Sheffield Hallam University

College of Business, Technology and Engineering

Department of Computing

Project (Technical Computing)

[55-604708]

2020/21

Author:	Christopher John Noroozi
Student ID:	28014084
Year Submitted:	**2021**
Supervisor:	Mike Meredith
Second Marker:	Syafiq Zolkeply
Degree Course:	BSc Computer Science
Title of Project:	Realistic Cityscape Generator for Content Creators

Confidentiality Required?

NO

I give permission to make my project report, video and deliverable accessible to staff and students on the Project (Technical Computing) module at Sheffield Hallam University.

YES

Contents

1. Introduction 4
1.1 Project Aims 5
1.2 Report Structure 5
2. Investigation & Research 6
2.1 Potential User Persona 6
2.2 Existing Software 7
2.2.1 SimCity™ Game Franchise 7
2.2.2 Medieval Fantasy City Generator 8
2.2.3 Cities: Skylines 9
2.3 Land Zoning 10
2.3.1 City Planning & Land Zoning 10
2.3.2 Analysis of Land Zoning Applications: Tallahassee, Florida 10
2.3.3 Analysis of Land Zoning Applications: Manila, Philippines 12
2.3.4 Conclusions of Land Zoning Applications 13
2.4 Devices 14
2.5 3D Engines 15
2.5.1 Godot 15
2.5.2 Unity 15
2.5.3 Unreal Engine 4 15
3. Planning and Design 17
3.1 Data Structures 17
3.1.1 Graph 17
3.1.2 Two-Dimensional Array 17
3.2 Algorithm Outline 18
3.3 Development Tools 23
3.3.1 Trello 23
3.3.1 Github 23
3.4 UI Wireframe 24
3.5 Class Diagram 25
4. Development 26
4.1 Getting Started with Unity 26

4.1.1 Controlling the City Generation Algorithm 26
4.1.2 Framework for the Algorithm 26
4.2 City Generation 27
4.2.1 Industrial Zones 27
4.2.2 Green Spaces 28
4.2.3 Residential & High-Rise 29
4.3 Displaying the City 29
4.3.1 Models 29
4.3.2 Orientation 31
4.3.3 High-Rise Height 31
4.3.4 Industrial and Green Spaces 32
4.3.5 Roads 32
4.4 User Interface 33
4.4.1 Camera Control 33
4.4.2 On-Screen UI 34
4.4.3 Lighting Control 34
5. Testing 36
5.1 Manual Unit Testing 36
5.1.1 Unit Testing Plan 36
5.1.2 City Generation Tests 36
5.1.3 City Visual Formatting Tests 37
5.1.4 UI Tests 37
5.2 User Survey 42
5.3 Project Aims 43
5.3.1 Primary Aims 43
5.3.2 Unachieved Aims 44
6. Critical Evaluation 45
6.1 Deliverable Successes 45
6.2 Deliverable Limitations 46
6.3 Ethics 46
6.4 Future Aims 47
6.5.1 Important Improvements 47
6.5.2 Potential Added Functionality 47

7. Appendices 48
A. References 48
B. Project Specification 50
C. City Generation Testing Grid 52
D. City Visual Formatting Testing Grid 54
E. Survey Design 57
F. Survey Results 60
G. Survey Consent Form 63
H. Survey Participant Information Form 64

1. Introduction

The internet has enabled a multitude of new industries populated by more independent creators than was ever possible before - such as online video producers who upload their content to social media sites or independent game developers who create video games either alone or in a very small team. This independence allows such content creators a high level of creative freedom, however it comes with the drawback that every facet of work on the content they produce must be handles by themselves.

One such aspect that content creators must handle are background graphics to be used in:

- Intro and outro scenes in online videos, such as the point in a video where viewers are directed to other content by that creator.
- Skyboxes (the background graphic that users cannot interact with) for video game levels/menus.
- Images for social media layouts, for example the header image on Twitter or elements to be used in a story post on Instagram.

While these may be relatively minor tasks, they are important for creating professional looking content and a number of factors must be considered such as composition and colour to create a realistic & aesthetically pleasing result. Hence, I will be designing and prototyping a system that is able to create a realistic (regarding shape and structure) background with colours that are customizable by the user.

For the subject matter of the backgrounds my system will generate I have decided to procedurally generate a city layout that the user can maneuver a camera around to get the perfect angle for their content. The city generation algorithm will consider real world land zoning principles used in city planning in order to create a realistically distributed city; with a gradient of building heights moving from retail and industrial areas closer to the city center to smaller yet more numerous suburban areas closer to the green belt.

The system will generate a cityscape because it is a modern theme that can be applied to a wide variety of content, with the tone of the produced image able to be changed with a variety of colour options for different zones of the city and the lighting produced by the buildings. Additionally, vapourwave/retrowave styles are currently popular with content creators and cities feature heavily in such aesthetics. The output should not require a high-powered machine to run to ensure that all kinds of content creators can use the system.

Production of this project will require research into 3D graphics technologies to select an appropriate engine that has all the features needed to produce the system (such as luminosity and ability to output a randomly generated environment). Research into how land is zoned in actual cities will also be needed so the procedural generation algorithm can be developed with a basis in reality and thus can create a realistic cityscape.

The project will be referenced as the Cityscape Generator from this point onwards.

1.1 Project Aims

The objectives that I will aim to achieve through this project are as follows:

- To explore procedural generation algorithms and create an algorithm that can generate a city layout based on research done about real land zoning principles.
- To research different 3D graphics engines and their unique features to select an appropriate engine to develop the system in.
- To create a method to convert a generated city layout into a 3D output with various aesthetic additions in order to allow the output to be used as a graphic in content.
- To define and utilize professional-standard development methodologies/technologies to streamline the development of this project.
- To evaluate the success of the system in regard to creating graphics that can be used by content creators.
- To critically analyze the development/research process upon completion and find areas for improvement.

1.2 Report Structure

The second chapter of this report will cover research done into the tools I will use to complete this project (3D engine, programming languages and development tools/methodologies). It will also show the research done into how the project will meet the project aims; for example how land is zoned in cities, how this could be translated into a procedural generation algorithm and how the aesthetic features I require could be implemented.

The third chapter will cover the planning done before development begins such as UI wireframes, personas with scenarios and class diagrams. It will also cover the development tools that will be used such as version control. The fourth chapter will detail the development process and the fifth will cover testing done on the final prototype.

Finally, the sixth chapter will evaluate the success of the project throughout all its stages, including a reflection on the development process and indication of areas that could've been improved. Following chapters will feature referenced material from the rest of the report such as a bibliography and appendices.

2. Investigation & Research

2.1 Potential User Persona

As explained in section 1, there are a number of user groups that could have use for this system. Primarily, I feel that online content creators would have the most use of the system for creating graphics to display in content such as online videos or social media posts/layouts. Hence, I will define a user persona revolving around this use that will allow conclusions drawn from further research to be related back to this userbase:

Name	Edward Hall
Job Title	Online Video Producer
Demographics	• 26 years old. • Avid video game player and content creator. • Has a bachelor's degree in psychology. • Living alone in a flat.
Biography	Edward is a young post-graduate living in a studio flat in Manchester - and while he has only recently moved there, he has lived in Greater Manchester his whole life. He studied psychology at Manchester Metropolitan University, being motivated to do so out of personal interest. Edward, being introduced to video games at a young age by his older brother, has been a keen player of them ever since. Being well versed in grand strategy and simulation games he decided to create a YouTube channel 3 years ago to document strategies he has devised in such games. Steadily he has gained a small following on the platform.
Environment	Edward is quite proficient in using computers and video editing software to produce his online video content; however his knowledge is self-taught and does not extend past the necessities required to produce the content. His YouTube channel is not his primary source of income, however as the channel grows he is looking to transition towards making a full-time career.
Scenario	In an effort to increase the quality of his online video content (and thus increase traffic towards his channel), Edward is looking for graphics to use in the outro of his videos that will direct users to other popular uploads by him. Currently his outro scene is simply a flat image with annotations, and Edward would like the new outro to feature a cityscape that can change with every video, with background music added too. He feels this new dynamic background to the end of his videos would thematically fit his content and give his content a more professional appearance.

Figure 2.1: User persona of an online video content creator.

2.2 Existing Software

2.2.1 SimCity™ Game Franchise

The SimCity games are a series of videogames produced from 1989 to present that have the player managing a simulated city, maintaining and generating resources and infrastructure for the city; in order to improve and expand said city. While the Cityscape Generator is not a game, both it and the SimCity games display a 3D version of a city that can change with every instance of the system. Additionally, as identified by the persona in section 2.1: online content catering to such games has a thematic link to the Cityscape Generator and thus the output of the system being inspired by such products may help the system appeal to players of such games.

There are many games in the franchise but the one I have available to test is SimCity3000 (Maxis, 1999). The in-game city is displayed in an isometric view which is not analogous to the way the Cityscape Generator will display cities. However many similarities between the conclusions drawn from section 2.2 and the game exist such as: roads being organized into blocks or the player being encouraged to keep industrial areas away from housing and give suburban areas plenty of green spaces. Seeing these facts persisting between my research and actual published products confirms that they should be considered in the procedural generation algorithm.

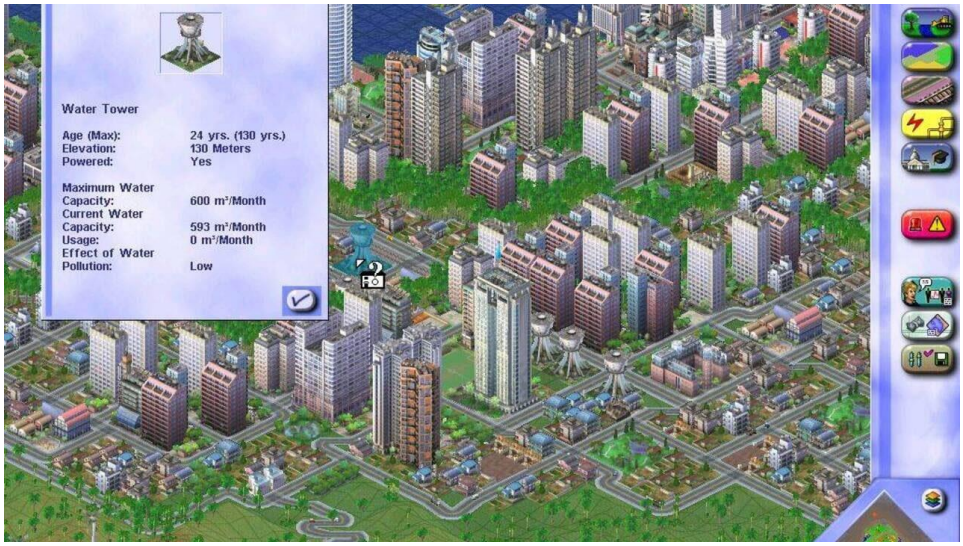


Figure 2.2: Screenshot of 3D view of city within SimCity 3000 Unlimited

Additionally SimCity3000 features some functionality that has not been identified for the Cityscape Generator yet such as the ability to save a city's layout and load it into the system at a later date. This may be useful for my system if the user wishes to work on the same city layout for multiple sessions. The game also allows the player to add features to their city whenever they want, and while this editing of the city layout may be useful in the Cityscape Generator, I believe it is outside the scope of the development timeframe.

2.2.2 Medieval Fantasy City Generator

The Medieval Fantasy City Generator is a web-based application that creates a layout of a medieval city and allows the user to view and edit an overhead view of the layout. As stated by the website itself: “The generation method is rather arbitrary, the goal is to produce a nice-looking map, not an accurate model of a city” (Watabou, 2020). Even though the procedural generation algorithm being used does not have the same goal as the one that will be developed for the Cityscape Generator; the non-uniformity of the edges of generated cities is a notable aesthetic feature that could be incorporated into my system by allowing the generation of “blank” land spaces close to the edges of the city layout.

Figure 2.3: Screenshot of the Medieval Fantasy City Generator

While the city generation algorithm in the Medieval Fantasy City Generator may not share the same goal as that of my system, the features provided by it could be worth including in the Cityscape Generator. The Medieval Fantasy City Generator provides a button to completely regenerate the city layout at any point. This is incredibly useful due to the random nature of the city generation algorithm meaning that any given layout may not be desirable, thus being able to regenerate it quickly is definitely a feature that the Cityscape Generator should include. Additionally, a selection of curated settings such as city sizes that a user can easily and quickly select from is helpful for new users to quickly get into the system without having to tweak a variety of settings. Although this feature may not be as useful for users such as the persona in section 2.1 who are already technically literate. Finally, the UI elements fade away after a period of inactivity, and including a similar way to hide UI elements to take screenshots of the generated city is paramount within the Cityscape Generator.

2.2.3 Cities: Skylines

Cities: Skylines is a 3D city building and management game much like the SimCity games. However Cities: Skylines differs from SimCity 3000 as that game displayed the city via a pre-rendered isometric view whereas Cities: Skylines renders its 3D city in real time and allows complete control of the player's perspective.

Figure 2.4: Screenshot of a custom player perspective in Cities: Skylines

While the game principles are very similar to the SimCity games and thus do not require further analysis, much can be learned from the presentation of City: Skylines. For example, the luminosity that comes from building windows greatly increase the aesthetic appeal of the city scape and the 3D engine I choose for developing my prototype will need to support such lighting effects. Additionally roads (which also generate in blocks much like SimCity) feature simulated traffic, giving the illusion of the city being populated and creating a much more dynamic view. Illuminated trails could be added to the roads of the Cityscape Generator to achieve a similar effect, however unique behavior would have to be developed for these trails to follow the roads and thus this would be stretch functionality.

Cities: Skylines also allows players to apply filters to their view of the city to highlight specific information about the health of various aspects of their city. Being able to filter specific information (such as which land zones have generated where) could be helpful not only for testing purposes but also for users who wish to see exactly how the cityscape has generated. This would require a texture swap of each model to flat colors representing the filter applied.

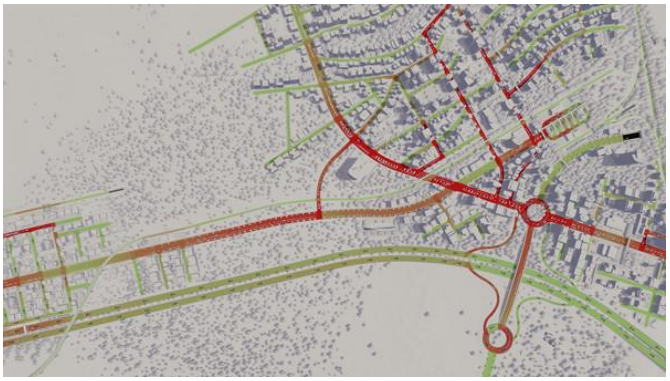


Figure 2.5: Screenshot of the roads highlighting filter in Cities: Skylines

2.3 Land Zoning

2.3.1 City Planning & Land Zoning

It is essential to decide what exactly the procedural generation algorithm used to create the city scape must do before further research can be done into the back-end coding language or the 3D graphics engine. The first step in developing the algorithm is defining the criteria the output of the algorithm must meet. Since the algorithm will generate a city with realistic land zoning, this the topic that must be researched first.

Land zoning is a form of city planning that designates different areas of land with different usages, within which buildings and operations have restrictions to adhere to. Not every country uses land zoning as a basis to plan the layout of a city, for example England judges how land should be used when planning permission is submitted for building something there (hence land is not zoned in advance, with the exception of the green belt). Often land zoning is done on a case-by-case basis, with different towns'/cities' communities requiring different things from their land, hence requiring different zoning laws (Planner's Web, 2001 & Michael Allan Wolf, 2008). As a result of the vast variance between different land zoning paradigms and the relative unimportance of specific laws (as the Cityscape Generator's output is for aesthetic purpose) - I will research this topic by looking at real-life applications of land zoning and finding the trends in how land zoning is applied, and the common types of land usage that exist (for example residential, industrial or green spaces). Laws that are often applied to land zoning regulations such as building height should be taken into account to decide upon the kind of buildings that will be displayed in these zones within the Cityscape Generator, but specifics on number of floors or types of businesses that work within can be abstracted.

2.3.2 Analysis of Land Zoning Applications: Tallahassee, Florida

The first example I will be using due to its well documented land zoning and the clear example of the differences between land usages it shows is Tallahassee, Florida.

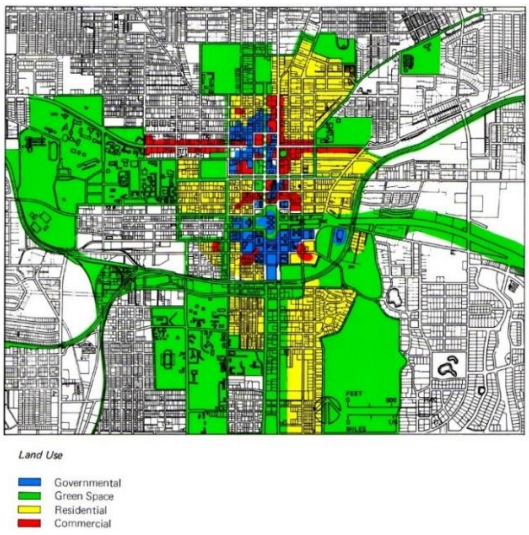


Figure 2.6: Zoning map of Tallahassee, Florida (Hisham N. Ashkouri, 2006), used under the Creative Commons Attribution-Share Alike 3.0 International license.

Figure 2.7: Ariel view of Tallahassee, Florida (Forbes, 2019)

As shown by figure 1.4 the city center is mostly comprised of commercial and governmental land usage, which the ariel view of figure 1.5 shows consists of large, high-rise buildings. As the city extends further away from this central business hub the land becomes increasingly used for suburban purposes such as relatively small residential housing or green spaces. In fact, green spaces are an especially important facet of land zoning in the proximity of residential areas. They have numerous benefits for physical and mental health (Nadja Kabisch, 2014) such as providing a space to encourage physical exercise or the proximity of nature breaking up the suburban sprawl (resulting in a positive effect on mental wellbeing). Trees and other greenery within these spaces also help to reduce pollution from city roads. It is estimated that an ideal city should have green spaces within 300m from any given suburban area, with the example of Ljubljana, Slovenia given by the research paper “Modern Compact Cities: How Much Greenery Do We Need?” (Alessio Russo & Giuseppe T. Cirella, 2018).

Another aspect of land zoning to notice from Tallahassee is its use of roads to separate plots of land. Roads are mainly straight and intersect at junctions to form blocks, within which each block is zoned for a purpose and built upon accordingly. However - as will be shown in the second example of land zoning - this is not the only way roads are incorperated into a city, and how roads should be implemented in the Cityscape Generator will be discussed at the end of this section.

Figure 2.6.1: Close up view of the road structure of figure 1.4

From looking at Tallahassee’s distribution of land usage we can conclude that high-rise, business-oriented zones should be focused near the city center whereas urban areas should be zoned closer to the outer

edge of the city boundaries, with green spaces equally and liberally distributed throughout. It is important to notice the lack of industrial land usage within this example.

2.3.3 Analysis of Land Zoning Applications: Manila, Philippines

The second example of land zoning I have chosen to analyze is Manila, the capital of the Philippines. I have chosen Manila as the second example as the land zoning that has been done there not only adheres to different regulations than Tallahassee, but also features a lot of cases that the fist example did not such as industrial zones, less uniform roads and a necessity to plan the city around the local geography.

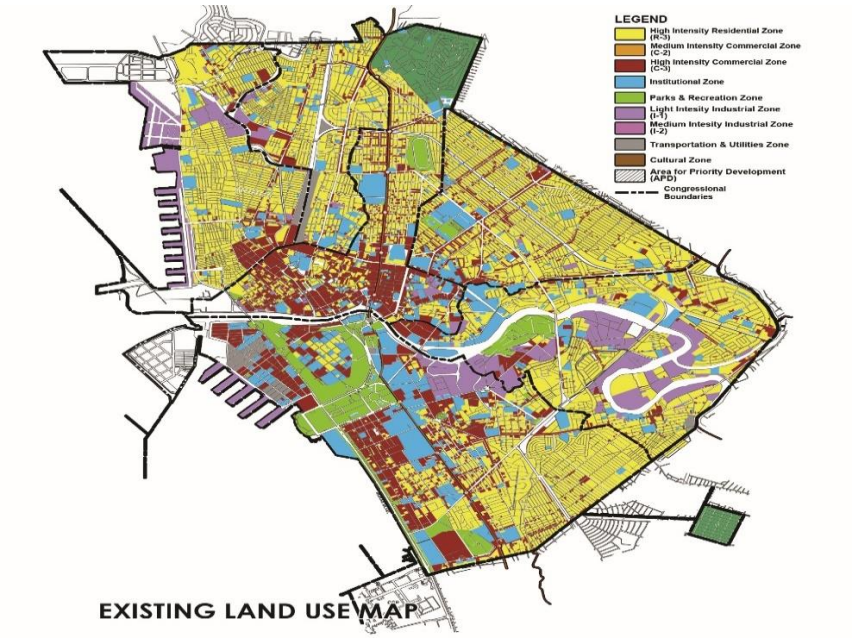


Figure 2.8: Land use map of Manila, Philippines (Manila City Planning and Development Office, 2017), used under the Creative Commons Attribution-Share Alike 3.0 International license.

Figure 2.9: Aerial view of Manila, Philippines (Patrick Roque, 2018), used under the Creative Commons Attribution-Share Alike 4.0 International license.

Manila is the antithesis of Tallahassee when it comes to land zoning. Firstly it has existed since at least the 15th century (Britannica, 2011) whereas Tallahassee was founded in the 1800s. As a result much less forethought was put into the layout of the city and thus land was zoned much more gradually, resulting in, for example, commercial zones being much more distributed throughout residential areas and green zones being much further away from most suburban areas. Additionally Manila was built along a river and on the coast; resulting in not only industrial zones being much more prevalent in areas with access to the ocean but also causing roads to be much less uniform as they must be oriented to the profile of the city along these bodies of water. In fact, cities are often built along the ocean due to the potential for industry such as transport or fishing (United Nations ESCAP, 2019).

The positioning of Manila's industrial zones is important to note. While they do primary exist along the cost/river side; the suburban sprawl has, in turn, begun to expand towards the north away from the industrial areas. It is better to keep industrial zones in one area and allow the rest of the city (especially residential areas) to grow away from them so the proximity of people's housing to potentially un-sightly (or even polluting) facilities can be kept to a minimum.

From this study of Manila we can conclude that: Industrial zones should be grouped together to keep exposure with suburban areas to a minimum. Additionally, while Manila shows that less uniform city layouts than Tallahassee are possible (roads not forming blocks, coastal areas and more distributed land zones); I believe it can be concluded that designing the algorithm to account for such city layouts is unwise as it would not only increase the complexity of the algorithm but also result in a less aesthetically pleasing cityscape.

2.3.4 Conclusions of Land Zoning Applications

From these two case studies, a couple of conclusions can be drawn about the kind of land zoning that the Cityscape Generator should adhere to:

- The roads generated should be straight and intersect at 3-way or 4-way junctions, forming blocks. This makes displaying the generated city easier as connections between roads would always be on the cardinal directions, and it would look more aesthetically pleasing as the blocks of buildings will be evenly spaced.
- Green spaces should be distributed across the generated city evenly, ensuring that all suburban areas are within range of at least one.
- Industrial areas should all be grouped together in order to keep exposure between residential areas and industrial areas to a minimum.
- The city center should comprise of high-rise and retail areas, whereas the rest of the city should be mostly residential land (excluding the aforementioned industrial areas and green spaces).
- Land usage should be abstracted into more general categories: high-rise, residential, industrial and green spaces. This is because the difference between, for example, different kinds of suburban housing is relatively irrelevant for displaying the city.

2.4 Devices

Before beginning to research ways the system can be developed it is first important to decide what platform it is being designed for. Ideally when selecting a platform it should be: widely used, accessible to develop for and able to offer all the functionality that will be foreseeably required for development. In the case of the Cityscape Generator this means being able to support 3D graphics libraries and programs developed in widely used 3D graphics engines. The content that the Cityscape Generator is intended to be used in tandem with is mostly produced on personal computers (such as game development or video editing). In that regard there are three major operating systems: Windows, macOS and Linux based operating systems (most commonly Ubuntu). These three operating systems make up over 80% of operating systems currently in use (W3Schools, 2020), hence I will analyze their suitability for the system. According to Net Marketshare, Windows machines make up the vast majority of Marketshare. (Net Marketshare, 2020):

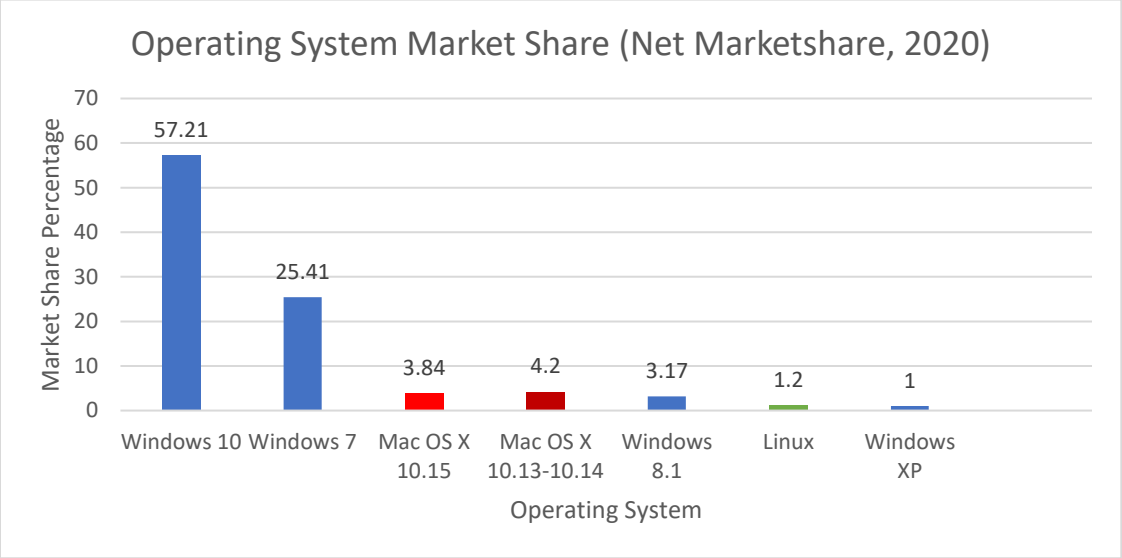


Figure 2.10: Market share of popular operating systems

Another facet of research on these operating systems that is pertinent to the development of the Cityscape Generator is what 3D engines they are compatible with. Based on research done in section 2.5 there are three major engines to consider, whose compatibility with various operating systems is shown by figure 2.10. Information on the compatibility was found from the system requirements documentation of Unity (Unity, 2020), Godot (PC Game Benchmark, 2020) and Unreal (Epic Games, 2020) respectively.

	Windows	**macOS**	**Linux Ubuntu**
Unity	Windows 7	Sierra 10.12	Ubuntu 16.04
Godot	Windows 7	“Unknown”	Any
Unreal Engine 4	Windows 7	✗	✗

Figure 2.11: Minimum operating system requirements for 3D engines

As Windows systems were dominant in both of these metrics, the system should function on Windows operating systems of at least the version Windows 7.

2.5 3D Engines

2.5.1 Godot

Godot is an open-source (Godot,2020) game development engine capable of creating both 2D and 3D environments. The engine being open source means there is no licensing fee for publishing products made with it, and while that is not useful during the prototyping stage it is an important factor to consider in potential later stages of the system. It supports a number of languages such as C++, C# and its own proprietary language - meaning there would be a lot of freedom when developing the city generation algorithm using pre-existing technologies (Godot, 2020).

A number of useful rendering features are also available within Godot such as real-time global illumination (allowing for dynamic lighting from either the sun or windows in buildings) and physically-based rendering which would help smooth out the edges of the very angular building shapes it will have to render within the city. However, the engine is relatively new and lacks a wide library of assets for prototyping and support for engine bug fixes compared to larger, more established engines. However since I am not intending to utilize pre-made assets for the prototype, Godot appears to be a competitive choice.

2.5.2 Unity

Unity is a 3D game development engine that is notable for its wide variety of easily accessible development features and large community supporting the engine (New Gen Apps, 2018). It is often cited as being very easy to develop with as a result of its component system that allows functionality to be placed onto objects within the 3D space in a modular fashion. This modularity along with the ability to create prefabricated objects that will be instantiated often (such as the buildings in the Cityscape Generator) and its wide variety of placeholder assets which are useful for prototyping, makes Unity seem like a good choice for developing the system.

Like other engines Unity features its own lighting engine. Unity's lighting options are not very user friendly but are very extensive and require tweaking to achieve an aesthetically pleasing result. Unity's renderer also features no built-in post-processing effects, meaning they have to be added in manually which increases development time. Unity appears to be a good choice for developing the system due to its ease of development and the level of control given over the lighting. However, the lack of post-processing, along with the licensing cost this engine requires for published products, means there are some pros and cons to consider.

2.5.3 Unreal Engine 4

Unreal Engine 4 is an industry standard 3D game development engine created by Epic Games, originally used to develop many AAA games. It features very powerful art pipelining features such as advanced shader settings and many lighting options (Unreal Engine, 2020) meaning it has the best aesthetic options out of all the engines I have evaluated in this section. Additionally, Unreal Engine 4 features a blueprint system for code development that allows for visual scripting of functionality (which is helpful for prototyping or quickly making changes to code). C++ can also be used to code within the engine.

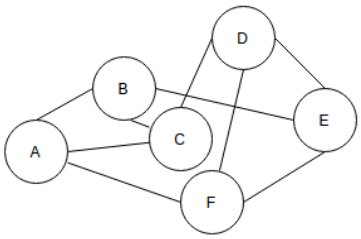
The engine is extensively documented and has built-in stock assets made by Epic, allowing for efficient testing of code before final assets are created, without the need to import placeholder assets from elsewhere. However, the engine is also the most advanced of the engines I have evaluated, and is the engine that I personally am least familiar with, so development time with this engine may be extended by the need to carry out further learning before prototyping can commence. Overall Unreal Engine 4 looks to be a powerful choice for creating an aesthetically pleasing output, though the added learning time must be considered.

3. Planning and Design

3.1 Data Structures

The first step in designing the procedural generation algorithm is deciding what data structure should be used to hold the generated city layout. The information that will be held about any given land that is generated is what kind of land it is and where its location is relative to the rest of the city. There are two data structures that immediately seem like they could be suited to hold this information.

3.1.1 Graph



A graph is capable of storing data relative to its connections with other data in the graph. Each node has a connection with other neighboring nodes. In the case of the city layout: each node would hold data about the piece of land such as its usage and information needed to display it, while the connections between it and other nodes would show the land that it is adjacent to.

Graphs have the benefit of being able to represent the relationships between each node within, which could be useful when deciding how a piece of land should be generated based on its neighbors. However the adjacency of each node is rather nebulous when considering a physical space, and additional information would need to be stored regarding what the relationship between each node actually is (such as what direction the land is in when compared to another neighboring land). Since the only way to do this would be using a weighted graph, which even then is only useful for representing distance than direction, I do not think a graph is suitable for holding the city layout data.

3.1.2 Two-Dimensional Array

A	B	C
D	E	F
G	H	I

A two-dimensional array is another possibility. An array can store a set quantity of data in an indexed order from which you can traverse it in order or get a result from a particular index. A two-dimensional array is an array that has two indices, meaning each position in the array holds another array. The result of this is a “grid” of data where a piece of land in the city can be accessed using the two indices. The location of the land within the two-dimensional array would represent the land’s relative location in the displayed city.

Two-dimensional arrays would be useful for deciding upon the exact location of a piece of land when placing it in a 3D space, as the two indices that indicate where data on that land is stored also represent the relative co-ordinates of that land. Neighboring land could be located simply by traversing one index back, forward, up or down within the array. However an index is only capable of holding one data type - so all land objects would have to inherit from one base class, or the array would have to be translated into different land objects after generation and upon display. Additionally, arrays have a set size, so the size of the array would have to be decided and validated before procedural generation could commence. These restraints are quite amenable, so I believe a two-dimensional array is the best choice.

3.2 Algorithm Outline

The two-dimensional array that the procedural generation algorithm populates will represent the grid of the city. Each space in the array is a single block of land. As mentioned in section 2.3.4, there will be four kinds of land usage: high-rise, residential, industrial and green spaces. A fifth that must be used to separate the different land usages are roads. The first constraint of two-dimensional arrays that must be considered is their fixed size, meaning the dimensions of the array must be decided upon creation. To decide how the size will be calculated we must consider the size of each generated land block. Since roads will be used to separate each block of land, and each block within the array is the same size, it would result in the vast majority of the generated city being roads which is not aesthetically pleasing. To rectify this, land zoned for non-road usage should always generate in groups, however such groups may be different depending on the land usage. To ensure that different shapes of zoned land can all fit together within the array, they must follow some rules. The first constraint I will impose upon different zones of land is that they must be comprised of 2x2 squares of land, such that they are noticeably larger than the single width roads that border them:

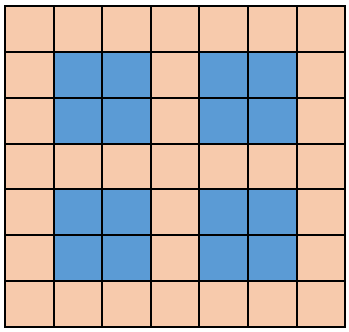


Figure 3.1.1: Example of 2x2 land squares in 2D array. Orange = roads, blue = non-road land usage.

However, for example, high-rise zones of land may only be 2x2 whereas residential zones may be longer than that to emulate long suburban lines of houses. In this case they may comprise of two 2x2 plots. This creates an issue however:

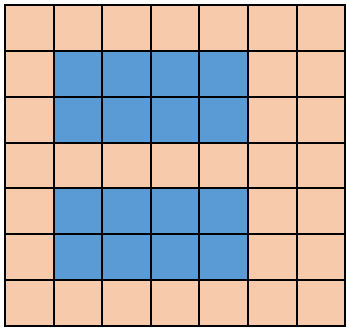


Figure 3.1.2: Example of 2x4 land squares in 2D array. Orange = roads, blue = non-road land usage.

As shown by figure 1.8.2, using multiple 2x2 land zones together results in a smaller amount of space used compared to two individual 2x2 zones – this causes issues when generating the city layout within the array as these two different land zones no longer tessellate. To resolve this, any land zone comprising of

multiple 2x2 squares of land must also incorporate the roads that would normally sit between the 2x2 zones. In the case of figure 1.8.2 the 2x4 zones would become 2x5, resulting in this formation:

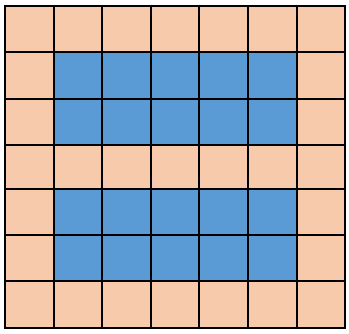
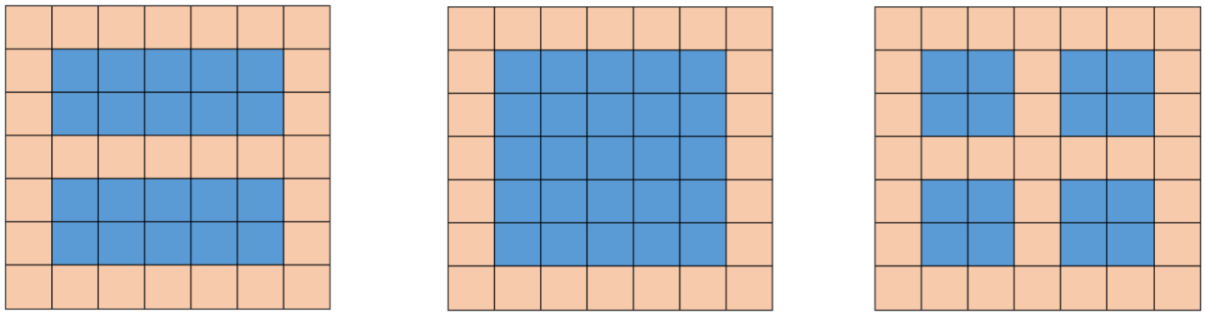


Figure 3.1.3: Example of 2x5 land squares in 2D array. Orange = roads, blue = non-road land usage.

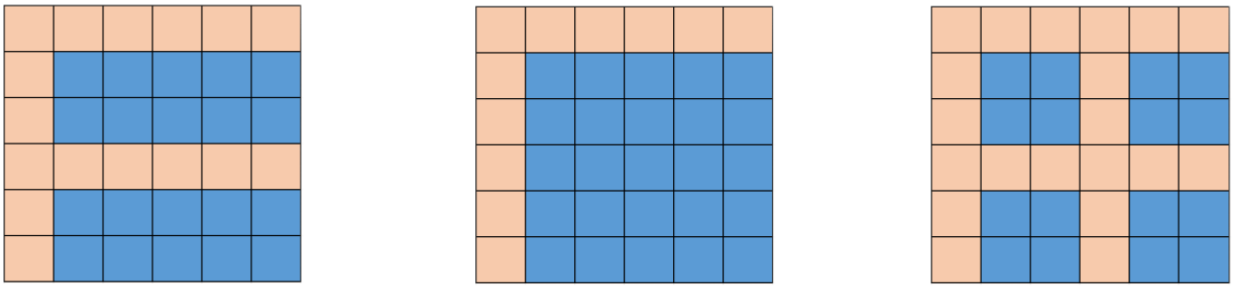
With this in mind, the land sizes I will use are as follows:



2x5 residential | 5x5 green space/industrial | 2x2 high-rise

Figure 3.1.4: Example of land templates for different land usages. Orange = roads, blue = non-road land usage.

Using tiles such as these that are always the dimensions of 7 x 7, the dimensions of the array will always be a multiple of 7 and thus can be generated using a random function, then rounding down to the nearest multiple of 7. However, in practice the function should round down to a multiple of 6. This is because a road of 1 width should be placed between each zoned land, meaning the tiles generated would actually look as follows, with a road added at the end of each row and column afterwards:



2x5 residential | 5x5 green space/industrial | 2x2 high-rise

Figure 3.1.5: Example of land templates with edge roads removed. Orange = roads, blue = non-road land usage.

As concluded in section 2.3.4, each land zone also must follow a rule as to how they should be placed within the city layout:

- Green spaces should be distributed across the city evenly, ensuring all land is within range of one.
- Industrial land should all be grouped in one area of the city.
- High-rise areas should be grouped around the center of the city.
- Residential areas should be towards the outside of the city.
- Roads should form blocks and intersect at junctions.

Since every space must be within range of a green space, the array should be populated with these first. The city layout can be iterated through, placing green spaces at slightly irregular intervals so the city does not look too artificial. These irregular intervals can be found by traversing the city at regular intervals, then adjusting the position of traversal across the axis by multiples of 6 (the size of the 6x6 templates) as shown in figure 3.2:

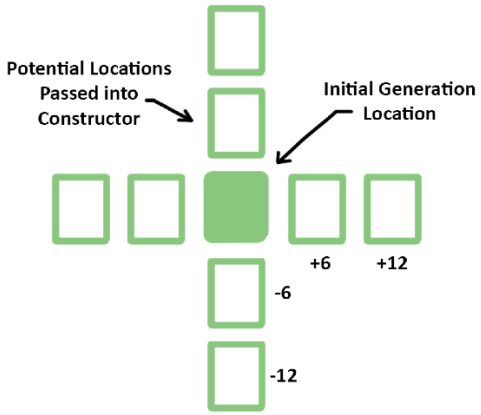


Figure 3.2: Diagram showing how the placement of green spaces is randomized.

To ensure that industrial spaces are all grouped together they will be placed in one corner of the city, randomly selected using the size of the city layout as the coordinates of the corner:

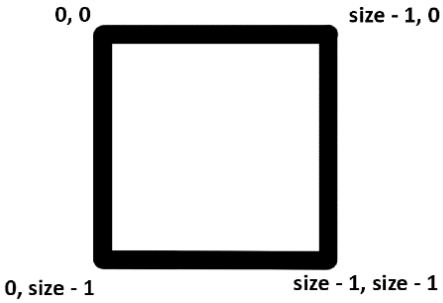


Figure 3.3: Diagram showing how the index of a corner in the city layout is found.

To populate this corner with 6x6 industrial land templates, the direction that the corner should be filled from is found by checking which cardinal directions from the corner are within the bounds of the city layout 2D array:

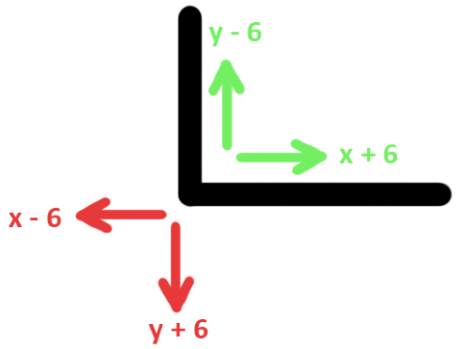


Figure 3.4: Diagram showing how the direction to fill the corner from is found.

The number of industrial spaces will scale based on the size of the city layout, so to ensure that the corner is filled evenly: industrial spaces will be placed in one direction until a limit is reached, at which point a new row of industrial spaces is started. This limit can be found using the square root of the total number of industrial spaces to place (therefor creating a square of industrial zone 6x6 layouts):

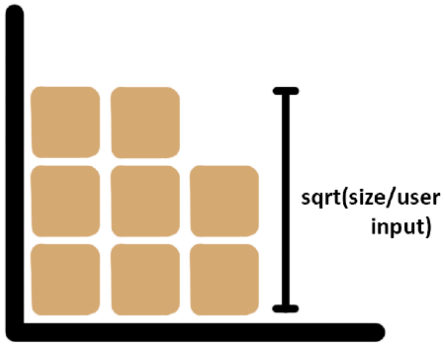


Figure 3.5: Diagram showing how industrial zones are evenly populated within a corner of the city.

Finally the high-rise and suburban areas – they are placed the same way, filling all the other spaces left in the array. However closer to the center of the array (halfway to the max value of each index) there should be a far higher chance of generating high-rise land, whereas closer to the extreme values of the indices there should be a much higher chance of generating residential land:

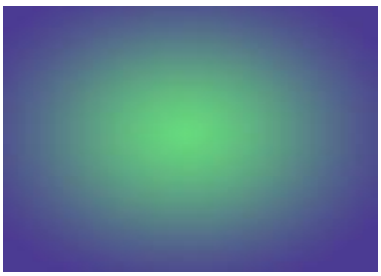


Figure 3.6: A gradient map showing the distribution of high-rise areas vs. residential areas. Green = high-rise, Purple = Residential.

The distance a particular index within the city layout is from the center of the 2D array can be found using Pythagoras' theorem:

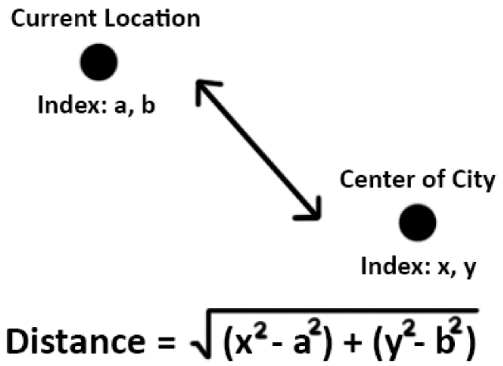


Figure 3.7: Diagram showing how the distance between an index and the center of the city is calculated.

There would need to be some visual formatting of the land generated before it could be output via iterating through the array using a nested loop and creating 3D objects as it goes. For example:

- Roads need to know if they are a 3-way or 4-way junction based on their cardinal neighbors (if they are surrounded on 3 or 4 cardinal directions by more roads, they must be a junction).
- High-rise buildings should become higher the closer to the center of the array they are, requiring a comparison to the center of the array and value of height being assigned.
- All residential buildings should be facing the same direction (as houses often all face towards the sun).
- Green spaces may have a different appearance depending on their proximity to the built-up city center.

This visual metadata can be generated either upon displaying the city or by instantiating all the buildings in the array as objects; containing functions to calculate and store this kind of information.

3.3 Development Tools

3.3.1 Trello

Trello is an online tool that allows for the organization of various tasks within a project using a Kanban framework. A project is broken down into self-contained tasks which can be moved onto different stages of completion and categorized with other similar tasks. The tool is useful for not only keeping track of every facet of the project development cycle but also for gauging the progress made on the system to keep development on-time.

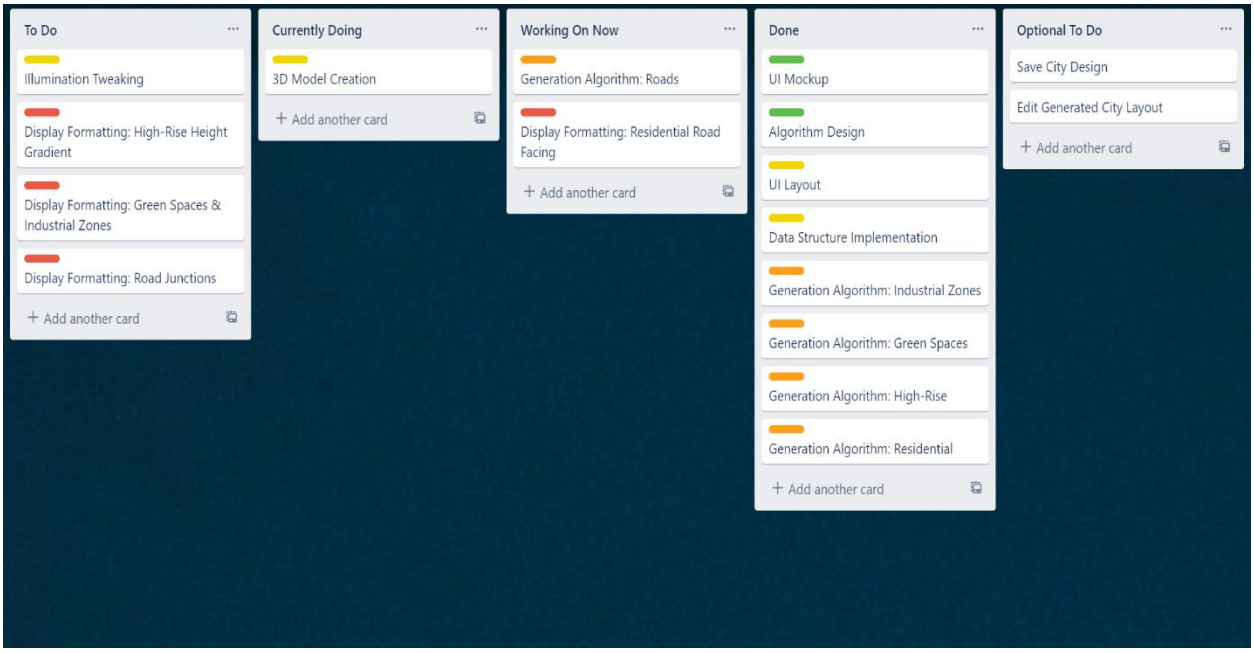


Figure 3.3: The Trello board used to organize the project at a mid-point during development.

3.3.2 Github

Github is an online version control tool that allows users to push codebases to a cloud-stored repository. This code can be branched so that changes can be made while still keeping the original code intact. I will be using Visual Studio's Github extension to push changes I make to the various scripts of the system to a repository. This will allow efficient backing-up of changes between coding sessions and allows the reversal of erroneous changes via git pull functionality.

3.4 UI Wireframe

To design a user-friendly UI with all required functionalities present: a wireframe of the final UI layout is created. All required inputs and controls are presented by it with usability functions such as descriptions of controls or conversions from numeric inputs to sliders for ease of input. The wireframe may need to be iterated upon during the development phase for non-core functions that did not make it into the final system or for additional options that were not identified during the design phase.

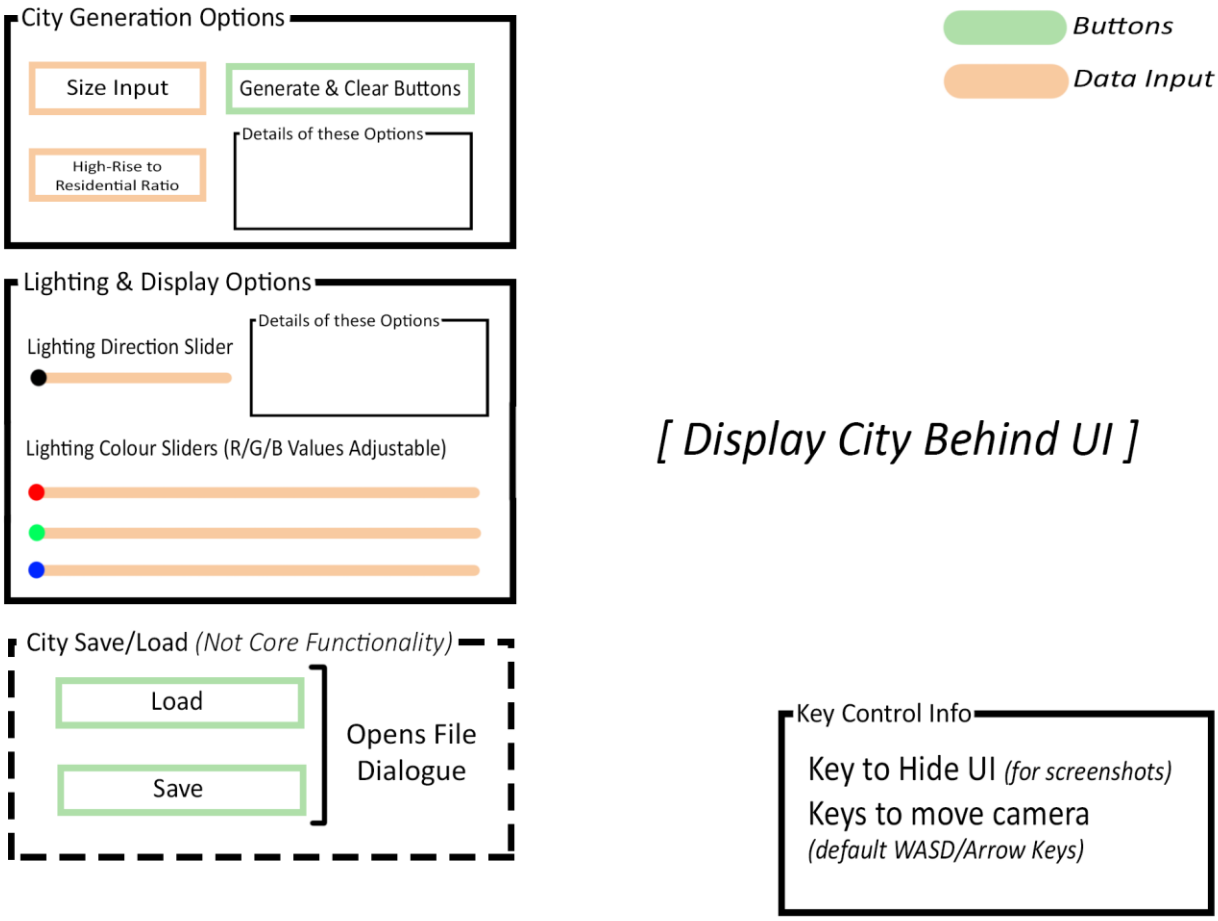


Figure 3.4: The UI wireframe of the Cityscape Generators controls. Colour key in top right of figure.

3.5 Class Diagram

Designing a class diagram before commencing development allows for early development to be done with consideration for integrating it with functionality due to be developed later on. Interactions between objects in the system are key to the modular design of each city zone (such as being able to check its neighbors or inherit from a generic land zone base class), thus a UML diagram is well suited to designing the system's back-end structure.

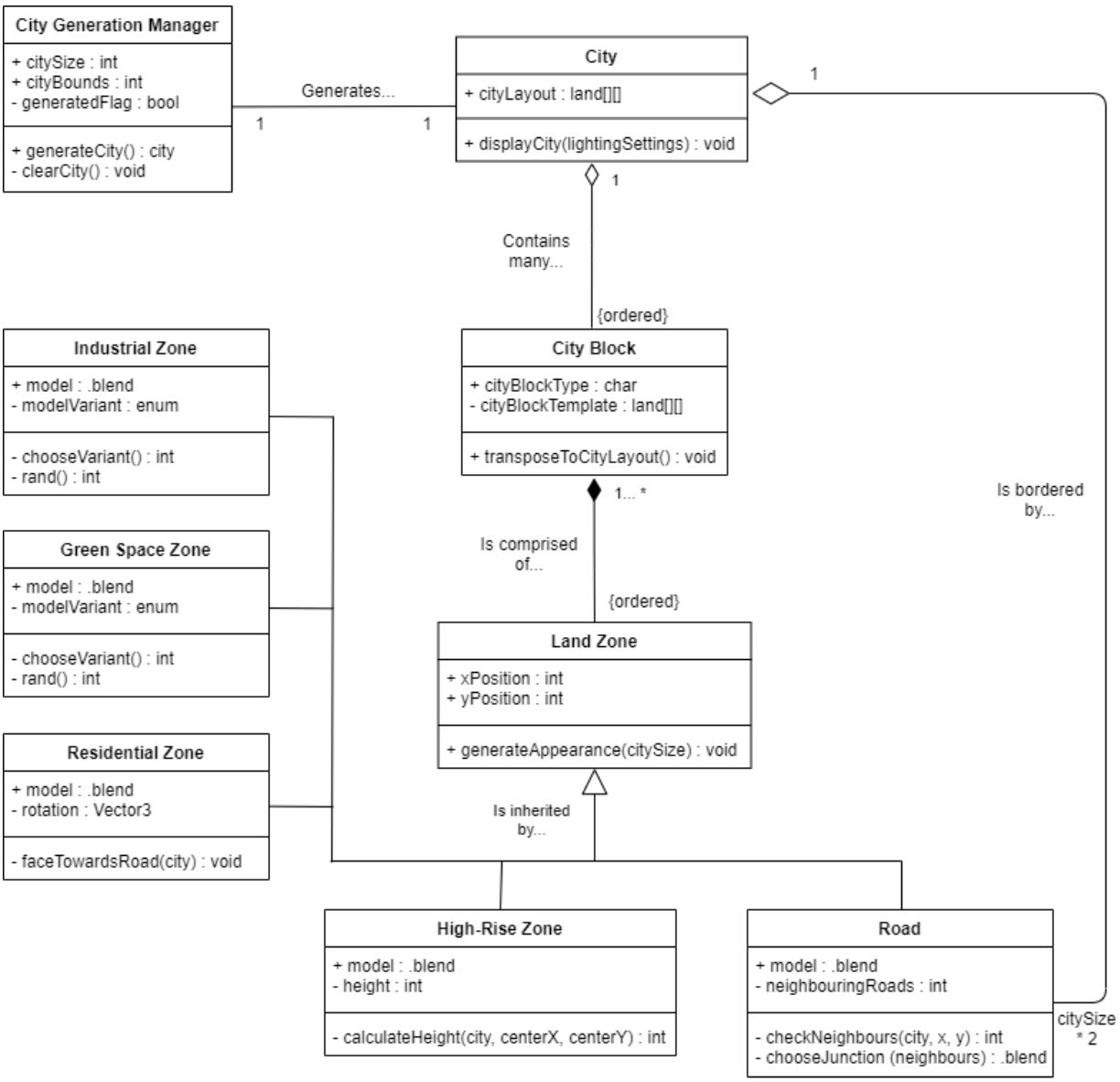


Figure 3.5: UML class diagram which describes the inheritance of different land zones, the aggregation of these land zones into the city layout and the object controlling city generation.

4. Development

4.1 Getting Started with Unity

4.1.1 Controlling the City Generation Algorithm

The system was developed using Unity due to my familiarity with the engine combined with the component system's synergy with the object-oriented approach this prototype will implement. For example (as shown in figure 4.1), the city generation component of the system can interact with the 3D space by adding a C# script component to a cube with its renderer component removed. This script can create prefab objects of each land zone with their own components for their aesthetic display:

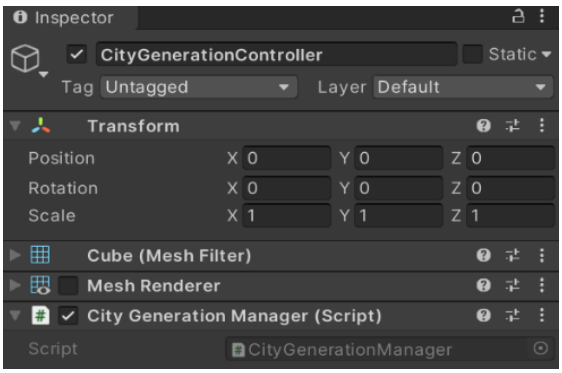


Figure 4.1: The attributes of the controller object, note the disabled mesh renderer.

4.1.2 Framework for the Algorithm

Land zones are represented using an inheritance-based structure within which the base abstract class contains data and methods for positioning a land zone within the city. Inherited classes will contain specialized functionality for generating the visual component of each different kind of land zone:

```
abstract class LandZone
{
    private int citySize;
    public int xLocation, yLocation;

    public LandZone(int citySizeInput, int xLocationInput, int yLocationInput)
    {
        this.citySize = citySizeInput;
        this.xLocation = xLocationInput;
        this.yLocation = yLocationInput;

        this.GenerateAppearance();
    }

    public virtual void GenerateAppearance()
    {
        // To be polymorphed by inherited classes
    }
}
```

Figure 4.2: Base class definition for the LandZone object that will specialize into different zones.

As shown in section 3.1, there is a 6x6 template for each land usage generated by the algorithm. Hence, templates are created for each - stored in 2D char arrays as shown in figure 3.3. A function to transpose this template into the city layout will replace each char with the land zone object it represents.

```
templateResidential = new char[6, 6] { { 'R', 'R', 'R', 'R', 'R', 'R' },
                                       { 'R', 'r', 'r', 'r', 'r', 'r' },
                                       { 'R', 'r', 'r', 'r', 'r', 'r' },
                                       { 'R', 'R', 'R', 'R', 'R', 'R' },
                                       { 'R', 'r', 'r', 'r', 'r', 'r' },
                                       { 'R', 'r', 'r', 'r', 'r', 'r' } };
```

Figure 4.3: A template holding the layout of a residential area. R = road. r = residential tile.

Finally a 2D array is created based on a user input size (rounded to a multiple of 6 to adhere to the 6x6 template format) to hold the city layout.

4.2 City Generation

4.2.1 Industrial Zones

To create the industrial zone generation behavior described in section 3.2, a corner is randomly selected first using rand() to generate a number between 1 and 4. This number is an enumeration of a corner and a switch statement takes this enum and returns the coordinates of the corresponding corner.

Using the method shown by figure 3.4, variables are set to the increment/decrement of the X and Y axis that will be used to place industrial spaces. The number of industrial zones to generate scales based on the size of the city using the following method:

- Calculate the number of 6x6 land templates within the city by dividing the size of the 2D layout array by 36 (the number of individual land zones within a 6x6 template).
- Divide the total number of templates to generate by a scaling value (default 5).
- Cast this value as a double as it may not be a whole number and use Math.Round() to get an integer, shown in figure 4.4:

```
int industryZonesGen = (int)Math.Round((double)(landToGenerate / 5));
```

Figure 4.4: The calculation of the variable that contains the number of industrial zones to generate.

Finally a nested for loop traverses through the axis placing industrial templates from the starting corner until the current number have been placed. Once the rounded square root of the number of industrial zones to generate is reached across one axis, the next axis across is populated to create a neat formation as shown in figure 4.5; in which 3 zones had to be generated and two were placed on one axis before placing the final one in the adjacent column.

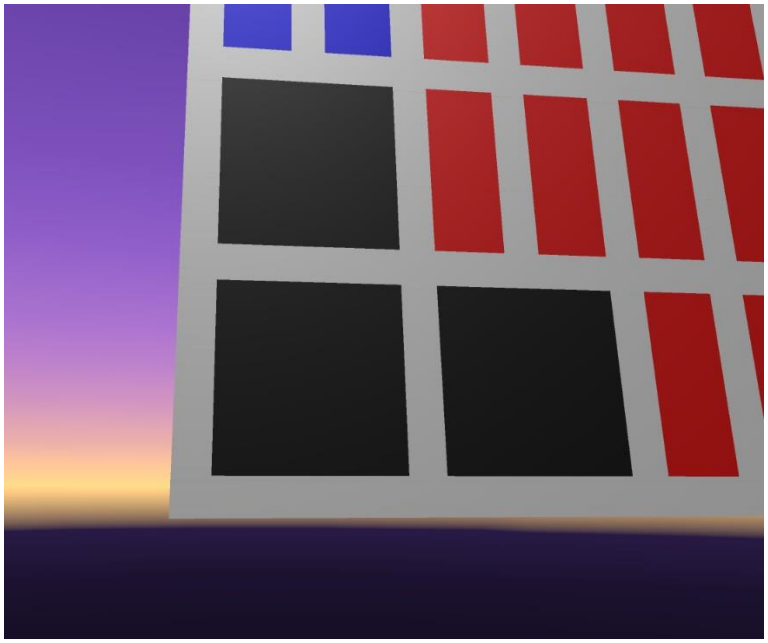


Figure 4.5: Example of industrial zone generation. Black = industrial zones, Red = surrounding residential zones.

4.2.2 Green Spaces

Distribution of green spaces across the city layout is done via a nested for loop that traverses the X and Y axes of the 2D array in steps of 6x6 land templates. At regular intervals along the traversal a green space template is transposed into the 2D array, distributing the green spaces across the city. The X and Y position arguments passed into the transpose template helper function have a random multiple of 6 (that can be negative) added to them to ensure that the position of green spaces are randomized within a small boundary (as shown in figure 4.6):

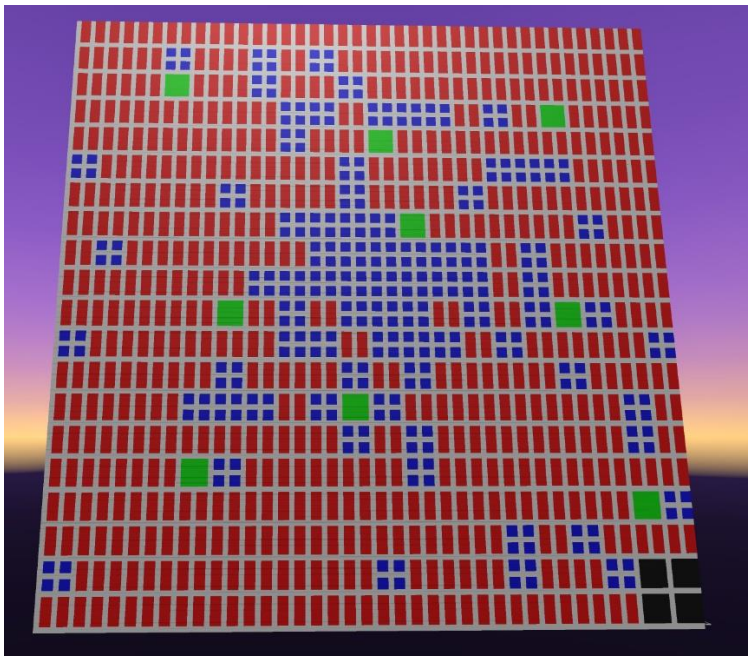


Figure 4.6: Example of green space distribution within the city layout array. Green = green space land zones.

4.2.3 Residential & High-Rise

Residential and high-rise templates fill the rest of the city layout, with a gradient of mainly High-Rise templates occurring towards the center of the city and mainly residential templates occurring towards the edges. Once other land zones have populated the city layout, the 2D array is once again traversed using a nested for loop to travel across the X and Y axis.

With each traversal step of 6x6, the current space is checked to see if it is populated. If it is, a green space or industrial space already exists at this 6x6 template space and so traversal continues. If the value at the current space is null, it must be filled with either a high-rise or residential template. The distance from the current space to the center of the city is found using Pythagoras' theorem (as explained in section 3.2).

A user input boundary (representing the ratio of high-rise to residential land) is divided by this distance, and a random value is generated using rand() between the edge of the city and the center. If this random value is less than the result of the division, a high-rise template is transposed, else a residential template is transposed. This has the effect of an increasingly higher chance to generate high-rise spaces towards the center of the city, while also having a significantly lower change to generate high-rise spaces outside the user input ratio (as can be seen in figure 4.6). The random value is used to ensure that it is not a perfect gradient of high-rise to residential, thus creating a more realistically uneven cityscape.

4.3 Displaying the City

4.3.1 Models

Upon generating a space within the city layout, the initialize function of the space is called. A different initialize function is called depending on the type of object being created via polymorphism (as shown in section 4.1.2). However, all zones require a 3D model to represent them. Since the city layout is comprised of equally sized zones, each zone's model will be modelled upon the same square space using Blender:

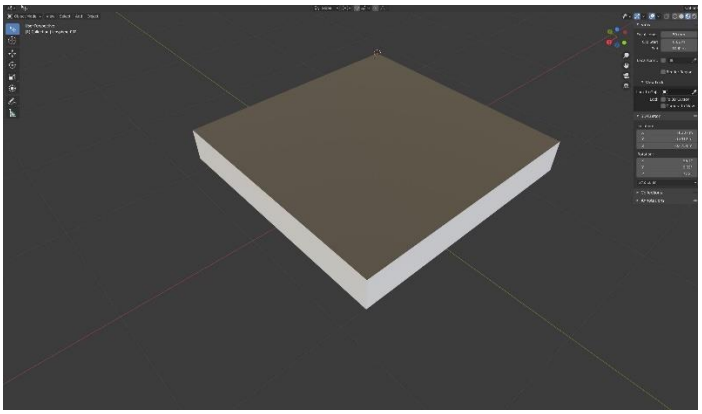


Figure 4.7: The base model in Blender upon which all tiles will be modeled.

From this base some basic models are created representing various land zones:

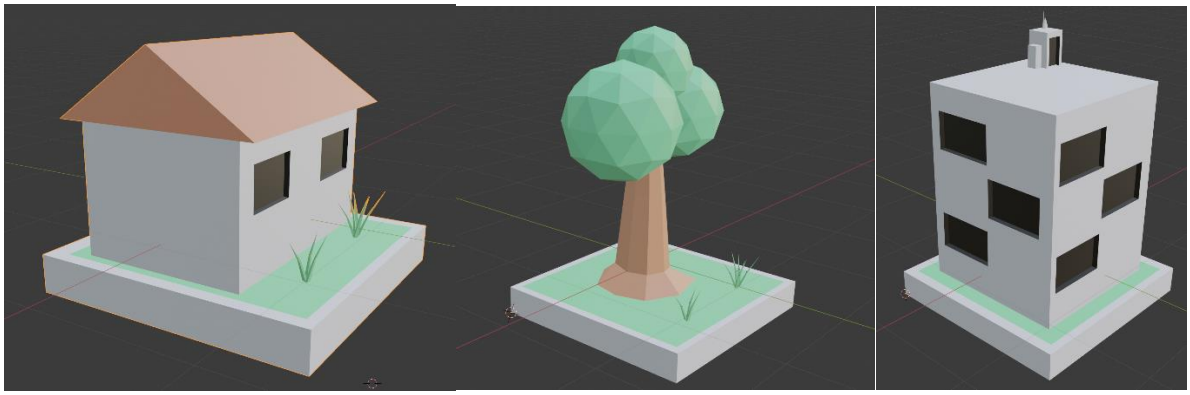


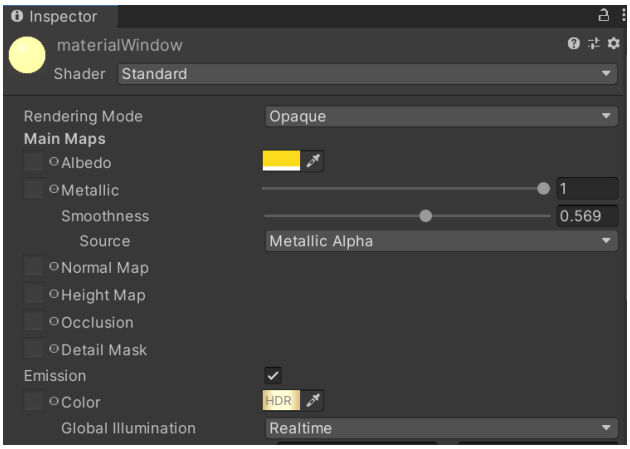
Figure 4.8: Examples of some models used to represent different land zones. From left to right: residential tile, a variant of a residential tile for flavor, smallest high-rise tile.

These models have their origin set to their north-western corner and have their scale and position reset so that Unity can handle their size relative to each other. Once imported into Unity, to ensure that they can be instantiated as GameObjects each is stored in a prefab. Prefabs can then be created as GameObjects script-wise via the following call:

```
GameObject newLand = GameObject.Instantiate(prefabLand, new Vector3(xLocation, 0, yLocation), Quaternion.identity) as GameObject;
```

Figure 4.9: Line of code used to instantiate a prefab.

Within the prefabs of these models, where appropriate, planes are created with a luminous material to emulate light coming out of windows:



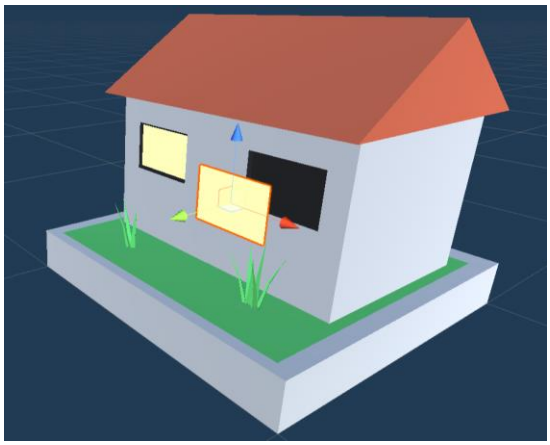


Figure 4.10.1 (Left): Material attributed to add luminosity. Figure 4.10.2 (Right): Placing windows into residential model.

4.3.2 Orientation

To ensure that each land zone placed faces the correct way (towards the road that bounds it), the land zone adjacent to the zone being initialized is checked. Since each zone is generated from the bottom left of the template to the top right; the zone south of it will already be initialized, so it is polled to check its type. If its type is a road, the model of the zone being initialized is rotated 180 degrees and the x & y positions are adjusted by one to ensure it stays in line with adjacent zones.

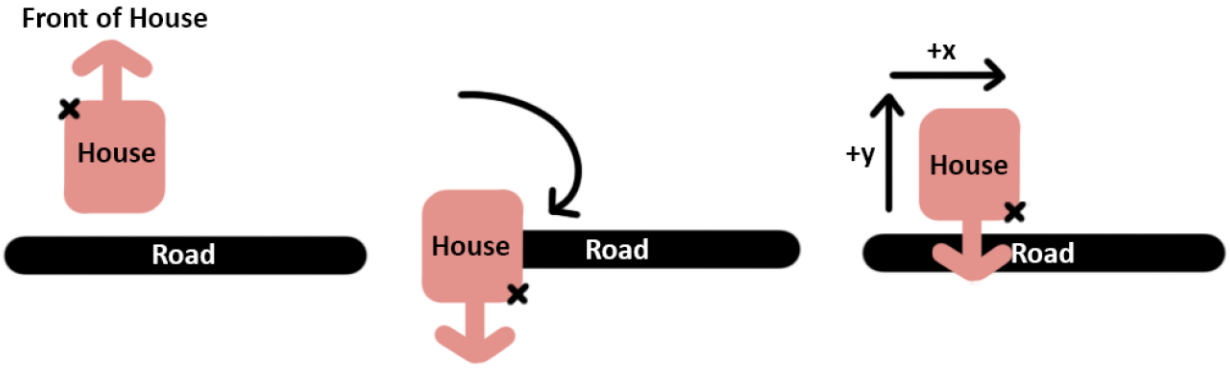


Figure 4.11: Diagram showing how residential tiles are rotated to maintain their direction facing the road.

4.3.3 High-Rise Height

To create the appearance of a varied skyline comprising of buildings getting higher closer to the center of the city, each high-rise zone is formatted upon calling its generate appearance function by comparing its location to the location of the city center. Depending on how close the two points are, there is a higher chance that a higher building model is placed. The chance that a higher building is placed is determined by calculating borders within which certain heights of buildings can generate. Within these borders it is much more likely that a higher building will generate, but not guaranteed to ensure an uneven skyline is created.

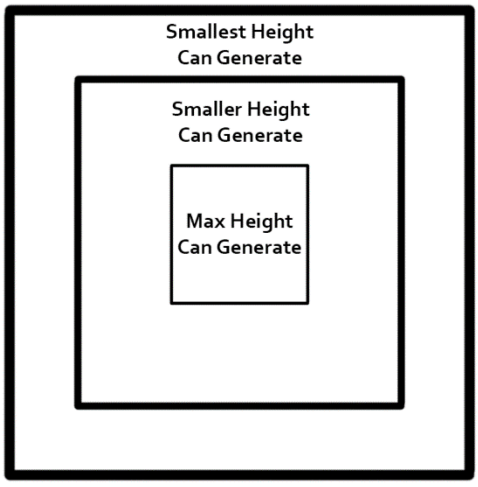


Figure 4.12: Examples of boundaries dictating where certain heights of high-rise tiles can start generating.

4.3.4 Industrial and Green Spaces

These zones are relatively simple to generate the appearance of: each tile that makes up the zone rolls a random number and places one of a selection of models themed on the zone in that space in the city based on the number:

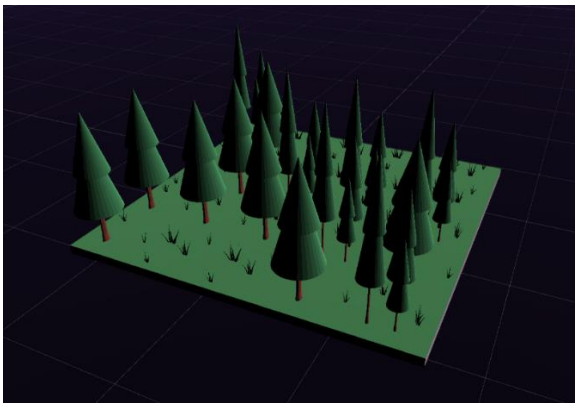


Figure 4.13: An example of random green-space tiles placed together to form a green-space zone. The same method is used for industrial spaces.

4.3.5 Roads

Roads can either be straight, a bend, a 3 way junction or a 4 way junction. A separate model is created for each possibility:

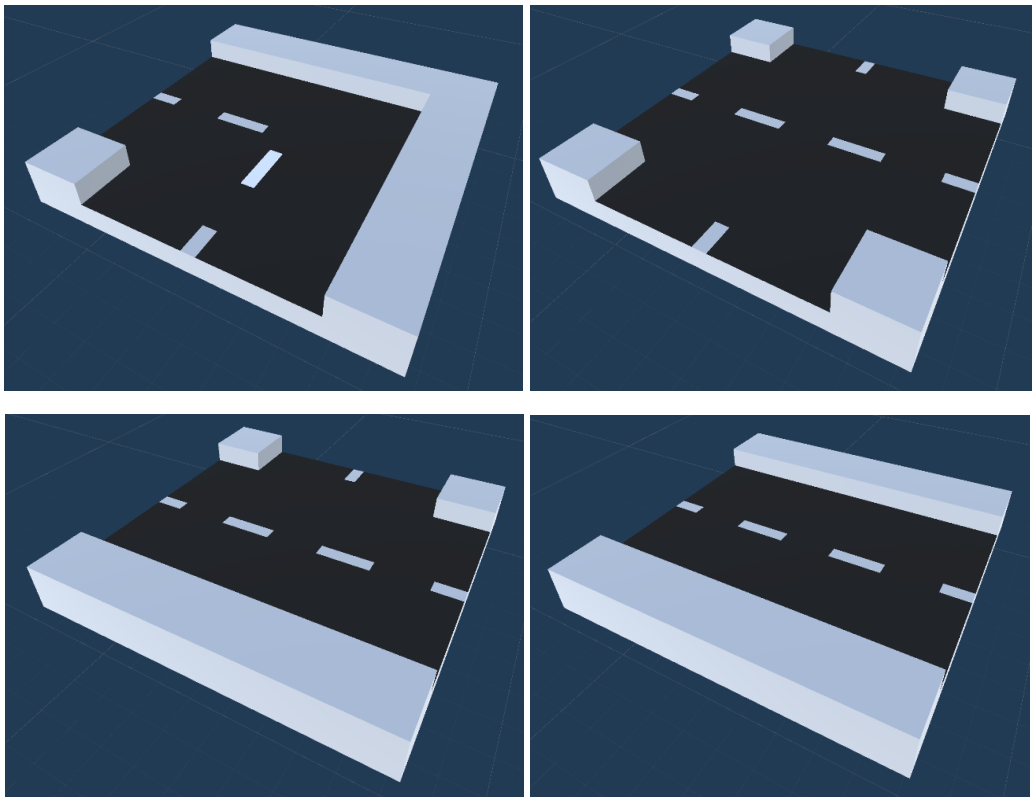


Figure 4.14: All possible road prefabs.

When a road's appearance is generated, its cardinal neighbors are checked (the positions north, south, east and west of the road). The number of neighbors that are either a road or are null (and therefor will be initialized as a road in the future) is summed, and the number indicates the kind of road it is.

2 roads or null = straight road **3 roads or null = 3-way junction** **4 roads or null = 4-way junction**

Upon checking each cardinal neighbor, the direction checked is also stored as a Boolean value indicating if road/null was in that direction. This data can then be used to decide upon the direction of the road.

The corner roads and roads added at the end of each row and column mentioned in section 3.1 are added in a separate loop outside the object definition - as they are deterministic and as a result occur outside the data structure's bounds.

4.4 User Interface

4.4.1 Camera Control

To allow the user to choose their own position to view the city generated they must be able to move the camera's coordinates and rotation. To create a camera that can freely look through the city, a script component is attached to the scene's camera. Within this script the update function (which is called every frame by default in Unity) checks for key pressed that correspond to the controls of the camera. The controls for the camera are WASD/arrow keys, and the position of the camera is transformed while the corresponding key is pressed using deltaTime to continue the movement over time. For example moving the camera forward.

```
if (Input.GetKey(KeyCode.W) || Input.GetKey(KeyCode.UpArrow))
{
    transform.position += (transform.forward * camMovementSpeed * Time.deltaTime);
}
```

Figure 4.15: Line of code that moves the camera position gradually using deltaTime.

The use of transform.forward allows the camera to move not along an axis, but relative to where the camera is pointing. To move the camera, the mouse axis are used, though movement only occurs while a mouse button is held down so the UI can be accessed without changing the view.

```
if (Input.GetKeyDown(KeyCode.Mouse1))
{
    StartLook();
}
else if (Input.GetKeyUp(KeyCode.Mouse1))
{
    StopLook();
}
```

Figure 4.16: Logic to decide when to move and stop moving the camera.

4.4.2 On-Screen UI

The following features need to be controlled by the user via an on-screen UI:

- Specify city generation parameters and initiate generation.
- Control lighting direction and colour.
- Hide and show the UI.

To display the UI a canvas is created in the scene, which is drawn in 2D across the top of the camera view. From this canvas buttons, tags and text boxes are placed, aligned with corners to ensure that the UI remains in position with different resolutions.

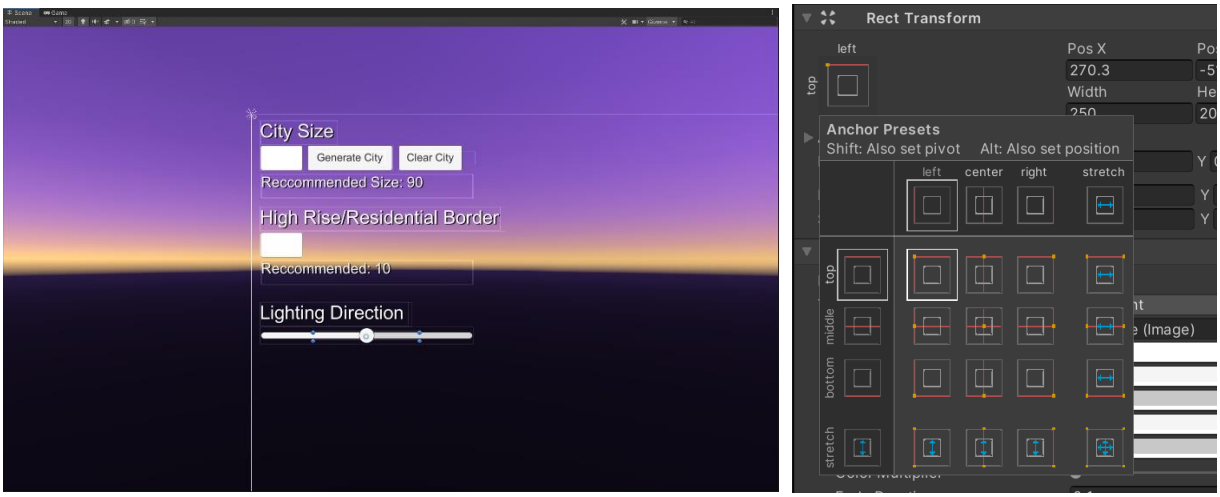
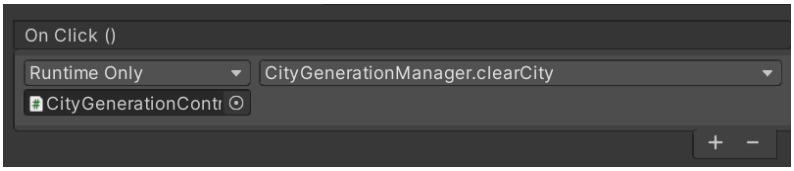


Figure 4.17: Anchoring UI components to a corner for scaling.

Passing in variables to the city generation manager from the UI can be done via the onclick component of the button, and by referencing the input component of the text boxes within the city generation script:




```
InputField citySizeUserInputField = GameObject.Find("CitySizeInput").GetComponent<InputField>();
citySizeUserInput = int.Parse(citySizeUserInputField.text);
```

Figure 4.18: On click event to get the value from a text box within the UI canvas.

4.4.3 Lighting Control

A slider controls the colour and direction of the lighting on the city. Within the update function of a script attached to a directional light, the values of these sliders are obtained by using Find and the silders' unique name.

```
LightSlider = GameObject.Find("LightingSlider");
```

Figure 4.19: Finding a UI object by its name.

To change the direction of lighting the silder has a value from -180 to 180, and the localEulerAngle of the directional light is changed accordingly.

```
this.transform.localEulerAngles = new Vector3(50, LightSlider.GetComponent<Slider>().value, 0);
```

Figure 4.20: Transforming the angle of the lighting object based on the value of the silder.

Similarly, separate sliders are used to control the individual RGB values of the light component:

Figure 4.21: The RGB sliders, formatted for visual clarity with corresponding-coloured controls.

Finally a script is created and attached to the canvas to check for a key press that will toggle on or off the visibility of the canvas to allow the user to clear the screen for viewing/taking screenshots.

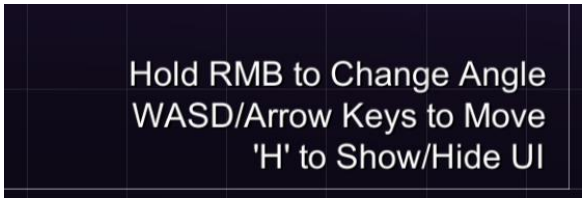


Figure 4.22: Code to hide and show the UI by getting the canvas element of the controller script and setting its enabled attribute.

5. Testing

I will be utilizing two forms of testing to ensure the Cityscape Generator adheres to the goals it set out to accomplish:

- Manual unit testing to ensure correct functionality of each individual component of the system, including an evaluation of the UI's effectiveness.
- A user survey to gauge the appeal of the output to a wider audience and gain feedback on the system from the point-of-view of a consumer of the output.

5.1 Manual Unit Testing

5.1.1 Unit Testing Plan

To ensure that city layouts are generated according to the algorithm outlined in section 3.2, that the city is output visually as intended and that the UI has all necessary functionality; I will carry out unit tests on each.

I have decided to use manual unit tests because the system output (in regard to the procedural placement of land zones and the formatting of buildings within) will vary greatly between tests due to the randomized nature of the algorithm. As a result the output does not necessarily have objectively measurable success and failure conditions.

5.1.2 City Generation Tests

Testing the algorithm to generate the city will require a set of input conditions tweaking the generation variables between all accepted values, including values at the extremes of the accepted boundaries. Erroneous values will not be tested here as the user's ability to input unacceptable variables will be evaluated in the UI tests. To facilitate these tests only the city generation script is placed in a scene, with testing values manually entered into the script and all visual formatting replaced with simple colored cubes to represent the output. This is done to ensure that testing is done only on city generation with no interference from other components.

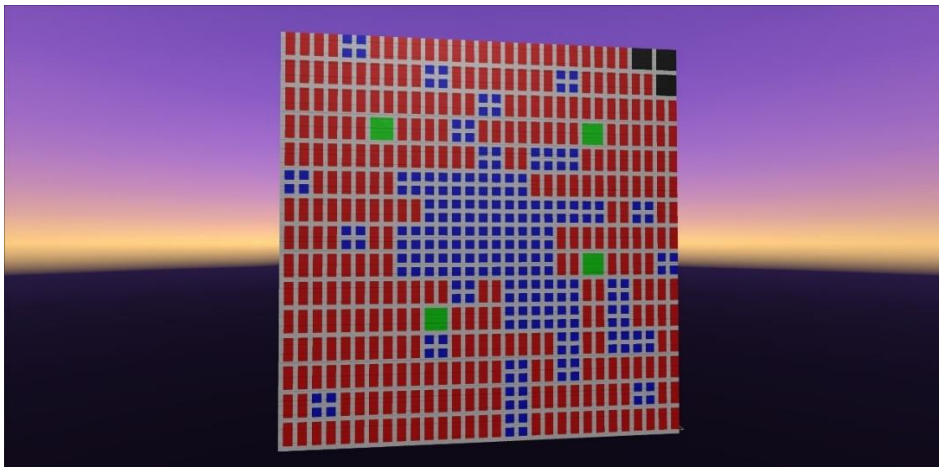


Figure 5.1: Example of the unit testing output created from the manual city generation unit tests.

With each test I will manually evaluate the output to determine how successful and true to the input variables the result is. The results are displayed in the test table within appendix C.

From the testing done, it seems that city generation is very successful for most sets of input generation parameters. Land zones generated according to the rules set for them and scaled as expected as the size of the city fluctuated. The only major issue encountered is that very small city sizes can lead to green-zones not generating as the interval to generate a green-space template does not fall within the bounds of the city.

5.1.3 City Visual Formatting Tests

As the city's visuals are dependent on the generated city layout, it is impossible to truly unit test this component alone. However decomposing the 'generate visuals' function on each land zone object into the unique functionality that each land type has will allow a much more focused testing methodology. Thus I will manually test the appearance of each land zone in every condition it can be in:

- Roads bordered by 2, 3 and 4 other roads.
- Residential areas facing either east or west.
- Industrial zones in a corner generating each possible industrial model.
- Green spaces generating each possible green space model.
- High rise zones at different distances from the center of the city.

From the testing grid in appendix D it appears every land zone is formatting its visuals as is expected, with two exceptions. Roads bordering the city to the east are being generated deterministically as they are on the south, however there is actually some variance in the land zones they can border and thus sometimes roads on this side will form 3-way junctions when they should not. Additionally there is a minor rendering issue with shadows under the roofs of residential zones, which may be due to the model's normals being calculated inside.

5.1.4 UI Tests

To test the UI I will walkthrough how a user would interact with the UI, while attempting to input both expected values and erroneous values. I will also attempt to perform illegal actions such as generating a city when one has already been generated. All cases of erroneous actions should be met with appropriate exceptions, and all expected inputs should result in expected behavior.

Upon launching the Cityscape Generator, the user is presented with a blank environment with the UI as described in section 4.4.2.

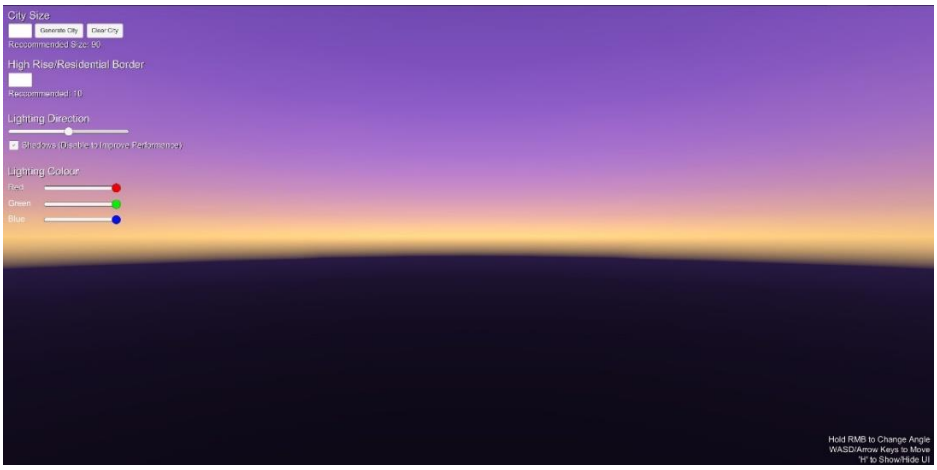


Figure 5.2: The UI as it appears upon first launch of the system.

To ensure that the UI scales based on resolution I also scaled the window down to 1920x1080, as the current screen is 3840 x 2160.

Figure 5.3: The UI as it appears upon first launch of the system, scaled down to a lower resolution

The clarity of the on-screen text appeared to decrease slightly as the UI is downscaled, but the on-screen elements functioned correctly despite this.

Initially I attempted to input some erroneous values into the text boxes, first in the form of non-integer inputs. With a string input the city generation button would not generate a city, but the program continued to run.

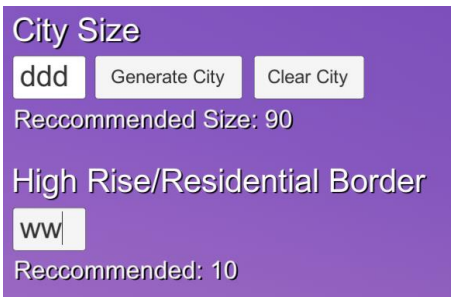


Figure 5.4: Inputting incorrect city generation parameters.

I also input values too large for the integer input, and got a corresponding error message as intended.

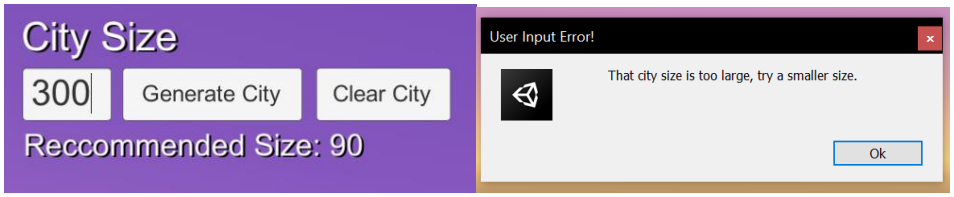


Figure 5.5: Inputting incorrect city generation parameters.

To ensure that the UI was passing data to the city generation algorithm correctly I generated a city using the recommended settings.

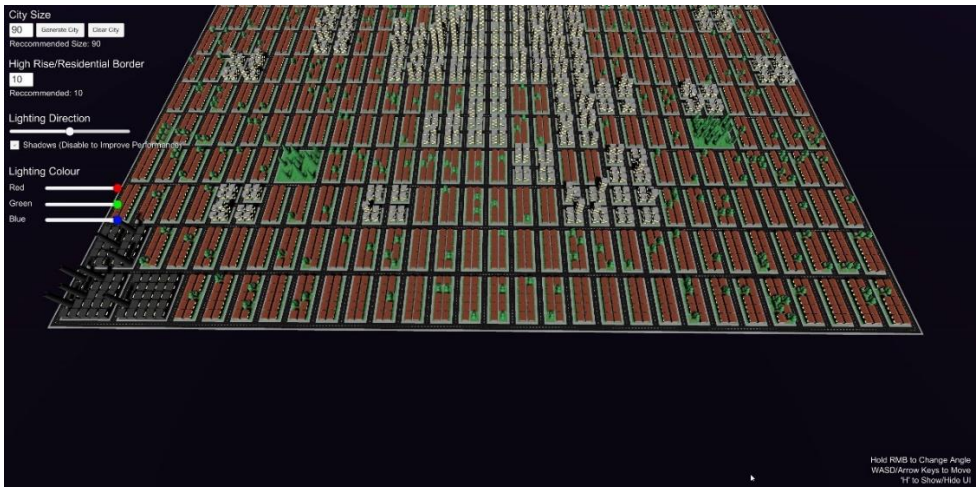


Figure 5.6: A city is generated.

With these recommended settings a city of the correct dimensions of 90 x 90 and a standard distribution of residential to high-rise buildings was created. However, further tests are needed to ensure the data from the UI is being passed in - so I also input a variety of numbers around the borders of permitted values.

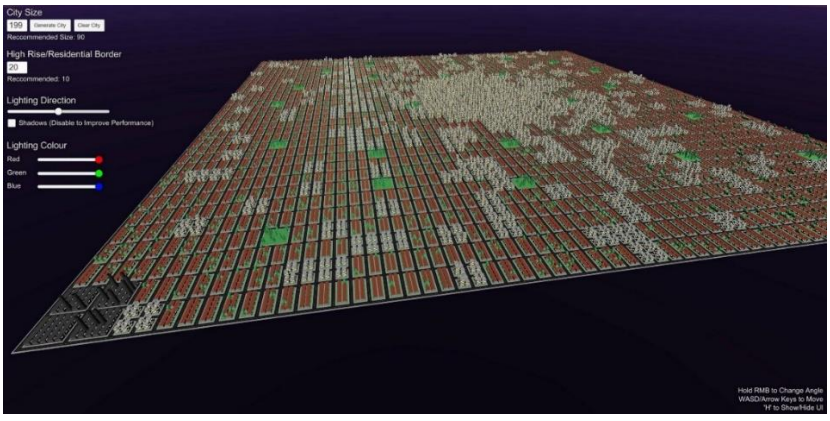


Figure 5.7.1: A larger city is checked for the correct sizing.

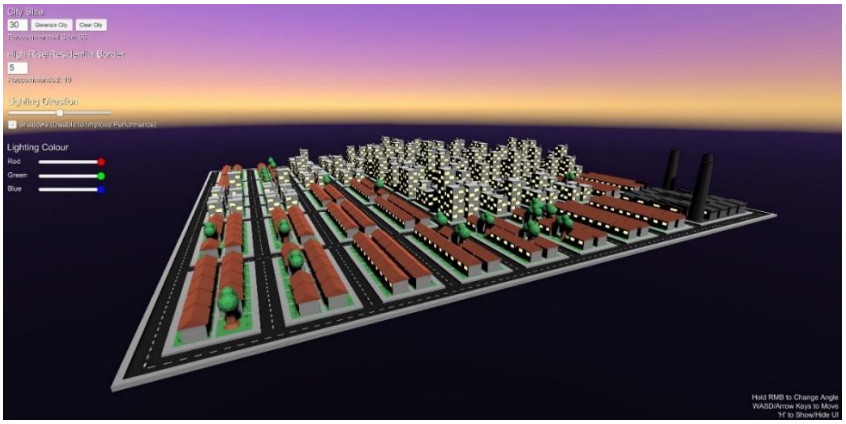


Figure 5.7.2: A smaller city is checked for the correct sizing.

As shown by the above figures, the values input by the on-screen UI scale the generation algorithm accordingly. During this testing, the functionality of the clear button was also tested on these same sized cities. The button was able to clear the city layouts and prompted users to click it before generating another.

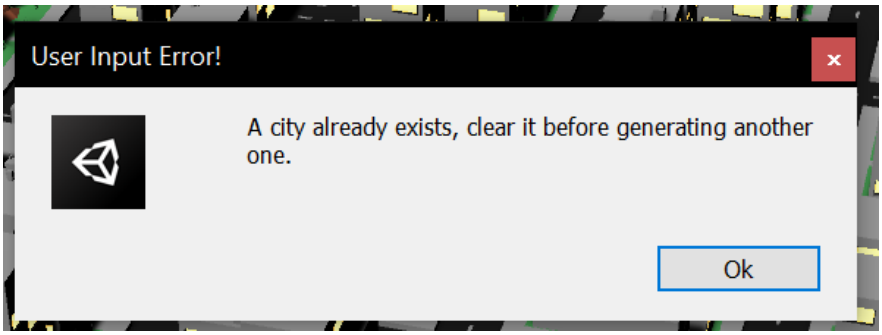


Figure 5.8: An error is displayed, prompting the user to clear the current city.

Finally the visual options available to the user had to be tested. Firstly toggling shadows on and off.

Figure 5.9: Shadows toggled on/off.

Shadows could be disabled and re-enabled as required. I used the direction of shadows to test the lighting direction slider too.

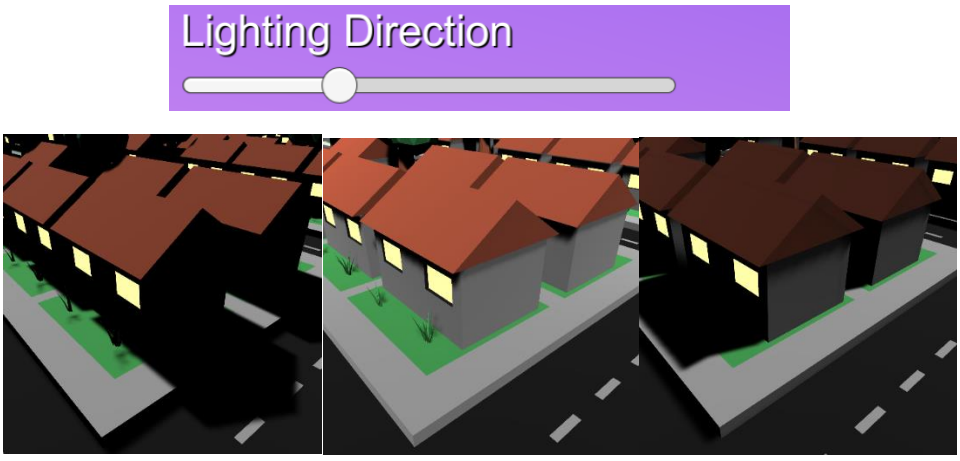


Figure 5.10: Changing the lighting direction.

Finally I did some testing on the lighting colour selection, attempting to use different levels of light and different balances of colour.

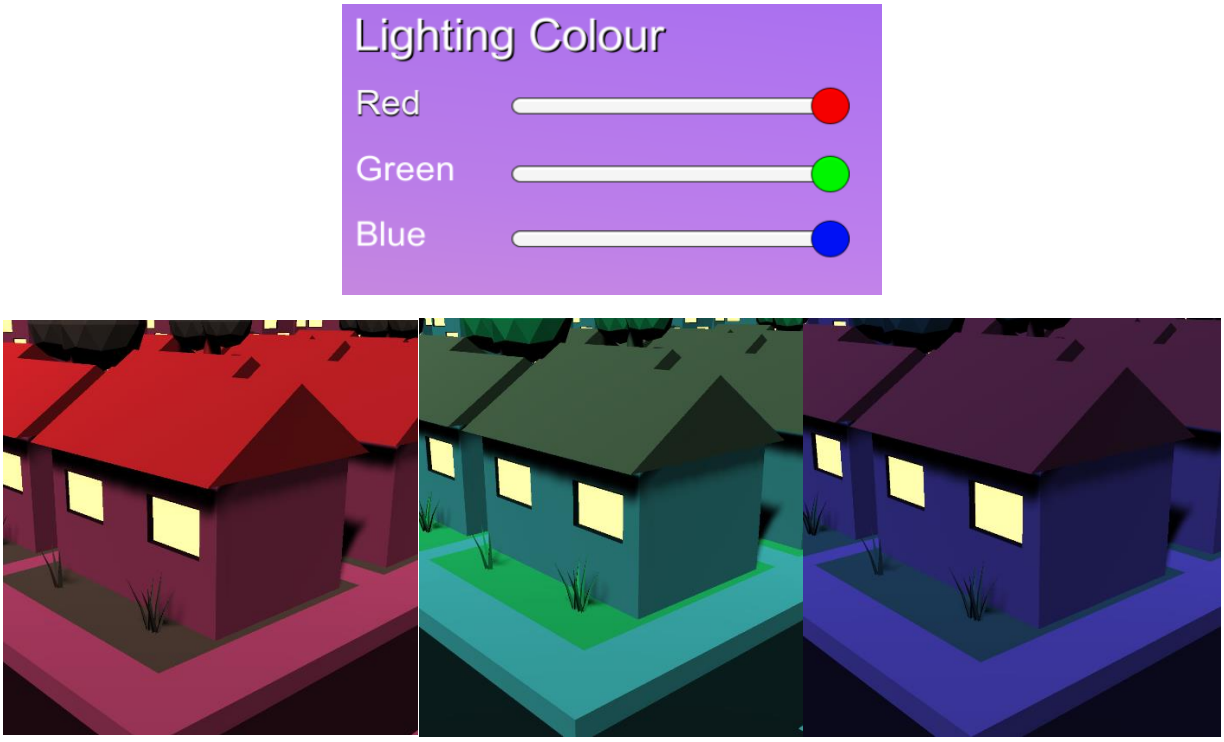


Figure 5.11: Changing the lighting colour.

The lighting colour's RGB value changed as the sliders were altered based on the colour of the slider changed, and thus UI has complete functionality that is working as intended.

5.2 User Survey

To assess the success of the system's output I decided to create a survey that would ask participants for their opinions on 5 varied images of a generated cityscape (see appendix X for the images and survey) along with asking for a participant's favorite and least favorite aesthetic features of the images. The images were rated on a scale of 1 to 10. As aesthetics are largely subjective, getting a group of people that are not biased towards the system would get a more accurate representation of the output's success than my own assessment.

Images focusing on green-spaces and high-rise areas were the most positively received, such as the image below:

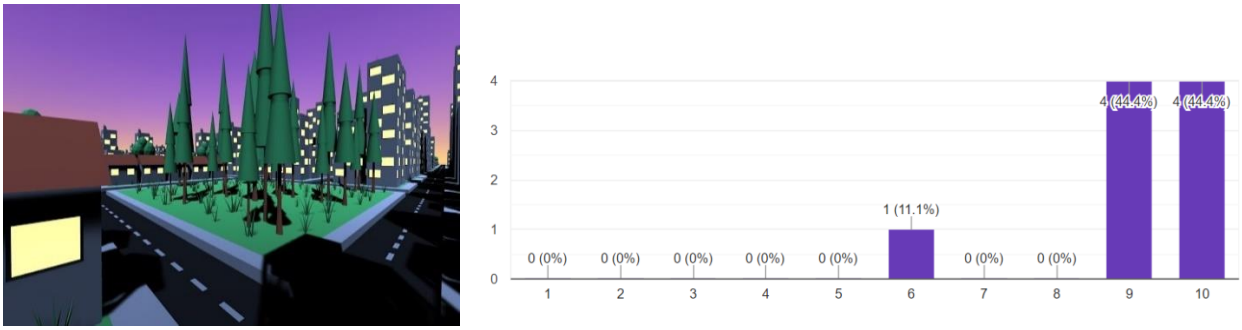


Figure 5.12: Results of the survey of the image to the left of this figure.

Whereas far-shots of the city with more emphasis on the industrial areas were least liked by participants:

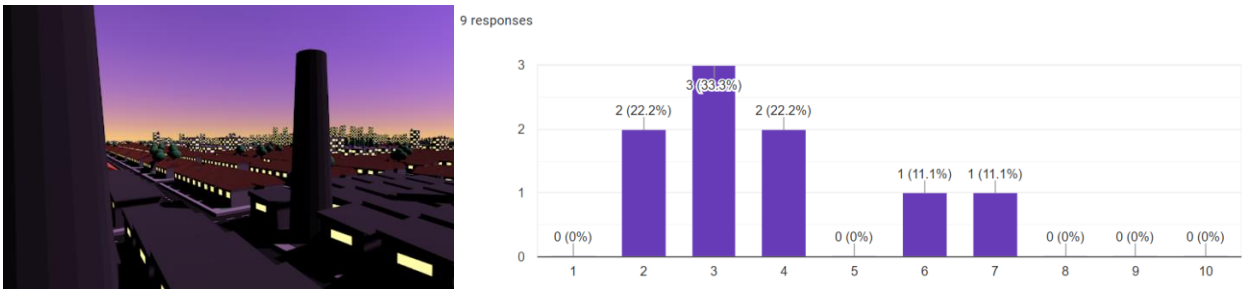


Figure 5.13: Results of the survey of the image to the left of this figure.

Additionally, from the question asking what participants favorite and least favorite aesthetic features of the city were, most positive points echoed what the image in figure 5.12 shows. However, a negative point was repeated multiple times; the shadows being cast in the city were too dark and harsh:

I really enjoy the colour of the sky but I think the shadows are a bit harsh.

I loved the lighting from the windows. The factory parts were a bit dull.

I like the long distance views of the big buildings in the city's centre! The shadows feel sorta gloomy though.

Figure 5.14: Feedback on participants' favorite and least favorite aesthetic features.

From these results it can be concluded that the basic buildings of the city were quite successful such as residential and high-rise areas. The alternative lighting of the image in figure 5.12 was especially favorable, as were the addition of luminous windows. However the rendering of the sun's light does require work as currently a major complaint is the dark shadows being cast. Additionally the industrial zones were not seen positively and may require further variations of factory models, though these make up a very small amount of the city layout and thus are not a pressing issue.

5.3 Project Aims

5.3.1 Primary Aims

- **To explore procedural generation algorithms and create an algorithm that can generate a city layout based on research done about real land zoning principles.**
  The research and design of the city generation algorithm were an extensive part of this project, and I feel that this focus on these aspects lead to a successful end result. The progression from the aim to the research done and the final system was well documented.

- **To research different 3D graphics engines and their unique features to select an appropriate engine to develop the system in.**
  While I do believe that the engine chosen to develop the system in was an appropriate choice and provided the functionality required for the system, I also think that the research done on different choices could have been more comprehensive.

- **To create a method to convert a generated city layout into a 3D output with various aesthetic additions in order to allow the output to be used as a graphic in content.**
  The results of the user survey (see appendix F) in addition to the successful unit tests of the system's output indicates that the system can successfully create a cityscape that is aesthetically pleasing enough to use in content.

- **To define and utilize professional-standard development methodologies/technologies to streamline the development of this project.**
  The progression of the project and structure of development (designs informed by prior research, and development based on these designs) lead to what I consider a successful translation of the initial specification to the end result. Though a more rigid and pre-planned methodology may have been beneficial.

- **To evaluate the success of the system in regard to creating graphics that can be used by content creators.**
  The myriad of tests performed on the system's output such as user surveys, unit testing and a UI walkthrough have comprehensively evaluated the graphics generated by the system and the system's usability for content creators.

- **To critically analyze the development/research process upon completion and find areas for improvement.**
  The critique of the process within section 6 is extensive and I believe identifies a number of relevant positive and negative points that arose throughout development. The suggestions for future improvements were well justified and feasible within the scope of the system.

5.3.2 Unachieved Aims

While the initially identified project aims were all met there are some features that were identified in the project specification that were omitted from the project aims and thus were not achieved within this project:

- **Ability to export and load city layouts.**
  In the current system the user is required to generate a new city upon every new session, and should the user wish to generate a new city they must discard the current one. This functionality was omitted from the project aims and not developed due to the additional time and complexity required to develop a file structure to read and write generated city layouts.

- **Additional skybox objects such as clouds.**
  Currently the skybox is a simple gradient that does contain a directional light somewhat emulating the appearance of the sun. However clouds were never implemented due to time constraints and the additional strain they would place upon systems running the Cityscape Generator. With large city layouts there is already a large requirement of processing power; clouds - even with a basic implementation of randomly placing them in a similar fashion to land zones - may have severely exasperated this issue.

6. Critical Evaluation

In this section I will evaluate the successes and failures of this project, in terms of both the final product and the process used to research, develop & design the product.

6.1 Deliverable Successes

The deliverable of this project has met all of the aims it set out to during the research section and its outputs received very positive feedback in user surveys (see appendix F). Analyzing existing technologies and citing professional-standard academic reports and sources during the research stage gave the system some legitimacy and ensured that there was a strong knowledge base to build the design and development sections upon. Additionally the development of the system was well documented and justified decisions made in the system's code that were not foreseen during the design stage. The final system would meet the needs of the persona identified in section 2.1 and thus appeal to its intended userbase. The initial question of the survey asking participants how much they consume digital content proved that participants were all active viewers of such content and thus were ideal for the survey as shown by the figure below.

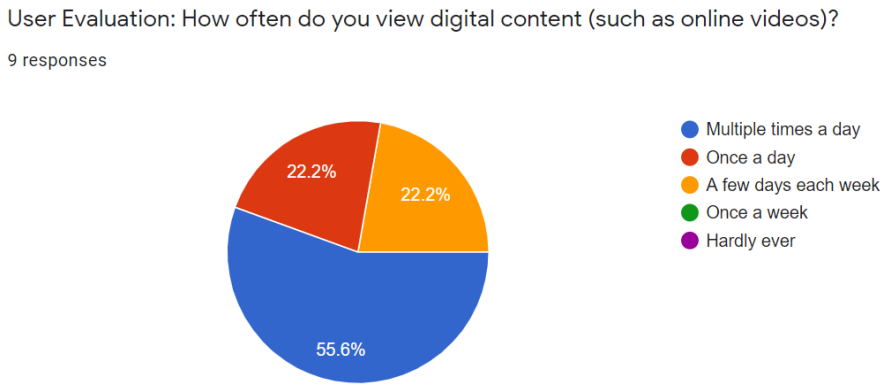


Figure 6.1: Results of the survey showing that the participants asked were relevant to the persona in section 2.1.

Development went smoothly with designs such as the UI wireframe and algorithm outline being realized fully. Use of the Unity engine proved to be a good choice as it facilitated all the functionality required. Furthermore, testing not only proved the successes of the system's development but also identified accurately the project aims that were met and the optional functionality that was not realized in the current deliverable.

The development of the Cityscape Generator has also been an excellent learning experience for me; within which I have had the opportunity to not only apply my existing interest in 3D environment development but also learn how to effectively research and plan a software development project. Starting the project I was unsure on how to develop a procedural generation algorithm or what constitutes a well-planned city, so it was immensely satisfying to see my knowledge of these topics progress as the system developed. Extensive use of the Unity documentation and software development tools/methods like UI wireframes and Trello benefitted the progression of the project greatly as they allowed me to keep a clear plan and

vision in mind moving forward into the development stage. The skills I have learned in performing academic research to inform development and in utilizing various software development technologies/techniques have helped me improve as a computer scientist. I will be able to apply this newfound knowledge in future endeavors, and even the mistakes made have been useful as a learning experience.

6.2 Deliverable Limitations

The project has been successful, but it also has room for improvement. Many of the features identified in other similar systems during research that would have been useful for the system were not able to be implemented due to time constraints. One survey participant noted that a particular city layout they assessed would have looked better with some changes to the positions of buildings which is a feature identified in section 2.2 but never implemented. Better management of the project may have allowed for such features to be implemented, as the timeline outlined in the project spec (see appendix B) was not adhered to - leading to a much smaller development window to make time for full completion of research and design. The Trello board, while useful for the development of the system, should have been created in tandem with the project timeline to allow for correct time management. There was not a great deal of unexpected issues during development, but one issue that was found is the poor performance of the system with large city layouts. This required the implementation of size limits and a shadow toggle setting. With better time management more elegant solutions to these problems may have been found.

The survey done provided some useful information on aesthetic features that could be improved, however the low number of participants and generalized answers to the more specific questions in the survey made assessment of the results quite speculative.

Additionally some aspects of the project's research and design could have been more comprehensive. The user group the system was designed for should have been more clearly defined as the persona created within the research section only conveys a nebulous idea of what a user may want out of the system and is not backed up with any research beyond this. The class diagram created for specializing a generic land zone class into each different zone that can generate also proved to be quite unimportant as the restrictions of scoping within the city generation code required a drastic diversion from this original plan.

6.3 Ethics

The system does not store any personal information on users and thus data protection and the GDPR were not researched or heavily considered during its development. However other data concerns must be considered for future iterations of the system. If the system were to have the identified functionality of storing city layouts for later use, the security of these saved files would need to be considered so that they do not pose a risk to a user's device. Additionally the use of the system's output

in online content that could be monetized means that an actual release of the Cityscape Generator would require licensing of the output.

Analysis of the project's success did require a user survey, and I feel that all ethical concerns were thoroughly considered and relayed to participants. The consent form and survey brief conveyed all necessary information to survey responders regarding the data that will be held on them and the contents of the survey.

Finally, the system could have done better in regard to disability support. While the customization of lighting colour allows colour-blind users to tailor the output to their condition and camera controls allow sight-impaired users to amplify details as they need; the text elements of the UI are in a Unity canvas and thus cannot be read by an on-screen reader. Furthermore, additional keyboard shortcuts for controlling on-screen UI elements such as buttons or sliders would help physically impaired users utilize their preferred input peripheral to interact with the system.

6.4 Future Aims

6.5.1 Important Improvements

While most of the project aims were met there are some issues with the system as it currently exists that would need to be addressed in future iterations:

- With large city layouts the system performance is reduced, and to improve it the user must sacrifice graphical fidelity in the form of toggling on/off shadows. Additional graphical options should be included such as occlusion culling to improve performance without reducing the output.
- Some users identified (in the user survey, appendix E) that the shadows of the cityscape were harsh and did not adapt well to the colour of lighting being projected onto the city. Improving on this is important as the output of the system is purely visual. Integrating the shadows cast with the lighting colour sliders could help alleviate this concern.

6.5.2 Potential Added Functionality

Some features were identified within section 2.2 from other similar systems that may be beneficial to the system in future iterations, although they did not fall under the scope of this project:

- The ability to edit a generated city layout to fine-tune the output. Tools to change land zones that are not in the location a user requires or options to reroll randomly generated tiles such as those within greenspaces would help users create their perfect scene.
- Options to generate different shapes of city layouts, as currently the system only produces square cities with a city-block based structure. More freeform city planning paradigms with roads and buildings that do not perfectly conform to a grid pattern could provide many different compositions of cityscapes that are currently not possible.

7. Appendices

A. References

W3 Schools (2020): Operating System Usage Statistics, Retrieved 26th October 2020 from

https://www.w3schools.com/browsers/browsers_os.asp

Unity (2020): Unity System Requirements, Retrieved 26th October 2020 from

https://docs.unity3d.com/Manual/system-requirements.html

Unreal (2020): Hardware Requirements of Unreal Engine, Retrieved 26th October 2020 from

https://docs.unrealengine.com/en-US/GettingStarted/RecommendedSpecifications/index.html

PC Game Benchmark (2020): System Requirements of the Godot Engine, Retrieved 4th November 2020 from https://www.pcgamebenchmark.com/godot-engine-system-requirements

Net Marketshare (2020): Operating System Market Share, Retrieved 26th October 2020 from

https://www.netmarketshare.com/operating-system-market-share.aspx?id=platformsDesktopVersions

GOV.UK (2020): Planning Permission England & Wales, Retrieved 31st October 2020 from https://www.gov.uk/planning-permission-england-wales/when-you-dont-need-it

Planner's Web (2001): Zoning Basics, Retrieved 27th October 2020 from

http://plannersweb.com/2001/04/zoning-basics/

Michael Allan Wolf (2008): The Zoning of America: Euclid v. Ambler. Book.

Nadja Kabisch (2014): Green justice or just green? Provision of urban green spaces in Berlin, Germany. Research paper.

Alessio Russo & Giuseppe T. Cirella (2018): Modern Compact Cities: How Much Greenery Do We Need? Research Paper.

Britannica (2011): Manila History, Retrieved 31st October from https://www.britannica.com/place/Manila/History

United Nations ESCAP (2019): Ocean Cities Regional Policy Guide. Manual/Guideline.

Godot (2020): Godot MIT license, retrieved 4th November 2020 from https://github.com/godotengine/godot/blob/master/LICENSE.txt

Godot (2020): Godot Engine Features, Retrieved 4th November 2020 from https://godotengine.org/features

New Gen Apps (2018): Pros and Cons of Unity, Retrieved 9th November 2020 from

https://www.newgenapps.com/blog/unity-3d-pros-cons-analysis-choose-unity/

Unreal Engine (2020): Shader Development, Retrieved 9th November 2020 from https://docs.unrealengine.com/en-US/Programming/Rendering/ShaderDevelopment/index.html

Maxis (1999): SimCity 3000 Unlimited. Published by EA, Video Game.

Colossal Order Ltd. (2015): Cities: Skylines. Published by Paradox Interactive, Video Game.

Watabou (2020): Medieval Fantasy City Generator. Retrieved 11th November 2020 from https://watabou.itch.io/medieval-fantasy-city-generator

B. Project Specification

PROJECT SPECIFICATION - Project (Technical Computing) 2020/21

Student:	**Christopher John Noroozi** **28014084**
Date:	**23/10/2020**
Supervisor:	**Michael Meredith**
Degree Course:	**BSc Computer Science**
Title of Project:	**Realistic Cityscape Generator for Content Creators**

Elaboration

This project is about designing and creating a system that generates a layout of a city and displays it in 3D with various aesthetic filters applied. It is intended to be used by a wide variety of content creators (such as video game developers or artists) to create graphics/backgrounds for their content. For example, an online video creator may want a unique background for their intro/outro segments that could be generated by the system.

The system will utilize procedural generation that considers zoning laws used in actual city planning in order to create a realistic cityscape. Zones such as residential, industrial, suburban and retail will be generated with green spaces and other important infrastructure added within a realistic range. As such I will undertake research into how zoning is undertaken in real cities and is outlined in professionally referenced city planning documentation.

The users will be able to change a number of settings to display the cityscape such as changing the colour of buildings in different zones, altering directional lighting, applying luminosity to building windows or changing the background gradient. Users can also change options upon generation such as the size of the city.

I will need to research not only how land is zoned in actual city planning in order to generate a realistic city, but also 3D technologies to produce an aesthetically pleasing output using a 3D engine that is fit for purpose. I will document my research and how it affected the decisions I made while developing the system.

Other optional but potentially useful functionality for the system include objects in the skybox (such as the sun or clouds) to add to the aesthetic effect, a set of curated colour pallets users can choose from if they don't wish to create their own or the ability to export generated city layouts.

I will evaluate the system's effectiveness by two metrics: the aesthetic appeal of the output and the accuracy of the city in regard to how the zones of the city have been generated. A successful output would be customizable to suit a variety of colour pallets and aesthetic moods, while also having a uniform and realistic distribution of different zones of the city. I will also create some example mock-ups of applications of the output from the system and show them to an anonymous selection of people to judge how aesthetically pleasing they find it.

Project Aims

- Research the principles used in real city zoning in order to generate a realistic city layout.
- Develop a system that can generate and output a 3D model of a city that considers real city planning paradigms.
- Allow the inputting of various parameters to control procedural generation and output aesthetics.

- Create a set of assets (models and textures) that can be used as a simplified representation of aspects of a city.
- Allow the user to modify the visuals of individual zones within the generated city.
- Create an intuitive way of moving a user's view around a 3D environment.
- Further my understanding of procedural generation and displaying 3D graphics.
- Implement software development techniques such as deigning the system using UML, creating user stories and unit testing.
- Use a work-flow management tool during development such as Trello.
- Evaluate the practical usage and success of the system upon completion in regards to how it meets the specification and how much information it abstracts from real city planning.

Project deliverable(s)

- I will deliver a program to: generate a city layout, output it in 3D and allow the user to make visual changes to its output - that can be run on a Windows 10 computer.
- I will document the planning and development process of the project.
- I will create and document a testing method to ensure that the developed system works correctly.
- I will research the standards and rules that city planning requires to ensure that the output from the system is useful in a real-world environment.

Action plan

Task	Date Due
Begin researching project and create an information review • Research how land is zoned in real city planning. • Document research done on land zoning and city planning. • Research the relative benefits of different 3D graphics engines and the aesthetic features they provide. • Document research done on 3D development engines. • Create an information review outlining the sources I used for my research.	1/11/20 10/11/20 17/11/20 3/12/20 **4/12/20**
Develop prototype of system • Develop city generation algorithm. • Create placeholder assets. • Develop 3D city viewer based on layouts produced by the algorithm. • Create final city/road assets and polish UI. • Final polish & if time allows develop additional optional functionality such as skybox objects (like the sun) or the ability to save/load layouts.	14/1/20 17/1/20 14/2/20 21/2/20 **1/3/20**
Submit provisional contents page	**19/2/20**
Test prototype using testing plan	**15/3/21**
Draft the critical evaluation	**19/3/21**
Fully write up the evaluation • Evaluate the design and development phases of the project. • Compare the system to its original goals and evaluate its success. • Conclude the project and assemble the full report.	1/4/21 8/4/21 **14/4/21**
Submit the report and prototype system	**15/4/21**

Demonstrate project	< 29/4/21

BCS Code of Conduct

I confirm that I have successfully completed the BCS code of conduct on-line test with a mark of 70% or above. This is a condition of completing the Project (Technical Computing) module.

Signature: 

Publication of Work

I confirm that I understand the "Guidance on Publication Procedures" as described on the Bb site for the module.

Signature: 

GDPR

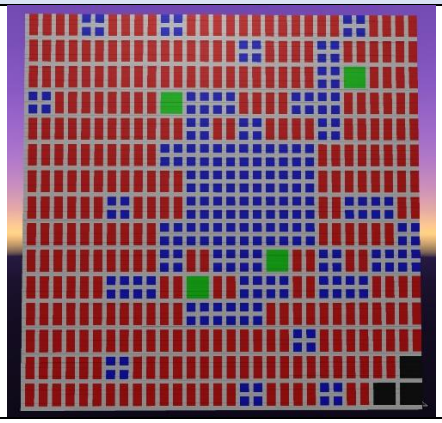
I confirm that I will use the "Participant Information Sheet" as a basis for any survey, questionnaire or participant testing materials. This form is available on the Bb site for the module and as an appendix in the handbook.

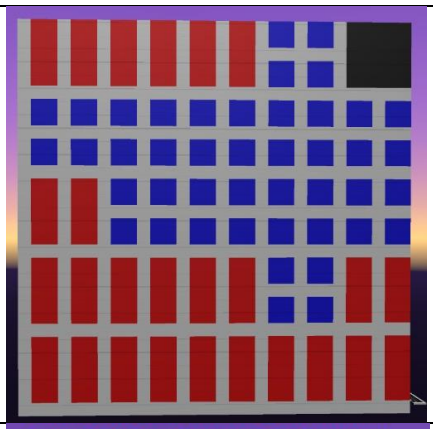
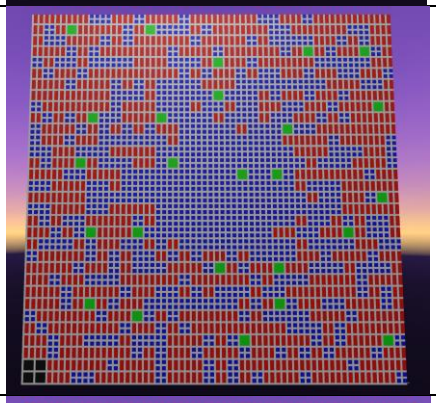
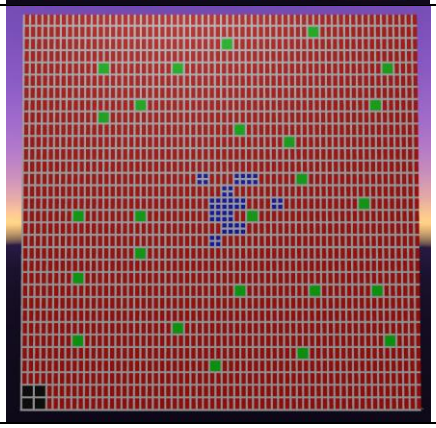
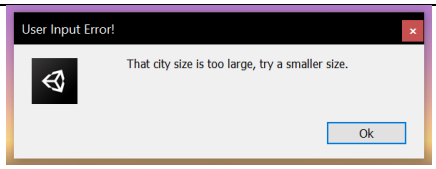
Signature: 

C. City Generation Testing Grid

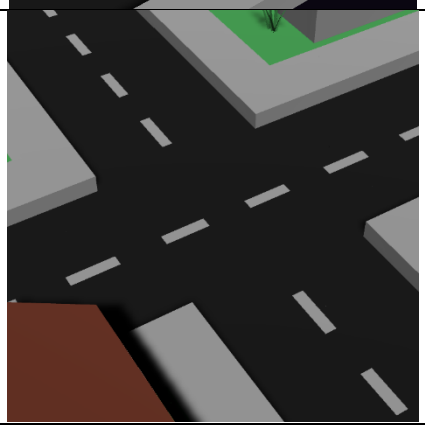
Within the following results:

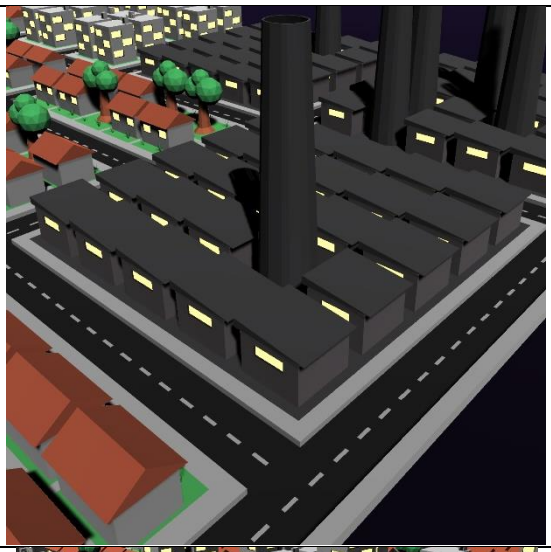
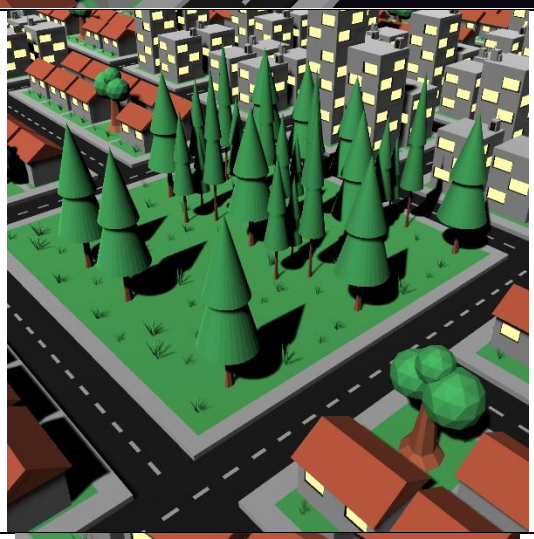
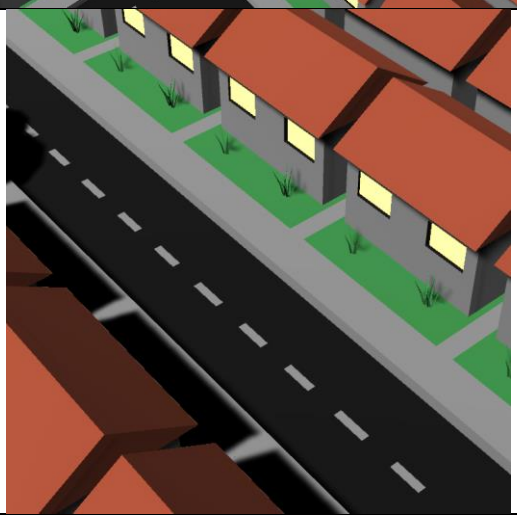
Red = Residential, Blue = High-rise, Grey = Road, Black = Industrial, Green = Green Space

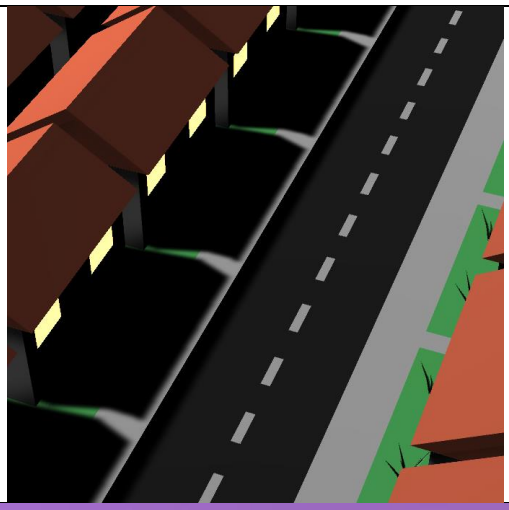
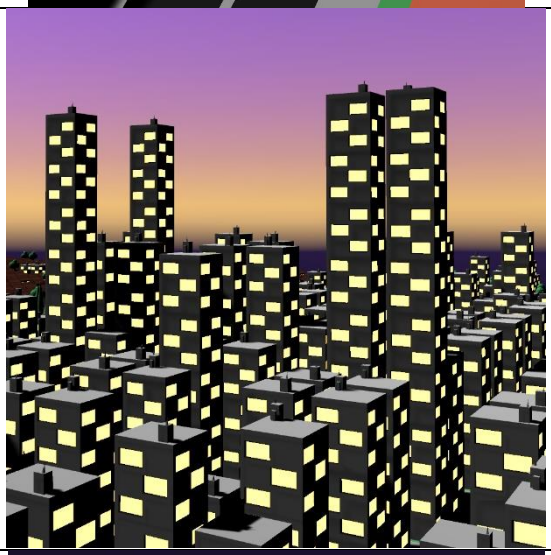
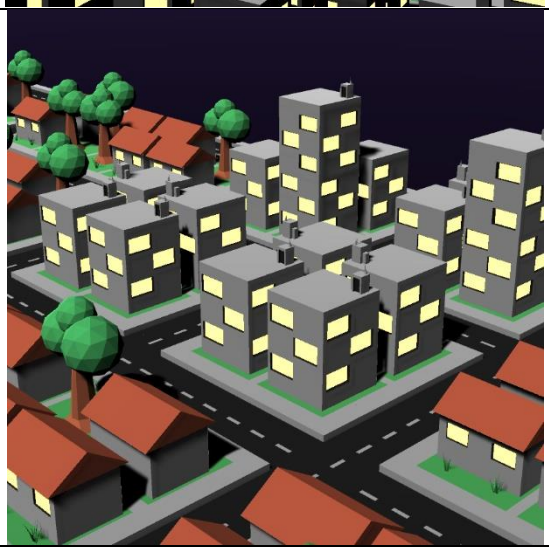
Test No.	Purpose	(City Size/High-Rise Ratio)	Result	Conclusion
1	Recommended settings input	90/10		A city of the dimensions 90x90 is created with: an even distribution of green spaces, a clearly defined city center that gradually becomes residential and industrial spaces grouped together. **The test was successful.**

2	Extremely small settings input	30/5		A small city of the dimensions 30x30 is created with: no green spaces, a definite city center surrounded by residential zones and an industrial zone in one corner. **The test was a partial success, however no green space generated as the city was too small.**
3	Extremely large settings input	200/30		A large city of the dimensions 200x200 is created with: many green spaces distributed throughout, a large sprawling city center and an area of industrial spaces in one corner. **The test was successful.**
4	Mix of extremely large and small settings	190/6		A large city of the dimensions 190x190 is created with: many green spaces distributed throughout, a very small but gradual city center and an industrial zone. **This test was successful.**
5	Erroneous settings syntax input	“Ten”/ ”Five”	*No city is generated.*	The inputs are not accepted, and no city layout is generated. **This test was a partial success, there was no error message to indicate incorrect syntax.**
6	Erroneous settings magnitude input	300/10		The inputs are not accepted and an error message indicating that “the city size input is too large” is shown. **This test was successful.**

D. City Visual Formatting Testing Grid

Test No.	Purpose	Result	Conclusion
1	Straight Road		Straight roads border zones correctly and connect seamlessly. **This test was successful.**
2	Corner Road		Corner roads connect roads generating at the corner of the city seamlessly. **This test was successful.**
3	3-way Junction Road		3-way junction roads connect roads in three cardinal directions seamlessly. **This test was successful.**
4	4-way Junction Road		4-way junction roads connect roads in all cardinal directions seamlessly. **This test was successful.**

5	Industrial Zones		Industrial zones generate a mix of the two models created for them with less chimneys than factory units. **This test was successful.**
6	Green Spaces		Green spaces are populated with a random and even assortment of trees and shrubs. **This test was successful.**
7	East Facing Residential		Houses bordering roads on the east face their windows towards the road. **This test was successful.**

8	West Facing Residential		Houses bordering roads on the west face their windows towards the road. **This test was successful.**
9	High-Rise Close to City Center		High rise buildings close to the city center are noticeably higher. **This test was partially successful, the gradient of buildings is not very smooth and may require more heights of buildings.**
10	High-Rise Away from City Center		High-rise buildings far away from the city center are a mix of the two shortest heights of buildings, primarily the shortest ones. **This test was successful.**

E. Survey Design

User Evaluation: How often do you view digital content (such as online videos)?

- ◯ Multiple times a day
- ◯ Once a day
- ◯ A few days each week
- ◯ Once a week
- ◯ Hardly ever

Aesthetic Evaluation: Bright Skyline *

Please rate how visually appealing you find this image of a generated cityscape from 1 to 10:

	1	2	3	4	5	6	7	8	9	10	
Not appealing at all	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Very appealing

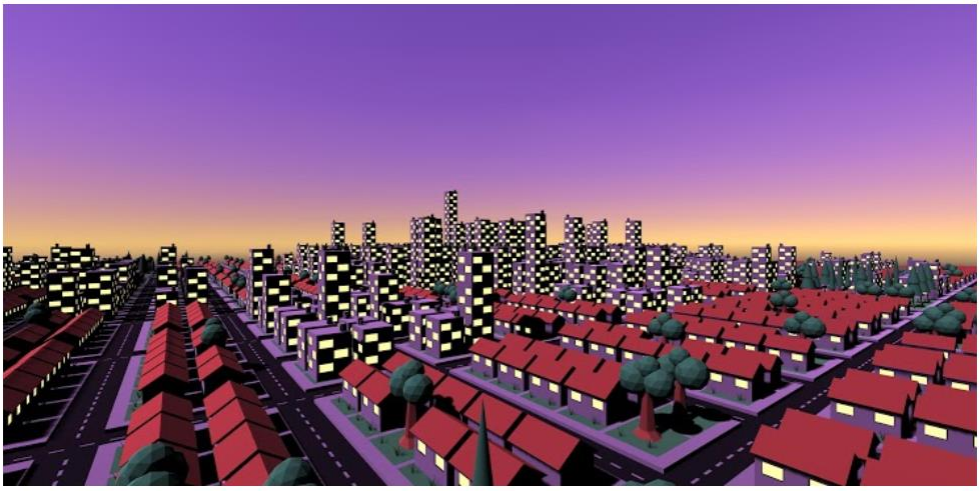
Aesthetic Evaluation: Dusk Lit Green Space *

Please rate how visually appealing you find this image of a generated cityscape from 1 to 10:

	1	2	3	4	5	6	7	8	9	10	
Not appealing at all	○	○	○	○	○	○	○	○	○	○	Very appealing

Aesthetic Evaluation: Purple Far Shot Skyline *

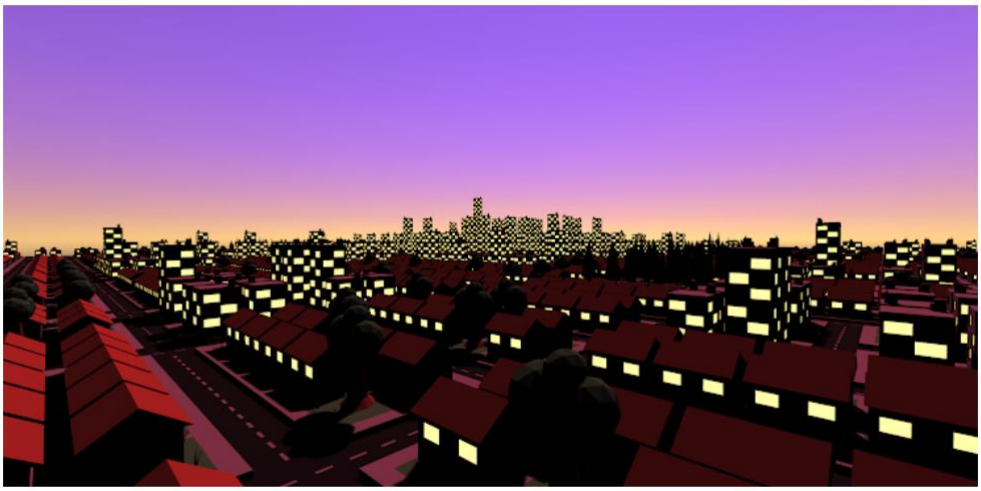
Please rate how visually appealing you find this image of a generated cityscape from 1 to 10:



	1	2	3	4	5	6	7	8	9	10	
Not appealing at all	○	○	○	○	○	○	○	○	○	○	Very appealing

Aesthetic Evaluation: Red Extreme Shot Skyline *

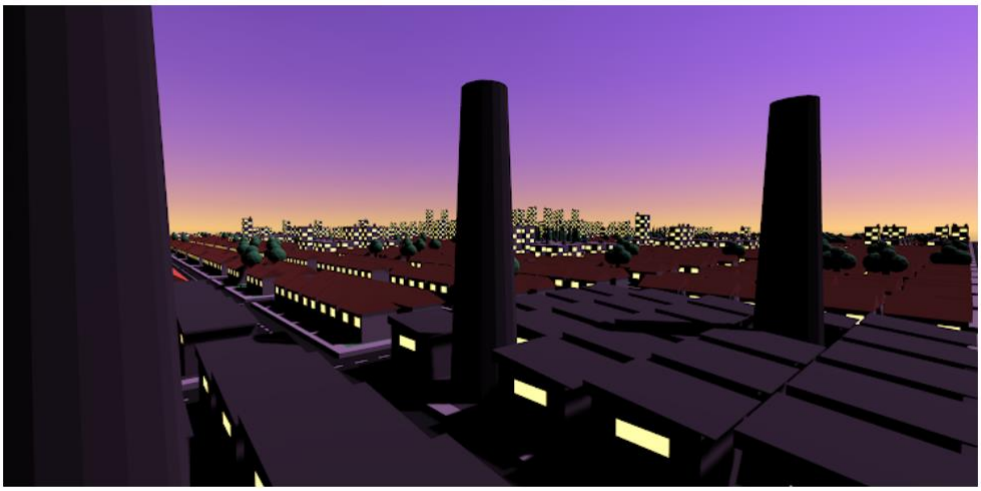
Please rate how visually appealing you find this image of a generated cityscape from 1 to 10:



	1	2	3	4	5	6	7	8	9	10	
Not appealing at all	○	○	○	○	○	○	○	○	○	○	Very appealing

Aesthetic Evaluation: Shadowy Industrial Zone *

Please rate how visually appealing you find this image of a generated cityscape from 1 to 10:



	1	2	3	4	5	6	7	8	9	10	
Not appealing at all	○	○	○	○	○	○	○	○	○	○	Very appealing

Aesthetic Evaluation: Conclusion

Based on the cityscapes you have just seen, what features of the cityscapes did you find most aesthetically appealing and which features do you think are in need of additional visual appeal (i.e. the composition of the city may have been to your liking, but you may have not enjoyed the shape of the buildings)?

Your answer

System Evaluation: City Generation

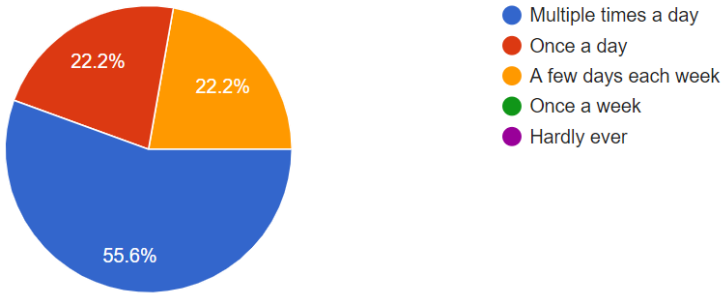
The user is able to choose the size of the city, along with the ratio of high rise buildings to residential houses. Green spaces are distributed evenly throughout, and an industrial zone is placed in one corner of the city. Would you like to see any additional features for generating a city (i.e. new kinds of buildings or more options to customise existing buildings)?

Your answer

F. Survey Results

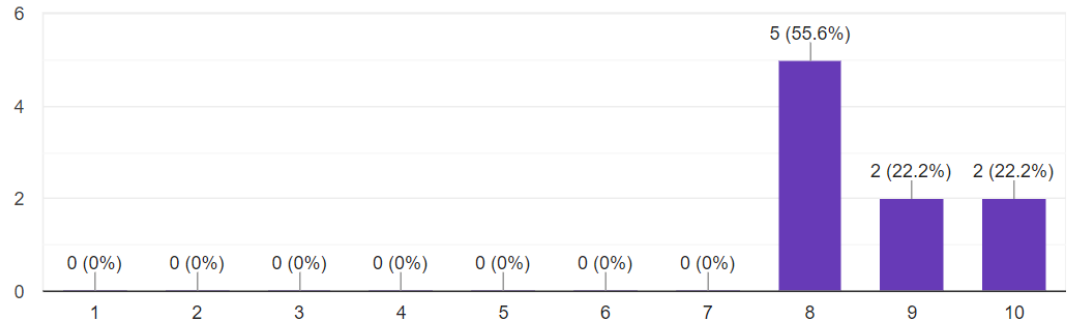
User Evaluation: How often do you view digital content (such as online videos)?

9 responses



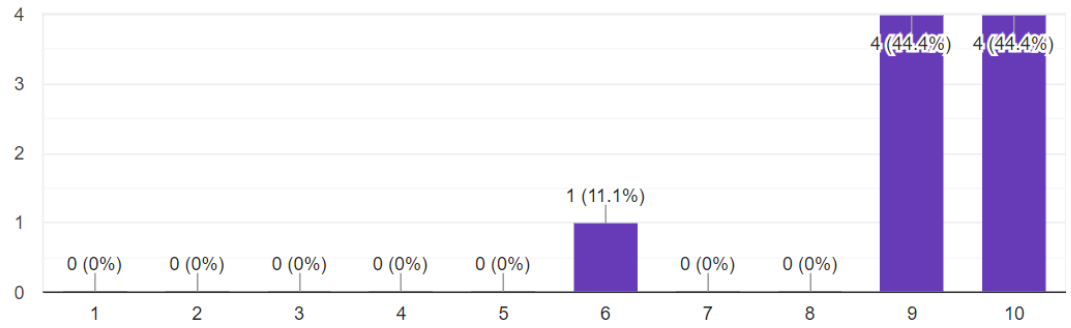
Aesthetic Evaluation: Bright Skyline

9 responses



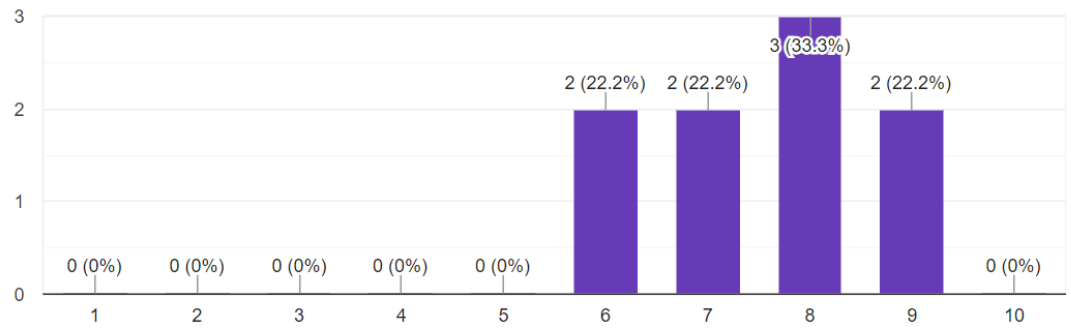
Aesthetic Evaluation: Dusk Lit Green Space

9 responses



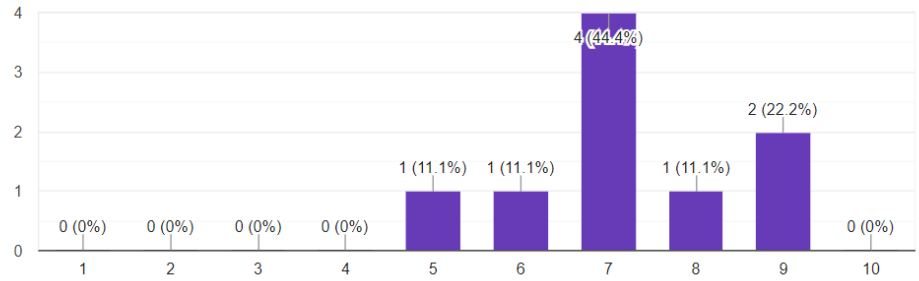
Aesthetic Evaluation: Purple Far Shot Skyline

9 responses



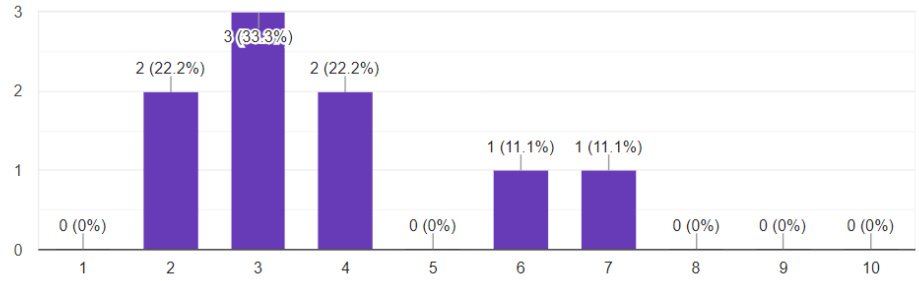
Aesthetic Evaluation: Red Extreme Shot Skyline

9 responses



Aesthetic Evaluation: Shadowy Industrial Zone

9 responses



Aesthetic Evaluation: Conclusion

4 responses

I like the long distance views of the big buildings in the city's centre! The shadows feel sorta gloomy though.

I really enjoy the colour of the sky but I think the shadows are a bit harsh.

I loved the lighting from the windows. The factory parts were a bit dull.

The light from the sun is pretty harsh, the park look neat though!

System Evaluation: City Generation

3 responses

It'd be nice to see some more kinds of skyscrapers, they all kind of look the same currently.

I'd like to see some different tall buildings

Some cars or trains or something populating the roads would look really good

G. Survey Consent Form

Sheffield Hallam University

CITYSCAPE GENERATOR SURVEY PARTICIPANT CONSENT FORM

TITLE OF RESEARCH STUDY: Cityscape Generator Evaluation Survey

Please answer the following questions by ticking the response that applies

	YES	NO
1. I have read the Information Sheet for this study and have had details of the study explained to me.	☐	☐
2. My questions about the study have been answered to my satisfaction and I understand that I may ask further questions at any point.	☐	☐
3. I understand that I am free to withdraw from the study within the time limits outlined in the Information Sheet, without giving a reason for my withdrawal or to decline to answer any particular questions in the study without any consequences to my future treatment by the researcher.	☐	☐
4. I agree to provide information to the researchers under the conditions of confidentiality set out in the Information Sheet.	☐	☐
5. I wish to participate in the study under the conditions set out in the Information Sheet.	☐	☐
6. I consent to the information collected for the purposes of this research study, once anonymised (so that I cannot be identified), to be used for any other research purposes.	☐	☐

Participant's Signature: _____ **Date:** _____

Participant's Name (Printed): _____

Contact details: _____

_____

Researcher's Name (Printed): Christopher John Noroozi

Researcher's Signature: CJNoroozi

Researcher's contact details: b8014084@my.shu.ac.uk
(Name, address, contact number of investigator)

Please keep your copy of the consent form and the information sheet together.

H. Survey Participant Information Form

The Cityscape Generator is a system that generates a city layout based on research done on real world cities; with a number of options for users to customize the appearance and size of the city. The city can be explored and pictures of the skyline taken to be used in various digital content.

To evaluate the success of the system's output you are being asked for your opinions on the cities the system generates. This form will ask you a series of questions about the aesthetic appeal of a few generated cities, and will also ask for some feedback on the options available to users generating and exploring these cities. Your participation will be immensely helpful for gauging the success of the system, and I will use them to determine the level of success.

There are a total of 6 multiple choice questions and 3 short written questions. The survey should take around 5 minutes.

You are under no obligation to take part should you wish to not answer the following questions and you may withdraw from the survey at any time with no reason required. A copy of the consent form you filled in is yours to keep should you require it.

Participation is done completely anonymously and no personal data is required to complete the survey. Responses to the survey - while anonymous - will be stored both within this Google Form and in a written academic report used to evaluate the system. Readers of the academic report will have access to the anonymous responses to this survey. The responses will not be passed to any other group for use in other studies, and the anonymous data will be held within the academic report indefinitely.

Should you wish to discuss your participation in this survey, or ask questions about it you may contact me at b8014084@my.shu.ac.uk at any time. Should you wish to find out the results of the study they will be accessible at the end of this survey.

The University undertakes research as part of its function for the community under its legal status. Data protection allows us to use personal data for research with appropriate safeguards in place under the legal basis of public tasks that are in the public interest. A full statement of your rights can be found at https://www.shu.ac.uk/about-this-website/privacy-policy/privacy-notices/privacy-notice-for-research. However, all University research is reviewed to ensure that participants are treated appropriately and their rights respected. This study was approved according to the guidelines of the Teaching Programme Research Ethics Committee of the College of Business, Technology and Engineering, Sheffield Hallam University (SHU). Further information can be found at https://www.shu.ac.uk/research/ethics-integrity-and-practice

You should contact the Data Protection Officer if:	**You should contact the Head of Research Ethics (Professor Ann Macaskill) if:**
• you have a query about how your data is used by the University • you would like to report a data security breach (e.g. if you think your personal data has been lost or disclosed inappropriately) • you would like to complain about how the University has used your personal data DPO@shu.ac.uk	• you have concerns with how the research was undertaken or how you were treated a.macaskill@shu.ac.uk

Postal address: Sheffield Hallam University, Howard Street, Sheffield S1 1WBT Telephone: 0114 225 5555