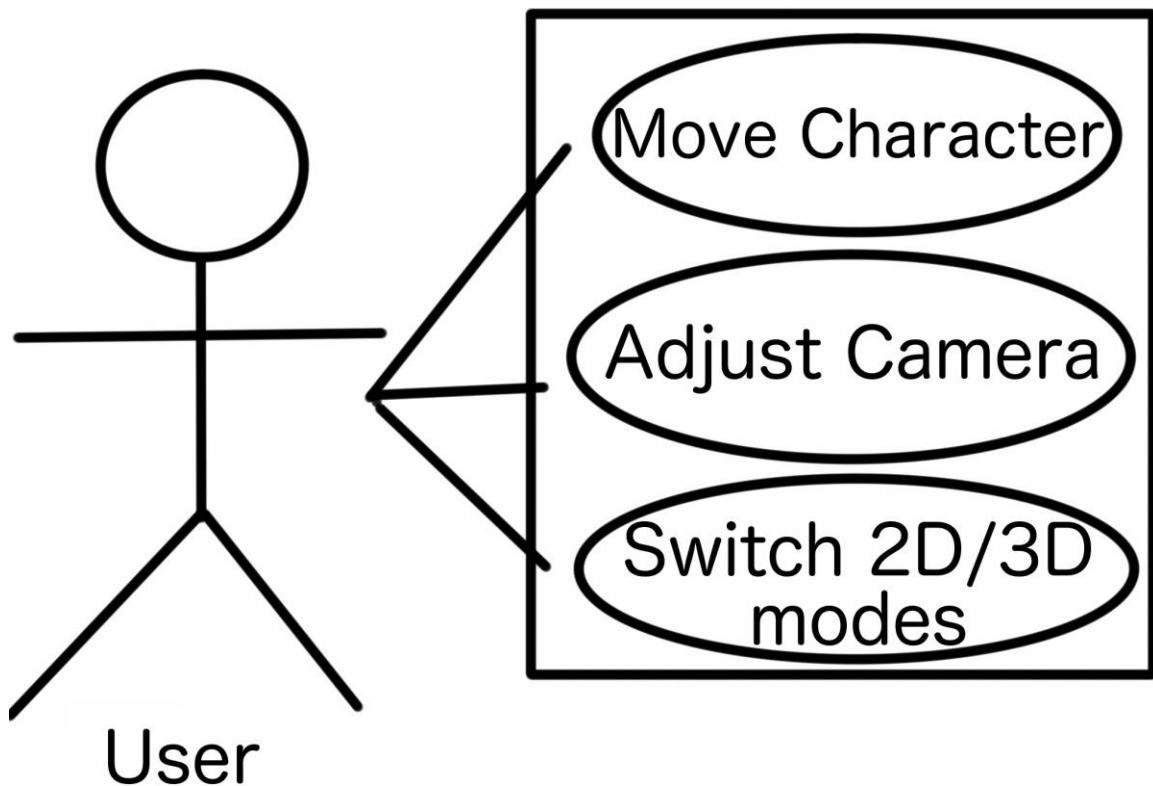


Figure 2. Use Cases Diagram

2.2 Technologies Used

Unreal Engine 4:

- Unreal Engine 4 (UE4) is a popular multi-platform game engine
- The engine and its Source Development Kit (level editor) are free
- UE4's editor runs on Mac OS X, but UE3's (UDK) is Windows-only
- Its high-end graphics are beautiful and state-of-the-art
- Unreal Engine 4 has built-in support for the Oculus Rift

Oculus Rift DK2:

- The Oculus Rift Development Kit 2 was Oculus's latest VR headset.

- It shows video game levels in true 3D with head-tracking
- The previous model, the DK1, was blurry and caused motion sickness
- DK2 has higher resolution and lower-latency head-tracking to fix this

Blueprint:

- Blueprint is Unreal Engine 4's event and sequencing tool
- A visual programming language that can customize 3D levels of UE4
- Perfect for setting up the player's interactions with the environment
- Much better suited for creating reusable behaviors than UE3's Kismet

3D Modeling:

- OpenStreetMap is an open source 3D world map (citation 1)
- Google Earth Pro is an extensive proprietary 3D model of Earth (cit 2)
- JOSM is a Java-based editor for OpenStreetMap (OSM)
- Blender is a free 3D modeling program. Features:
 - Quick Projection - Use a 2D image to edit a 3D model
 - Export model to FBX and heightmap (elevation map)
- 3DS Max is a paid modeling program with free licenses for students
- Also features FBX and heightmap export, with better FBX support

2D Textures:

- Bing Maps is a satellite mapping service allowed to be used in OSM
- tile-utils is an open source GitHub project that generates aerial maps
- It downloads up to 200 Bing Maps tiles and stitch them together

My Contribution:

Using modeling approaches, I accomplished the following:

- Used OpenStreetMap's model of SCU campus as a base (Figure 3)
- Used Google Earth Pro to approximate the heights of buildings
- Set all the school's buildings to their proper heights using JOSM
- Imported the model to 3DS Max by exporting w/ OSM2World (Fig 4)
- Exported a heightmap of the model with 3DS Max to 16-bit grey PNG
- Used the tile-utils program to download an 11x11 square aerial map of SCU
- In UE4, I imported the 8161x8161 resolution heightmap as a terrain
- Using Blueprint, I applied the aerial texture to the landscape terrain

Figure 3. Early Work in Progress screenshot, from OpenStreetMap:

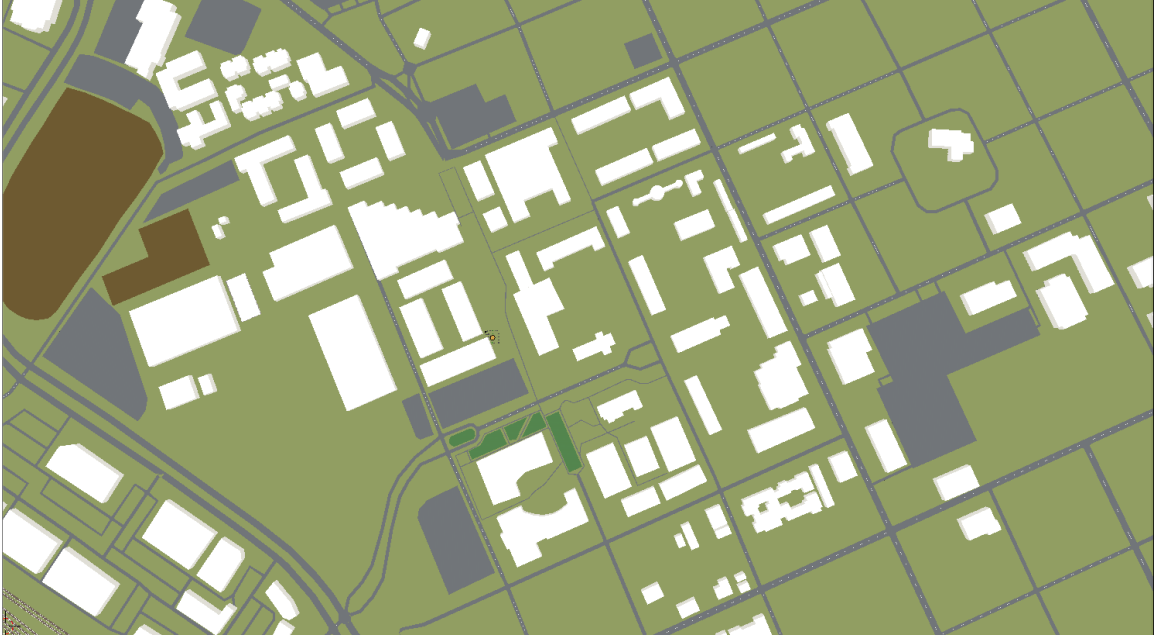


Figure 4. Design Conference WIP screenshot of Swig Hall in Blender:

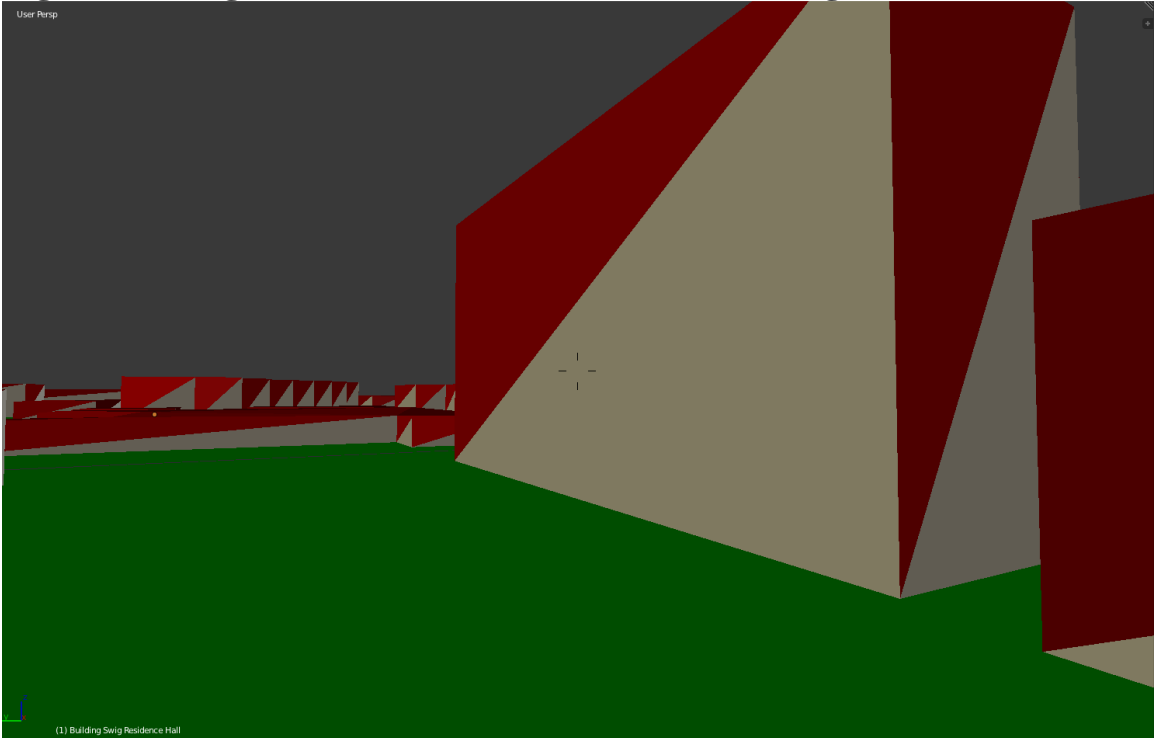
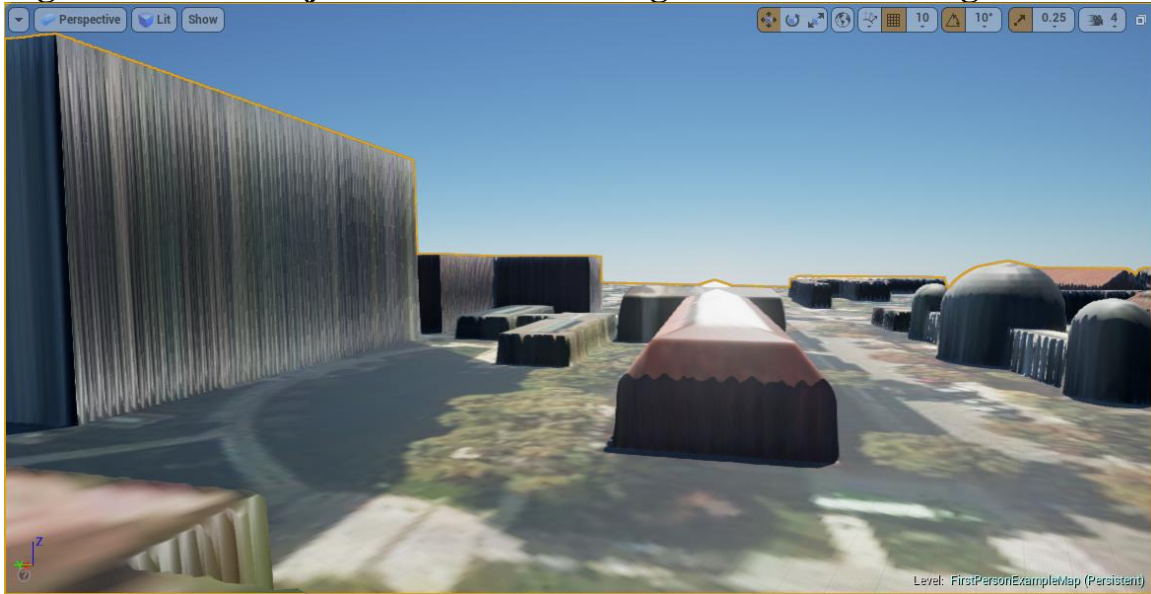


Figure 5. Final Project screenshot of Swig Hall in Unreal Engine 4:



2.3 Design Rationale

The 3D engine the model of campus runs under is Epic Games' Unreal Engine 4. An End User License Agreement applies, but the basic terms of it dictate that 5% of the gross revenue of any application utilizing the engine must be paid to Epic as royalties. However, Unreal Engine 4 does not require payment for non-profit and academic use (nor does Epic require royalties for any program that does not meet a minimum threshold for sales). The explorable campus application uses the Oculus Rift for advanced head-tracking and three-dimensional viewing. The Oculus Rift is the main product offered by the Oculus VR company (which developed the Oculus Rift).

My project is consistent with the Strategic Vision of Santa Clara University in the following ways. The project excels in competently educating men and women in the structure and scope of the University in full. The project gives them an integrated education in the Jesuit tradition by accurately depicting the Mission of Santa Clara and the various religious signs and imagery that can be found all over campus. The project makes a commitment to incoming students as persons by helping them get familiar with the college they'll be studying in before they even get here, and helps them navigate the area during their stay; preparing them for professional excellence.

My project is consistent with the Mission Statement for the School of Engineering in that the scholarly activities I am conducting will benefit the school's various constituencies by attracting more applicants with a fully functional inside look at the beauty of the campus. The project makes full use in advances in the state of the art of stereoscopic viewing and motion tracking technologies.

Chapter 3 – Analysis

3.1 Risk Assessment

Risk	Consequences	Mitigations
New Requirements	Project is unacceptable	Demo project before it's due
Incapacitation	Development falls behind	Get work done ahead of time
Time runs out	Desired functions incomplete	Prioritize essential features
Bugs and Errors	Find source of problem	Testing and Debugging
Don't understand how to use dev kit	Spend days relearning development kit features	Look through preexisting UE4 tutorials and projects to help
Loss of Data	Must redo weeks of work	Use GitHub or Google Drive and keep multiple copies

Table 1. Risk Table

3.2 Aesthetics Analysis

The creative design process for my senior design project inherently involves a distinct aesthetic component, and although the functionality of the final project is certainly the prime objective of any technical design, the sheer elegance in the simplicity of how this is achieved does matter. A description and analysis of the aesthetic elements in my senior design, explaining the rationale for choices made on my Senior Design Project, which is a 3D representation of SCU campus made explorable in the Unreal Engine 4 and utilizing the Oculus Rift, is as follows.

I decided on the Unreal Engine when choosing which engine to use for my project after considering a number of different engines both before and during the course of this assignment. The main decision to make Unreal my go-to game engine came in early February after reading some news updates from the team of Starry Expanse, a project to develop a 3D remake of Cyan Worlds' hit video game Riven, one of many such video game remakes I follow. They had previously been utilizing the Unity game development platform as their go-to game engine, and so had I, but they simply couldn't keep up with the cost of constantly paying to upgrade the engine and buying a license for each version. Cyan, who themselves had released a Unity remake of Riven's prequel that same month, had suggested the Unreal Engine as a powerful, cost effective alternative to Unity, as they were using Unreal Engine 4 for their latest production. The Starry Expanse team concurred that it was cheaper, faster, and prettier than Unity, and haven't looked back. I was familiar with the engine, having myself been part of a modding project for Deus Ex, which used Unreal Engine 1, and I found it a welcome change.

I also looked at the engines other projects utilized to make virtual campuses, and out of all the Chinese teams who documented their endeavors, the Second Life campus was the one I found the most impressive. Second Life is a decent and well-used platform, as evidenced by the multitude of Second Life programs I have found on the Library's computers over the years, but the rendering is simplistic and lacks the aesthetic beauty that myself and others have found in the Unreal Engine. Also, Second Life has a much smaller user base than it had in its heyday of popularity and

Second Life technical development now proceeds at a much slower pace compared to Unreal Engine and Unity.

My project demonstrates simplicity in that it is a very basic concept, putting SCU into the virtual world for all to see, although the means in which I carry out this task is quite intricate. My project demonstrates elegance in that it is stylish and modern in its implementation, and using cutting-edge visualization software, in order to achieve its goals. My project demonstrates balance in that on one hand, the project needs to look like SCU to a wide degree of detail, while on the other hand understanding that there are limits to the degree of accuracy one man can accomplish for an entire miles-wide area without exhausting oneself; the project must be just accurate enough for people to accept it as the campus without it being too difficult to accomplish within the project's limited timeframe. My project demonstrates unity in that it unites reality and software in a fabricated representation of a very real place made real by the marriage of modeling and rendering software, computing and virtualization hardware, and the human brain. My project demonstrates symmetry in that it is composed of similar, uniform parts; a spatially distributed assortment of virtual structures all in proportion to one another working together to form one big collective environment, just like the school.

My design is influenced by formal, aesthetic, "look and feel," and usability conditions to the extent that it had serious weight on the final product. The need I have imposed for the public to truly experience the look and feel of the model of campus as if they were actually at the real place has prompted my use of an Oculus Rift to enhance immersion. Feel and usability conditions have also prompted my push for acquiring Oculus Rift DK2 (Development Kit 2) hardware for the project, because it has higher resolution and higher fidelity tracking than the previous edition of the Oculus and the DK2 version of the Rift has more refined oscillatory systems which greatly reduces the causes of motion sickness, because I don't want users to feel dizzy and nauseous when they're trying to immerse themselves in the experience of augmented reality.

My project's design is primarily functional, with little need to consider formal, aesthetic issues in the programming and engine side of things; it's more about getting the rendering layer operational and making

sure Unreal acts appropriately. It is formal on the visual depiction of campus and the modeling thereof; it is important to know how detailed the campus-wide simulation must be in order to please potential applicants.

My project's design extends functionality into formality in the sense that in fulfilling the function of going to a campus without actually being there has required me to stand on ceremony in going about it, and rigidly and accurately representing the time-honored traditions that went in to SCU's design, such as the religious imagery and its significance.

My project's design extends formality into functionality in the sense that while fulfilling the strict set of social rules that apply to the audience's area of influence, it demonstrates and carry outs its function of depicting the university in a way that people will enjoy and recognize on a personal level.

Thoughts about aesthetics influence my design and improve the final functionality and appeal of the product in that the need to be aesthetically pleasing has required serious effort on my part to capture the true beauty of the campus. Being aesthetically pleasing has been hard for me, mainly because I am a group of one and only have one human opinion to offer over what that is, when it requires many people to capture a broad range of what the audience will see as artistically pleasant, but with extra effort and testing I believe it can be accomplished.

3.3 Ethics Analysis

I am making this project because I want to work with a game engine in such a way that it would benefit the school and the general public. More relevantly, I'm making this project so people who want to tour the school as if they were actually there can do so without actually being there, particularly for those of whom travelling to the school would be a long trip and thus a major inconvenience. Additionally, the virtual tours could potentially show more areas of the campus than real tours, which would help these people to get a better lay of the land. That makes me fundamentally ethically justified in what I am doing because it would significantly help Santa Clara University and anyone who is looking at going there, and it would not inconvenience anyone to do so.

What this project teaches me about the character of an engineer is that a good engineer must possess the determination to make good on his or her promises and see them to the end, even through a long development cycle. A good engineer must be punctual, and not be late with his or her meetings, presentations, and assignments. A good engineer must be diligent, and avoid putting assignments off until the last minute just because he or she can, and pace himself or herself in working on a well thought out time schedule. Most importantly, a good engineer must have ethical integrity, and put people's safety above all else. That means avoiding thinking of one's projects as a businessman would just because one finds oneself pressured, because it is not one's place to do so, and such thinking could cost people their lives, as it did in the Challenger disaster.

The specific engineering ethics challenges regarding safety and risks raised by the project itself are primarily limited to the scope of the VR hardware's side effect of sometimes causing motion illness in its wearer. I have to do my best to make sure this technology does not cause unwanted harm as a result of such seasickness. So the first specific ethical challenge I'm faced with are ensuring the Oculus causes as little dizziness as possible, which I have hopefully accomplished by securing the necessary funding to obtain the Oculus DK2 with reduced motion blur. The second challenge is determining which target demographics could suffer more severe trauma or

ailment as a result of this motion sickness, and advise these groups against using the VR headset functionality of the application.

I know my team is acting ethically in relation to Santa Clara University, my advisor, and the discipline of Computer Engineering I must work with to achieve my goals because I have gone over the project multiple times with these people and these institutions' ethical codes of conduct and will continue to do so. My walk-away; that is, at what point my personal code of ethics will require me to abandon the project would be if the circumstances were such that continuing the project will cause serious harm to befall anyone or put any group or organization associated with the project into considerable financial detriment. It is not ethical to produce such destructive technology as guns, poisons, and bombs, but my invention does not fall into this category. I am sure of this because it is a purely digital development, and I could not possibly see it do any harm or perform any matter of deconstruction in any way.

The ethical duty I have to potential users of my product is to ensure they have a safe and accurate experience of Santa Clara University through the project. I will know I've gone too far in taking the user's welfare into consideration when personal influences start to affect the project in ways I had not envisioned for it to be. The point at which the ethical responsibility for using a product or service transferred primarily if not wholly to the user or customer is when they have been given the full warnings for the potential downsides of using the product but continue to do so anyway.

If the user is sick, then they should not use the application, or at least not the VR headset. If the user is a child, then they may not fully understand the technologies behind the project and may not handle the devices with proper care, but they shouldn't have too much trouble with motion sickness because it is more common with older generations. If the user is elderly, then they might be a little too overwhelmed by the virtual interface aspect of the project and should be advised to simply view the 3D campus on the screen of their computer instead of wearing an Oculus headset. The ethical ramifications of my project, and more particularly, the unforeseen consequences range from minor dizziness to serious head trauma caused by bumping one's head on a nearby object while wearing the headset and exploring the school digitally. My obligation to worry about these more

remote issues is just as paramount as the other ethical obligations I have in this project.

I ensure integrity in my research and design project by doing as much of the work myself as I can and citing any other source I use for the project. The processes I have in place to ensure the truthfulness and accuracy of my designs are geometric and photographic crosschecking precision. I know my data can be trusted because I am very thorough and precise with my work, and I would not fudge the results of certain tests and experiments just because it would be beneficial or convenient. I know someone else's data can be trusted if I personally know them to be of decent moral character and engineering skill, or if they have the seal of approval from a well-regarded engineering professor or advisor or from a prolific and trustworthy engineering organization. In disclosing how I conducted a scientific experiment or in sharing research results, I go as far as to share my whole test group without breaking any previously agreed confidentiality. I distinguish between errors of commission and omission by the definition that errors of commission are lies or falsehoods and errors of omission are simply instances where the whole truth is not told. My ethical obligation about the product does not extend to subsequent teams who may continue my work because the project is about the university. Other teams who might later try to continue working the project would only do so if they were working with or for SCU, and by that time I expect to have graduated from the university and moved on to other things, so it would thusly be out of my hands.

Chapter 4 – Development and Testing

4.1 Application Development

During the development of my project, I encountered a number of pitfalls that would set me back for months at a time while I tried to come up with a solution. As a consequence, the version I presented at the Senior Design Conference in May of 2015 was an unfinished model of the campus demoed in Blender. Even after I finished the rest of my courses at the university and no longer had to constantly juggle and choose between working on my project by myself and working to maintain reasonable grades in each of my classes, I still managed to have this problem. In December, I for the most part finished work on getting the approximate heights of all the school buildings all right in OpenStreetMap. In January of 2016, I figured out how to make the roofs of the buildings other shapes besides flat, which particularly helped in finalizing the Observatory behind McLaughlin-Walsh Hall. Then in late February, I followed a tutorial that enabled me to finally solve the problem of rendering heightmaps from non-plane-based meshes, and development progress noticeably hastened from there.

In March, I found a program to generate a high-resolution aerial map, and finalized the heightmap's render position accurately mapped to that map, also adding buildings in that area I previously did not include from OSM's current data on that area because they were now within the scope of this 9 by 11 aerial map. In early April, I determined the terrain import settings that would import the heightmap to scale with the user. Around that time. I also discovered that by making the terrain and aerial texture both square, I could apply the texture directly to the UE4 terrain of the school, instead of using the heightmap mesh as a collision map and importing a viewing model in tandem, as I originally believed I would need to do. This square heightmap also necessitated me to again incorporate buildings from the full OSM map which I previously did not include in my own, because they were previously too far North and South to show up in the rectangular map, but would now show up on the full square map.

4.2 Demonstration of Final Project

During the Senior Design Conference, I demonstrated a much earlier version of my project, which was an unfinished OSM map with some of the buildings' heights programmed in imported into Blender, with the buildings' names displayed in the corner when you click on them and a flyover view. The current version, or versions if you count both the final Unreal 4 project and the textureless OBJ version that can be imported to Blender and clicked on for building names, as the UE4 version cannot, is fully textured and traversable, with all the buildings the proper heights, and gravity and collisions working properly.

4.3 Testing and Bugfixes

During my tests of the map, I learned that texturing rectangular terrains with rectangular aerial maps in Unreal Engine 4 was near impossible with my current skillset, because of the way the program tiles and stretches textures. What simplified things was making both the terrain heightmap and the aerial map square images, and from there the texturization process really fell into place.

Figure 6. Bing Aerial Map of SCU Campus used to texture heightmap:

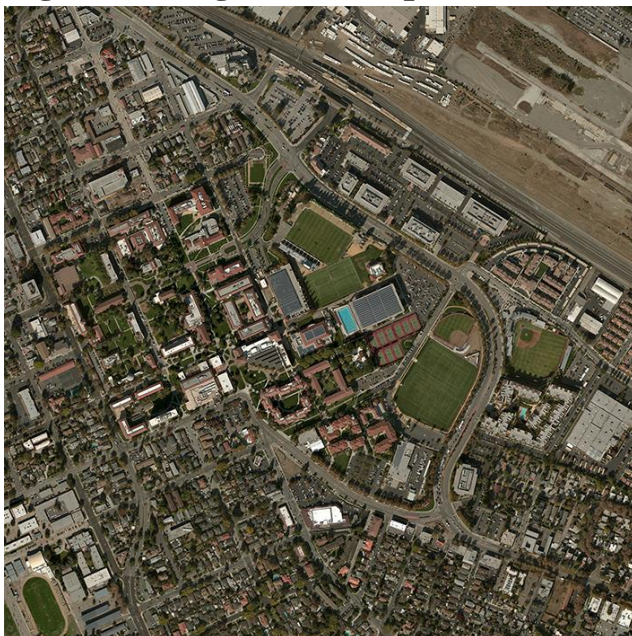


Figure 7. Heightmap of Campus generated from OSM2World OBJ:



4.4 Procedure

0. If you already have a heightmap and an aerial map of the campus, and you don't need to update either of them, skip steps 1-9.
1. Go to the SCU website's campus map and briefly review all the buildings they depict and describe as being part of the campus.

2. Go to the OpenStreetMap website and download a square slice of the campus using the approximate latitude and longitude coordinate boundaries of Left -121.94380, Top 37.343, Right -121.93017, and Bottom 37.35438 as an OSM file. (These may not be high or low enough, so just make sure the vertical coordinates line up with tile-utils' square Bing map.)

3. Download JOSM, and the latest version of Java if you don't have it already, and open the OSM file in it.

4. Download Google Earth Pro Free License Edition, set Santa Clara University as the destination, and using Tools → 3D Ruler, measure the approximate heights of all the buildings designated in the online map. Alternatively, you could retrieve the architectural specifications of the buildings and get their actual target heights from those archives/departments, or somehow measure all the buildings on site yourself, either of which would both be more accurate than the 3D Ruler.

5. Use the custom OSM file I made to do this. If updates are needed, export an OSM file of OpenStreetMap's latest version of the region using the method described above and merge them in JOSM by selecting and copying the needed structures in the first OSM file and pasting them into the other. There is a multi-OSM layer function in JOSM for doing just that, by dragging one OSM file onto a JOSM window with the other OSM file opened in it.

6. In JOSM, select Imagery → Bing Aerial Map to have Bing's satellite aerial map of the region overlaid under the OSM representations of the buildings. Warning: There may be some aerial offset from the real area, which it will warn you about, so other aerial maps might not line up in exactly the same way. Also, both the Soccer Training Center next to Locatelli and the McLaughlin-Walsh bridge are as of April 2016 not on Bing's map, as they were constructed relatively recently and so made before Bing's last map update of SCU. The only satellite map I know of that does currently contain those buildings is Google Maps, but those are also very proprietary.

7. Download OSM2World, and the associated textures on its website and also from <https://github.com/tordanik/isocore/tree/master/textures> and extract them to the same OSM2World folder. Then open the finished OSM file in it, using a batch command or bat file in the same folder with the command:

- `java -Xmx2G -jar osm2world.jar --config texture_config.properties --gui.`

8. In the top menus, select Export new OBJ, then open in in 3DS Max import it **as a single mesh**. Then execute the following sequence of menu commands:

- Click the MAX Button in top-left corner → Import → File to Import → SCU2.obj, and select Don't Import Materials. Or, alternatively, drag the OBJ file from Explorer/Finder onto the 3DS Max scene; import it as a single mesh.
- Open the Materials Editor, and make sure to use the compact materials editor.
- Materials Editor (Globe Icon on top menu) → Utilities → Reset Materials Editor slots → on scene
- Utilites → More → UVW Remove → Parameters → Remove all materials from scene
- New Material (Standard) → Diffuse → Gradient → Gradient Params → Gradient Type → Linear
- Gradient Parameters → Color 1: White, Color 2: Black
- Click Go back to parent material (the globe icon with an up arrow) → Blinn Basic → self-illumination 100 (Shadeless in Blender)

(Note: After subsequent executions of these steps, I found it faster to set the self-illumination before the Diffuse Gradient, because then you don't have to go back to the parent material.)

- Drag the material onto the mesh (specifically the buildings).
- Edit → Select All (to select all of the SCU Mesh)

- Right Menu → Modify (Icon of a Blue Arch in a Square) → Modifier List → Object-Space Modifiers → Unwrap UVW → Polygon
- Edit UVWs → Open UVW Editor → Freeform Mode [Free Transform], Select All (Again)
- Right Menu → Projection → Planar Map (Plane Icon) → Align Y (Y icon)
- UVW Editor → Top-Right Corner → Change from Checkerboard to Gradient (this is done to preview the texture mapping)
- Make sure the UV is aligned properly in the square (it should already be mostly prepositioned):
 - UVW Editor → Select All → Scale Selected Subobjects; slightly scale it down so the top line (that's Swig) is just within the grey top line of the grayscale square
- To make sure you can edit and scale the UVs in 3DS Max:
 - Select all in viewport → Unwrap UVW → Faces → Select none in viewport → UVW Editor → Select All → Scale
 (Note: If done wrong, the top of Swig Hall will look blocky or missing, or the buildings will look darker than the floor.)
- Select the top view from one of the four 3D preview boxes. Make sure it is set as an Orthographic top view.
- On your Orthographic Realistic Top View, vertically scale the window up so that it looks like a square and zoom so that the majority of the school is in the shot as is in the aerial map. To make the zoom more precise, make the following UI changes:
 - Customize → Customize User Interface → Mouse → Wheel Zoom increment 0.1, and then later 0.01
- Then narrow the render area so that the view shows exactly the same area of SCU campus that the square Bing aerial map does. Instructions on getting the Bing aerial map are further down.
- Of course, I already did all of this positioning, so if you need to import a new version of the OSM OBJ file without having to reposition the 3DS Max viewport, click/select the map's top-wise viewport and follow these steps:

- Go to Views → Save Active Orthographic View
- Delete and reimport the model
- Views → Restore Active Orthographic View
- Production Rendering → Render options → Output Size → Custom 8161x8161 → Render

(Note: To be clear, the settings should read as: Aperture Width 20.12 mm, Image Aspect 1.0, Pixel Aspect 1.0, and Render Resolution 8161x8161.)

- Save Image → Export render as a 16-bit greyscale PNG (No Alpha Channel)

(Note: Given more time, I could probably replicate this process in so as to avoid potential monetization issues in using 3DS Max.)

9. Obtain an Aerial Map of SCU

To generate the corresponding Bing aerial map, you will need tile-utils, a tile downloader that stitches large segments of Bing aerial maps into bigger ones. Please note that downloading and using Bing maps this way most likely violates Bing's Terms of Service, so I'd suggest as a replacement to use some of the aerial maps Santa Clara University is licensed to use, like the ones in the ArcGIS Lab on campus, though ArcGIS's aerial maps are even less up-to-date than Bing's. Make sure to cut a square slice of the new aerial map that exactly matches and corresponds to the area represented in the Bing map. There may be some offset with the buildings due to the zoom and angle of the two aerial map providers being slightly different.

- Download and install the following: tile-utils from vvoovv's GitHub project page, Python 2.7, and the Python Imaging Library for Python 2.7.
- Follow the online Bing tutorial called Getting a Bing Maps Key at <https://msdn.microsoft.com/en-us/library/ff428642.aspx>, then replace the text of the file in the tile-utils folder named bing_maps_key.txt with your key.
- Open the Windows Command line or the Mac/Linux Terminal, depending on the operating system being used, and execute the following two commands:
 - cd C:/path_to_tile-utils/tile-utils-master/

- C:/Python27/python.exe stitch.py -z 18 -o map.png -- -
121.94380,37.343,-121.93017,37.35438 bing

10. Import the campus heightmap and aerial map into Unreal Engine 4

- Get an Epic Games account → Download Epic Games Launcher → Download version 4.11.2 or newer of Unreal Engine 4
- New Unreal 4 Blueprint Project → First Person
- Once you've started the first-person project, delete all of the walls and cubes and blue text, most of which can be selected from inside the ArenaGeometry folder's subfolders in the top-right menu.
- Top-Right Corner → FirstPersonCharacter → Right Click → Edit → Select and delete the Spawn Projectile section in the Event Graph, then on the left side of the same window,
- FirstPersonCamera → Mesh2P → FP_Gun → Right Click → Delete, save blueprint,
- ConstructionScript → Delete AttachTo and its two connected nodes, select Mesh2P
- Right Panel → Rendering → Actor Hidden in Game, Save and compile, exit subwindow, save and build

(Note: You'll have to scroll a little bit down on that lower-right panel to get to the rendering section.)

- Left Menu → Select Terrain (3rd tab) → Import from file → Select the 16-bit heightmap → Input these settings: Quads 255x255, 1x1 sections, Scale X 17 Y 17 Z 6.3, Number of Components in X and Y direction 32x32, Location Z set to 1750 cm → Import

(Note: The exact measurements that allowed me to calculate those scale settings were aided by the Unreal Engine 4 project Unit One by Jeremy Baldwin. The Z Scale may or may not be a little off.)

(Note: The heightmap must be no more than 8161 on either side for UE4 terrain generator to display it properly.)

(Note: By importing the heightmap as terrain in UE4, the collisions and walkmeshes will autogenerate from them. The scale settings were chosen to linearly scale the mesh so that the buildings heights are to scale,

with the highest building, Swig Hall, being 33.2 meters high, and the lowest, the ground, being 0 meters high.)

- If needed, translate the terrain up so the player bounding box is just above the ground, then delete original floor.

(Note: Do not drag the player downwards; you should only move it on the X and Y axes. This is because if it goes down too low, i.e. below the editor's X-Y grid, it won't be able to traverse the map or move. And by "it", I mean a pair of robot arms. In earlier versions of UE4, the default player model was a blue mannequin.)

- In UE4 main editor, highlight FirstPersonCharacter on upper-right menu, on lower-right menu scroll down to CharacterMovement (Inherited)
 - Enable Crouch, Flying, Jump off ledge while crouched, and set character movement walking → Maximum walk speed to 2000 (So the user can traverse the large campus quickly)
- Top-Right Menu → Landscape → Bottom-right menu → Landscape → landscape material → new material in /Game/Landscape/Materials
- Right-click new material, edit (will take you to materials editor graph)
- Drag the map texture onto the NewMaterial graph. It will show up as a Texture Sample. Then, using the right Palette menu, find and drag LandscapeLayerCoords onto the graph.
- Select it and set LandscapeCoords' Mapping Type to TCMT XY and its Mapping Scale to 8161 (this is the UV scale).
- Connect the output of LandscapeCoords to the UVs of Texture Sample, and connect its white first output to the Base Color of NewMaterial.
- Save and check the landscape mesh.
- Rebuild the lighting on the map

11. Confirm the heights of the buildings are to scale (optional)

- Finally, use 3DS Max to export the OBJ map as an FBX, then import it into the Unreal Project and scale it up and translate it until the length and width of it perfectly match up with that of the map, then check to see if the heights are about the same. If the Swig Hall of one is a bit taller or shorter than the other, measure by how much, then use that to calculate

what the Terrain heightmap importers' new Z factor should be to make it the same height as the imported FBX's and reimport the heightmap.

12. Enable 3D Support

- Open your Oculus headset, if you have one, and follow the enclosed Oculus setup instructions to hook it up to your computer.
- Download OculusSetup.exe from Oculus's main website and install it.
- Create an account and register your VR headsets and any Oculus or Xbox controllers.
- Open the Oculus desktop app (if it crashes, run setup repair/reinstall)
 - Settings (Gear icon) → General → Allow unregistered apps
- Epic Games → Unreal Engine 4 → Project → Play Icon ► on the options dropdown → select VR Preview
- If VR Preview is greyed out, then either UE4 doesn't recognize the Oculus or the Oculus runtime is outdated and needs to be updated.
- Click Play to run the app. Use the mouse and arrow keys/analog sticks to move left and right, and press ~ to use console commands or exit out.

4.5 Conclusions

My project was successful in:

- Modeling whole campus to scale
- Displaying buildings with recognizable roofs and grounds
- Allowing one to walk around the whole campus

Weaknesses of my project due to time/personnel constraints were that:

- Models of the buildings are currently very flat and undetailed
- The textures of the campus are flawed because trees got in the way
- Rough edges due to processing of model into heightmap
- Stretched textures on buildings' sides due to use of aerial texture
- Did not label the buildings with names and descriptions in UE4

Future improvements others can make to improve this (by a team of people):

- Texturing the sides of the buildings
- Taking pictures of the campus to use as those textures
- Smooth the sides of the buildings (by using the OSM2World OBJ)
- Deapproximate the heights of all the buildings

- Add 3D labels and descriptions of the buildings in Unreal Engine 4
- Model the geometries of the buildings more precisely
- Model the interiors of the buildings

I represented the geometry every building on campus and provided the texture for the roofs on campus. I also completed the work on the underlying engine integration, including collision detection and first-person traversability. The project currently works with the Oculus Rift so that the project can either be viewed as a basic 3D walkthrough or as a virtual reality walkthrough. Work that I foresee others contributing to the project is texturing the sides of the buildings and smoothing the edges. However, from a programmatic and functional standpoint, all of the required mechanics for the walkthrough are in place.

I had hoped to write up a procedure on making heightmaps in Blender and believe I still could now, because it is both as well-equipped to make heightmaps as 3DS Max and, unlike 3DS Max, free-to-use commercially. Unfortunately, the video tutorials on doing so in Blender weren't quite as easy to follow as those in 3DS Max, and I've already perfected a process for completing the project in 3DS that would take days or even weeks to replicate in Blender. The project was already being submitted fairly late even without this addition, so, due to time constraints, I could not complete a procedure for generating the heightmap in Blender.

It was disheartening coming up against all those obstacles, but it was exhilarating making all those breakthroughs.

Chapter 5 – References

5.1 Literature Review

1. OpenStreetView

<http://www.openstreetmap.org/#map=17/37.34924/-121.94021>

2. Google Earth

<https://www.google.com/maps/place/Santa+Clara+University/@37.3499298,-121.9384107,439m/data=!3m1!1e3!4m2!3m1!1s0x808fcbaf36303ec3:0x561ca114f6d4e347>

3. Unreal Engine 4

<https://www.unrealengine.com/>

4. 3DS Max 2016 (Student License)

<http://www.autodesk.com/education/free-software/3ds-max>

5. tile-utils

<https://github.com/vvoovv/tile-utils>

6. JOSM

<https://josm.openstreetmap.de/>

7. Blender

<https://www.blender.org/>

8. Oculus Rift

<https://www.oculus.com/en-us/rift/>

Tutorials Used:

9. Unreal Engine 4 Landscaping Quick Start

<https://docs.unrealengine.com/latest/INT/Engine/Landscape/QuickStart/4/index.html>

10. Unreal Engine 4 Tutorial: Landscape Tool

<https://www.youtube.com/watch?v=FHVuVVHIUmM>

11. Unreal Engine 4 Oculus DK2 Quick Start

<https://docs.unrealengine.com/latest/INT/Platforms/Oculus/QuickStart/4/index.html>

12. Installing Oculus DK2 on a Mac

<http://www.uxconnections.com/installing-oculus-dk2-on-a-mac/>

13. UE4 Oculus Rift Template Introduction

<https://www.youtube.com/watch?v=oo2XKSvmSc0>

14. Unreal Engine 4 Oculus DK2 Best Practices

<https://docs.unrealengine.com/latest/INT/Platforms/Oculus/BestPractices/index.html>

15. Enabling Oculus Games and Apps From Unknown Sources [like Unreal Engine 4]

<https://support.oculus.com/878170922281071>

16. Oculus Software Setup

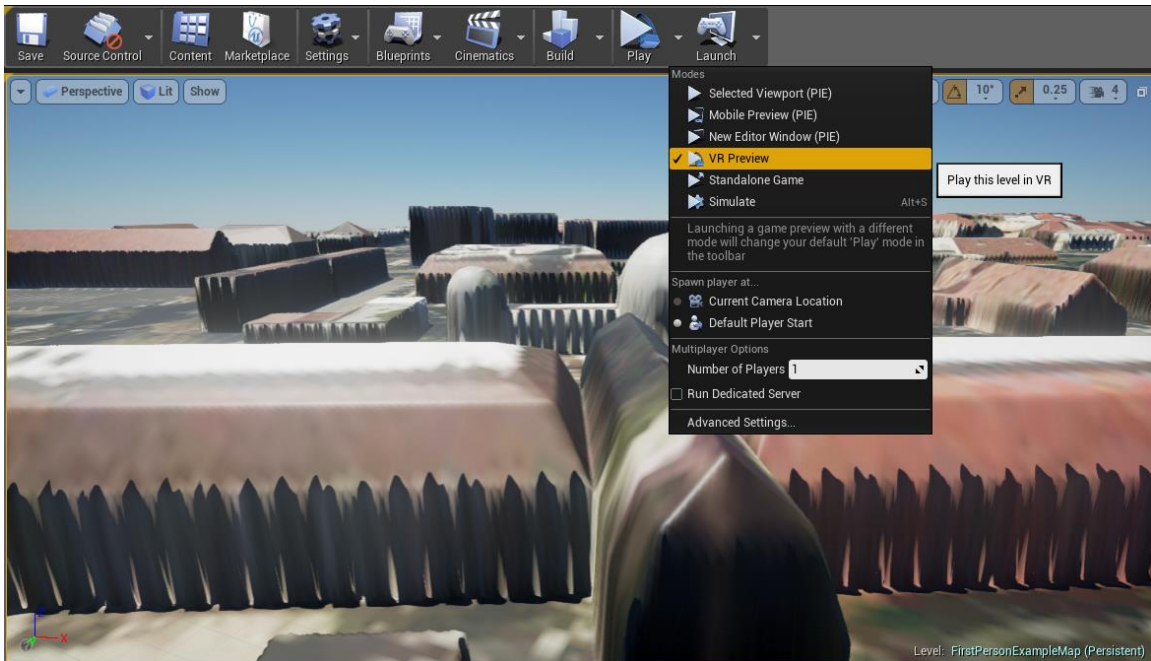
<https://www.oculus.com/en-us/setup/>

Chapter 6 – Appendix

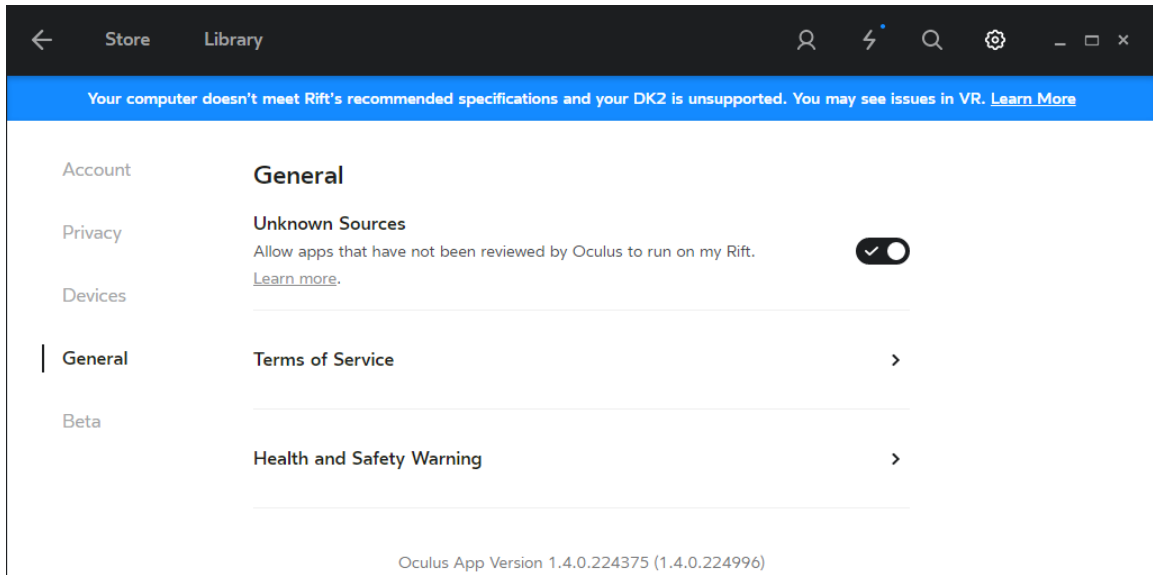
6.1 Screenshot of the campus model in JOSM, overlaying the Bing map



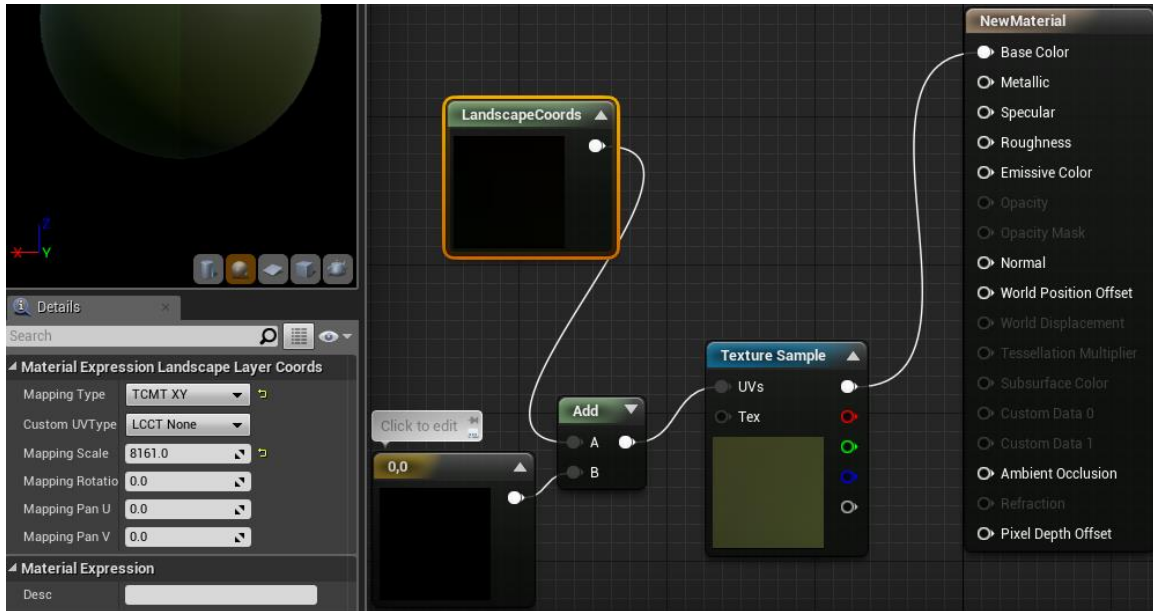
6.2 UE4 Project setting to run the SCU Level as a VR Preview App



6.3 Oculus Desktop App set to Enable VR in Unknown Sources like UE4



6.4 The SCU Landscape Texture's Nodes in the Material Graph Editor



6.5 Header file generated from the project's First Person Blueprint

FirstPersonCharacter__pf.h:

```
#pragma once
#include "Blueprint/BlueprintSupport.h"

#include "Runtime/Engine/Classes/GameFramework/Character.h"
#include "Runtime/InputCore/Classes/InputCoreTypes.h"
#include "Runtime/CoreUObject/Classes/Object.h"
class USphereComponent;
class USkeletalMeshComponent;
class UCameraComponent;
#include "FirstPersonCharacter__pf.generated.h"
UCLASS(Blueprintable, BlueprintType,
meta=(ReplaceConverted="/Game/FirstPersonBP/Blueprints/FirstPersonCharacter.FirstPersonCharacter_C",
OverrideNativeName="FirstPersonCharacter_C"))
class AFirstPersonCharacter_C__pf : public ACharacter
{
public:
    GENERATED_BODY()
    UPROPERTY(BlueprintReadWrite, NonTransactional,
meta=(Category="Default", OverrideNativeName="Sphere"))
    USphereComponent* bpv__Sphere__pf;
    UPROPERTY(BlueprintReadWrite, NonTransactional,
meta=(Category="MyCharacter", OverrideNativeName="Mesh2P"))
    USkeletalMeshComponent* bpv__Mesh2P__pf;
    UPROPERTY(BlueprintReadWrite, NonTransactional,
meta=(Category="MyCharacter", OverrideNativeName="FirstPersonCamera"))
    UCameraComponent* bpv__FirstPersonCamera__pf;
    UPROPERTY(EditDefaultsOnly, BlueprintReadWrite,
meta=(DisplayName="Gun Offset", Category="Default", Tooltip="Gun offset
from the camera location", OverrideNativeName="GunOffset"))
    FVector bpv__GunOffset__pf;
    UPROPERTY(EditDefaultsOnly, BlueprintReadWrite,
meta=(DisplayName="Base Turn Rate", Category="Default",
OverrideNativeName="BaseTurnRate"))
    float bpv__BaseTurnRate__pf;
    UPROPERTY(EditDefaultsOnly, BlueprintReadWrite,
meta=(DisplayName="Base Look Up Rate", Category="Default",
OverrideNativeName="BaseLookUpRate"))
    float bpv__BaseLookUpRate__pf;
    UPROPERTY(Transient, DuplicateTransient,
meta=(OverrideNativeName="K2Node_InputActionEvent_Key2"))
    FKey bpv__K2Node_InputActionEvent_Key2__pf;
    UPROPERTY(Transient, DuplicateTransient,
meta=(OverrideNativeName="K2Node_InputAxisEvent_AxisValue6"))
```



```

float bpv__K2Node_InputAxisEvent_AxisValue6__pf;
UPROPERTY(Transient, DuplicateTransient,
meta=(OverrideNativeName="K2Node_InputAxisEvent_AxisValue5"))
float bpv__K2Node_InputAxisEvent_AxisValue5__pf;
UPROPERTY(Transient, DuplicateTransient,
meta=(OverrideNativeName="K2Node_InputAxisEvent_AxisValue4"))
float bpv__K2Node_InputAxisEvent_AxisValue4__pf;
UPROPERTY(Transient, DuplicateTransient,
meta=(OverrideNativeName="K2Node_InputAxisEvent_AxisValue3"))
float bpv__K2Node_InputAxisEvent_AxisValue3__pf;
UPROPERTY(Transient, DuplicateTransient,
meta=(OverrideNativeName="K2Node_InputAxisEvent_AxisValue2"))
float bpv__K2Node_InputAxisEvent_AxisValue2__pf;
UPROPERTY(Transient, DuplicateTransient,
meta=(OverrideNativeName="K2Node_InputAxisEvent_AxisValue"))
float bpv__K2Node_InputAxisEvent_AxisValue__pf;
UPROPERTY(Transient, DuplicateTransient,
meta=(OverrideNativeName="Temp_struct_Variable"))
FKey bpv__Temp_struct_Variable__pf;
UPROPERTY(Transient, DuplicateTransient,
meta=(OverrideNativeName="K2Node_InputActionEvent_Key"))
FKey bpv__K2Node_InputActionEvent_Key__pf;
AFirstPersonCharacter_C__pf(const FObjectInitializer&
ObjectInitializer = FObjectInitializer::Get());
virtual void PostLoadSubobjects(FObjectInstancingGraph*
OuterInstanceGraph) override;
static void
__StaticDependenciesAssets(TArray<FBlueprintDependencyData>&
AssetsToLoad);
UFUNCTION(meta=(OverrideNativeName="ExecuteUbergraph_FirstPersonC
haracter"))
void bpf__ExecuteUbergraph_FirstPersonCharacter__pf(int32
bpp__EntryPoint__pf);
UFUNCTION(meta=(OverrideNativeName="InpAxisEvt_LookUpRate_K2Node_
InputAxisEvent_62"))
virtual void
bpf__InpAxisEvt_LookUpRate_K2Node_InputAxisEvent_62__pf(float
bpp__AxisValue__pf);
UFUNCTION(meta=(OverrideNativeName="InpAxisEvt_TurnRate_K2Node_In
putAxisEvent_34"))
virtual void
bpf__InpAxisEvt_TurnRate_K2Node_InputAxisEvent_34__pf(float
bpp__AxisValue__pf);
UFUNCTION(meta=(OverrideNativeName="InpAxisEvt_MoveRight_K2Node_I
nputAxisEvent_192"))
virtual void
bpf__InpAxisEvt_MoveRight_K2Node_InputAxisEvent_192__pf(float
bpp__AxisValue__pf);

```

```

        UFUNCTION(meta=(OverrideNativeName="InpAxisEvt_MoveForward_K2Node
_InputAxisEvent_181"))
        virtual void
bpf__InpAxisEvt_MoveForward_K2Node_InputAxisEvent_181__pf(float
bpp__AxisValue__pf);
        UFUNCTION(meta=(OverrideNativeName="InpAxisEvt_LookUp_K2Node_Inpu
tAxisEvent_172"))
        virtual void
bpf__InpAxisEvt_LookUp_K2Node_InputAxisEvent_172__pf(float
bpp__AxisValue__pf);
        UFUNCTION(meta=(OverrideNativeName="InpAxisEvt_Turn_K2Node_InputA
xisEvent_157"))
        virtual void
bpf__InpAxisEvt_Turn_K2Node_InputAxisEvent_157__pf(float
bpp__AxisValue__pf);
        UFUNCTION(meta=(OverrideNativeName="InpActEvt_Jump_K2Node_InputAc
tionEvent_6"))
        virtual void
bpf__InpActEvt_Jump_K2Node_InputActionEvent_6__pf(FKey bpp__Key__pf);
        UFUNCTION(meta=(OverrideNativeName="InpActEvt_Jump_K2Node_InputAc
tionEvent_7"))
        virtual void
bpf__InpActEvt_Jump_K2Node_InputActionEvent_7__pf(FKey bpp__Key__pf);
        UFUNCTION(BlueprintCallable,
meta=(BlueprintInternalUseOnly="true", DisplayName="Construction
Script", ToolTip="Construction script, the place to spawn components
and do other setup.@note Name used in CreateBlueprint function@param
Location          The location.@param          Rotation          The
rotation.", Category, CppFromBpEvent,
OverrideNativeName="UserConstructionScript"))
        void bpf__UserConstructionScript__pf();
public:
};

```