|  |
| --- |
| University Of The West Of Scotland |
| Design Document |
| Year 3 - Games Development Project |

|  |
| --- |
| Alasdair Hendry  9-26-2018 |

Contents

[1 Proposed Plan 3](#_Toc531040595)

[1.1 Development Team 3](#_Toc531040596)

[1.2 Game Outline 4](#_Toc531040597)

[1.2.1 Pitch 4](#_Toc531040598)

[1.2.2 Summary 4](#_Toc531040599)

[1.3 IDE 4](#_Toc531040600)

[1.4 Project Plan 4](#_Toc531040601)

[1.5 Risk Analysis 5](#_Toc531040602)

[2 Game Design Document 6](#_Toc531040603)

[2.1 Gameplay 6](#_Toc531040604)

[2.2 Influences 6](#_Toc531040605)

[2.3 Game Type & Research 6](#_Toc531040606)

[2.4 Target Audience 7](#_Toc531040607)

[2.5 Narrative 9](#_Toc531040608)

[2.6 Graphics & Design 9](#_Toc531040609)

[2.7 Level Design 9](#_Toc531040610)

[2.7.1 Level Layout 9](#_Toc531040611)

[2.7.2 Ambience 9](#_Toc531040612)

[2.8 Audio Design 10](#_Toc531040613)

[2.9 Moodboard 11](#_Toc531040614)

[3 Technical Design Document 12](#_Toc531040615)

[3.1 Development Approach 12](#_Toc531040616)

[3.2 Target Platform 12](#_Toc531040617)

[3.3 Hardware & Software 12](#_Toc531040618)

[3.4 Mechanics 13](#_Toc531040619)

[3.4.1 Summary 13](#_Toc531040620)

[3.4.2 Object Placement 13](#_Toc531040621)

[3.4.3 Citizen Actions 14](#_Toc531040622)

[3.4.4 Trade & Politics 14](#_Toc531040623)

[3.5 Development Environment 14](#_Toc531040624)

[3.6 User Interface 15](#_Toc531040625)

[4 Critical Appraisal 16](#_Toc531040626)

[4.1 Opinion 16](#_Toc531040627)

[4.2 Contribution 16](#_Toc531040628)

[4.3 Problems & Resolutions 16](#_Toc531040629)

[4.4 Future Implementations 16](#_Toc531040630)

[5 References 17](#_Toc531040631)

# Proposed Plan

## Development Team

The development team for this project will only comprise of myself. Due to this, I will take on all responsibilities to ensure delivery of the finalised product and will carry out the tasks for each role that a normal team would consist of.

In the interest of full transparency, a table has been provided which outlines the roles & responsibilities that are included throughout the development cycle of a game.

|  |  |  |
| --- | --- | --- |
| **Roles & Responsibilities** | | |
| **Role** | **Responsibility** | **Team Member** |
| Project Manager | Takes full responsibility for the project and ensures the final product is delivered to the client, whilst meeting the initial brief. Oversees the day-to-day tasks of each team member and provides support where available. | Alasdair Hendry |
| Lead Programmer | Stitches together each code snippet provided by other team members, whilst working on the more advanced mechanics of gameplay. As this is a one-man team, there are no other programmers. | Alasdair Hendry |
| Lead Designer | Ensures the overall quality of each stage of design, including Art, Audio, Level and Story. Is tasked with making sure each designer is taking the game in the correct direction with their contributions. | Alasdair Hendry |
| Art Designer | Creates 3D Models, textures, UV maps, sprites, textures and animations. | Alasdair Hendry |
| Audio Designer | Scores background music, sound effects and ensures all audio is designed to suitable level whilst also following the theme of the game. | Alasdair Hendry |
| Level Design | Creates the environment in which the game will take place. Ensures the placement of props & buildings are suitable to the genre of the game, and scaling is correct throughout each level. | Alasdair Hendry |
| Quality Assurance | Plays the game, whilst specifically looking for bugs. Suggests possible fixes for each bug and a step-by-step guide on how to reproduce the bug. | Alasdair Hendry |

## Game Outline

### Pitch

|  |  |
| --- | --- |
| **Title** | Township |
| **Platform** | Windows (PC) |
| **Genre** | City-Building / Economic Simulator / Real-Time Strategy |
| **Theme/Setting** | Set thousands of years ago, when settlers were laying the foundation for their new towns. |
| **Unique Selling Points** | Most games in this genre are restricted with either a grid or hex-like layout. This is not the case in Township, as the player will be able to build wherever they wish. |

### Summary

Township starts the player off with only a few peasants, some tools and a blank map which is their canvas. The player’s overall goal is simple; build and grow. To do this, they must satisfy the needs of their citizens by providing ample amenities and sufficient basic resources. On top of this, the player will be able to control the rate of growth by modifying laws & regulations and by creating trade routes through their towns.

## IDE

The games engine used throughout this project will be Unity Version 2018.1. This version is not the newest, however, it was recently released and offers a wealth of new resources compared to its predecessors. Notable features include;

* ShaderGraph – Offers a node-based environment in which the user can create PBR Shaders.
* Post-Processing Stack – Not a new feature, but noticeably updated to consider World Space Volumes.

Unity was selected as a development environment mainly because it is the engine which I have the most experience using. Also, the engine and all of its features are extremely well documented, allowing you to quickly check the Functions and Parameters that belong to a certain component. On top of this, Unity has tremendous community support, with a dedicated forum.

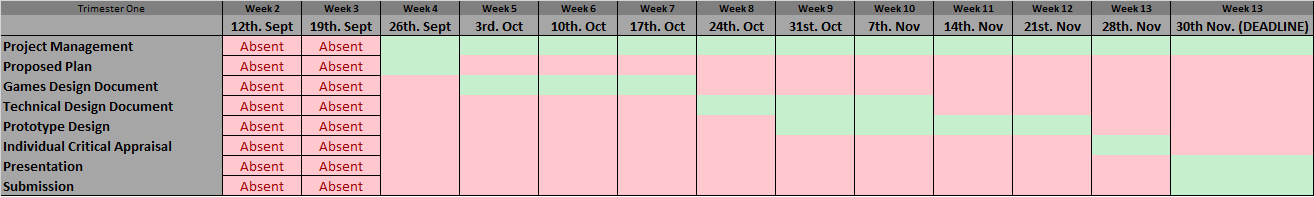
## Project Plan

To minimise the chance of any issues cropping up throughout this project, a project plan has been created, outlining the specific time-frames available to develop each section of the project. These time-frames provide a solid structure to reflect on when determining how much work is still to be completed.

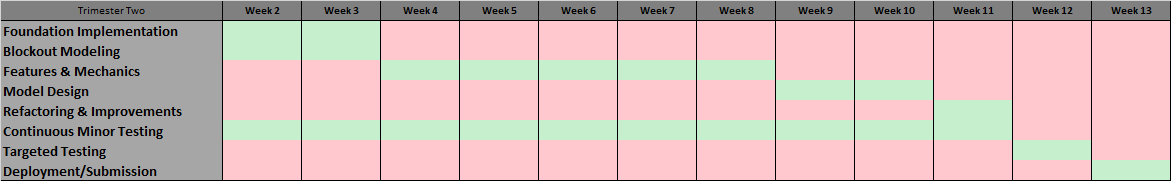
However, these time-frames are simply estimations that have been evaluated through reflection on previous projects, and as such, are subject to slight modification throughout the project if certain features are implemented quicker than first thought.

The following tables break the project down into a weekly schedule, in which each week has one or more tasks assigned to it.

**Trimester One – Gantt Chart**



**Trimester Two – Gantt Chart**



## Risk Analysis

As this project spans the length of two trimesters, around 24 weeks, analysing potential risk factors throughout the project is incredibly important. The risks include anything which may hinder progress on the development of the game or reduce the quality of the finished product.

Data loss and corruption is a major threat during any project and should be taken into consideration throughout. This means that data should always be handled correctly, and backups made at frequent intervals. Version control software is one of the most efficient methods of handling data backups, with services such as GitHub providing streamline cloud-based data storage. GitHub is also extremely useful for programming-based projects as it is able to compare merging conflicts ensuring work is not accidentally overwritten.

When multiple people are working on a project, measures must be taken to ensure that sickness or team members leaving do not affect the project. This can be done by creating backup plans or having multi-skilled team members. However, as this is a solo project, the solution is slightly different. If the only person working on the project becomes ill, then there is no-one else to take on any additional work. If this situation arises, then the best approach is to slightly modify the approach and re-evaluate the remaining work left to be completed, making modifications where necessary.

As a final consideration, understanding personal capabilities and project deadlines is also important as this ensures that the implementation of the game is not over-ambitious. This is an easy mistake to make as most ideas seem simple in theory, however, putting these implementations into practice is an entirely different matter. With each new feature implemented comes an exponential demand in work as these require further bug considerations, more testing, and more sprites, models or sound effects.

Fully understanding the risks involved in any project is extremely important, so to minimise any possible threats to the project. Having an extensive plan for the project is one of the most effective ways to reduce any potential threats.

# Game Design Document

## Gameplay

The player does not directly control any character within the game. Instead, they play as a governing body looking down on their citizens and interact with them by selecting them individually and providing commands and jobs/roles.

The camera in the game will be slightly angled towards the terrain, and the player will be able to pan, zoom and rotate around the global Y axis using the mouse and keyboard.

Interacting with the environment is performed by simply using the mouse. The outcome will differ depending on what the player is specifically interacting with, however, most interactions will be gaining information on specific parts of lands or instructing citizens to perform work in a certain area. On top of this, the player will be able to interact with their town through a series of GUI panels which will contain town-related information.

## Influences

Township takes influence from other successful games within the genre such as Anno 1404, Banished and Sid Meier’s: Civilisation series. These influences, however, only provide a basic foundation of features that will be built and expanded upon and affect the gameplay style more than the art style. Other games, out-with the target genre, such as Overwatch and Firewatch provide a source of artistic influence as the design leans more towards the stylized end of the spectrum.

## Game Type & Research

Township is a culmination of several genres. The most notable ones being city-building, economic simulation and real-time strategy. Whilst most games in these genres primarily focus on one specific genre, Township is truly a mix of all three. Players are able to build their city to their aesthetic design, ensure the economic growth of their city through trade and, finally, command individual citizens to perform specific tasks.

The majority of city-building games are restricted to either a hex-based or tile-based grid. This restricts the player as they must build their city around the game’s limitations. On top of this, most buildings must be connected to a road to allow citizens to access them. Township will allow the player to place a building anywhere they wish whilst still retaining functionality and efficiency.

Other games that fit in with the economic simulation genre usually handle trade in superficial ways. For example, Europa Universalis allows players to set up trade routes between countries. However, these trade routes are meaningless and only increase profit margins. Further to this, Sid Meier’s Civilization forces trade by creating trivial demand for resources the player does not have. This style of trade works for these games but is not the most realistic way of doing things. Township will work off of an import/export model, where the player is able to export excess goods that their citizens are not using and import goods that will provide comfort and happiness to their citizens.

## Target Audience

As Township is being developed for Windows, it is important to understand which audience this game will appeal to. This allows the game to be developed in a unique way which conforms to what this audience is used to, however, still providing different content to keep the user engaged.

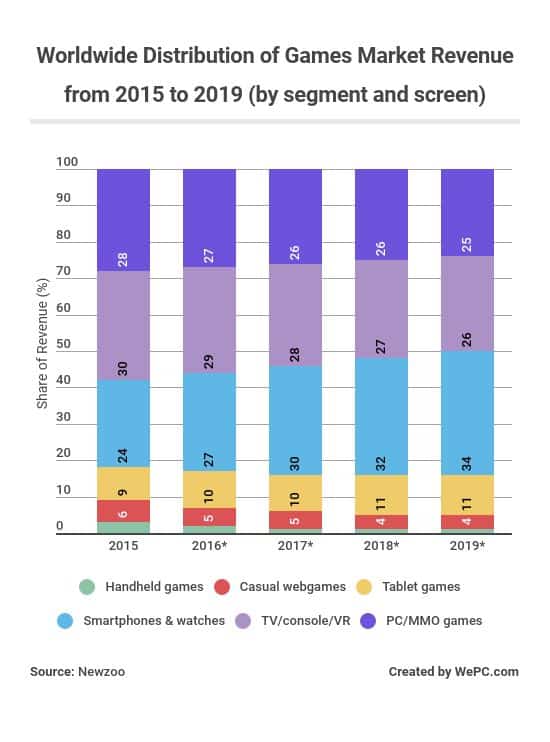
The PC market has been declining slightly in recent years, compared to other platforms. Taking up 28% of the global games market in 2015, falling to 26% in 2018. In contrast to this, the market value of the PC Gaming Industry has been steadily increasing over the past 7 years. Market value maxed out at $15,976,000 in 2011 and has almost doubled to $31,398,000 in 2018. It is important to include both of these statistics, as although the market isn’t competing with other platforms, sales are still rising and are predicted to rise through 2020 (WePC.com, 2018)[[1]](#endnote-1).

Figure 1 <https://www.wepc.com/news/video-game-statistics/#video-gaming-industry-overview>

Analysing the real-time strategy genre, it becomes clear that there are two clear motivations for playing such a game. These can be rated on a scale, defined as “Excitement” and “Strategy”. In a study of over 220,000 gamers, 20+ games where plotted on a map and rated on these scales. The study found that there is a “Cognitive Threshold”, beyond which the gameplay will no longer be fun, as the player tries to balance strategy with time-management. As can be seen in Figure 2 below, some games pass this cognitive threshold, and this can turn many players off the game as it overloads them with information (Quantic Foundry, 2018)[[2]](#endnote-2).

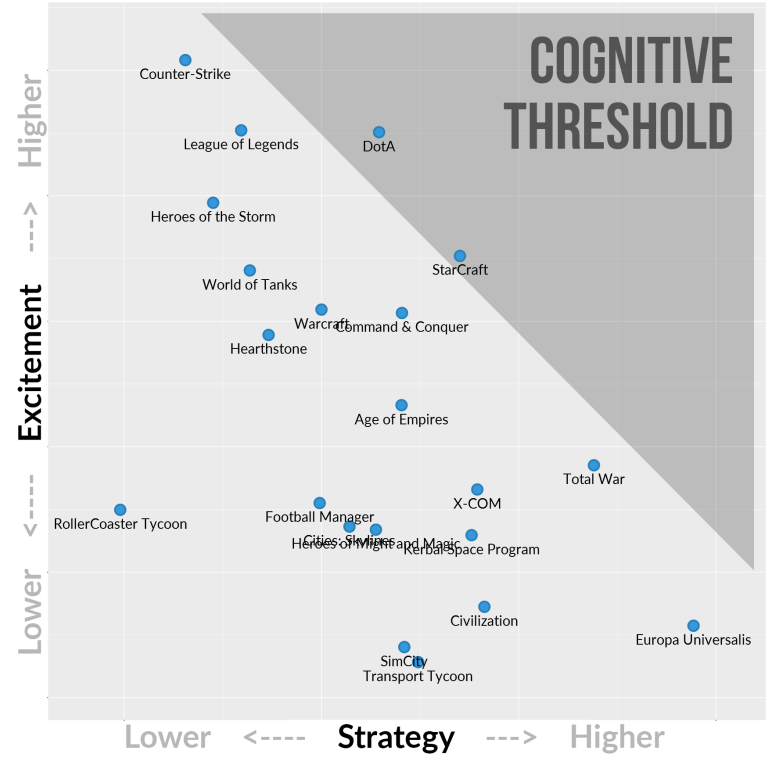


Figure 2 <https://quanticfoundry.com/2016/03/23/revisiting-the-strategy-genre-map/>

The same study of 220,000 gamers showed that the median age within this genre ranges from 21 to 31 years old. The younger audiences within this age-range tend to be drawn into fast-paced competitive games such as Counter-Strike and League of Legends. In contrast to this, “turn-based or pause-able games” such as SimCity or Civilization tend to have older audiences.

As Township will have a pause system and is designed to require an adequate amount of strategy to play, it is safe to assume that it would be plotted on the lower-right hand corner of Figure 2. On top of this, the age-range of the target audience will most likely fall between 26-31 years old. Having this information allows Township to be designed with a specific approach that will appeal to the target market.

## Narrative

As Township is not particularly story-driven, the player will most likely create their own narrative surrounding their town and its inhabitants as the progress through the game. As mentioned before, the gameplay follows the on-going development of the player’s civilisation as it grows from a petty settlement to a thriving metropolis, and the narrative is the actions that the player has taken to get it there.

## Graphics & Design

The graphical style that will be implemented in Township will be stylized and flat-shaded. The genre seems to be lacking any games with this art style. Due to this, there will be an extra layer of appeal to the game, as players will not have experienced this yet.

To achieve this graphical style, models will be created in Blender and uniquely unwrapped to allow different colours to be applied to each face of the mesh. Polygon count will also be taken into consideration when creating these meshes, trying, where possible, to reduce the number of vertices on each model. As many graphical items will be on the screen and many objects in the scene will be running many functions at any given moment, it is important to prioritize optimization in every aspect of the game.

## Level Design

### Level Layout

Map design will remain consistent throughout Township, using the same assets, theme and style regardless of the map location. However, the layout of each map will differ with each play-through. This is due to the fact that the map layout will depend on many variables, such as the difficulty that the player has selected, or if they opt to have a more mountainous terrain for that particular play-through. Allowing the player to have a slightly map dependant on their input will improve replay value and provides the player with more control over the experience they wish to have.

The player will be restricted to a play-zone that will be inset from the map border and they will not be able to move the camera outside of this zone. This allows for more terrain to be applied around the play-zone, which will look a lot more visually pleasing than simply having the terrain cut off at a certain point.

### Ambience

To create some ambience within the gameplay scenes, a day & night cycle will be added to the game to provide some extra diversity in the aesthetics department. Although this will not affect gameplay, the player will be able to toggle this mechanic off if they wish, as it may be distracting.

Ambient sound will also be used to add some extra life to the game. Sounds played will depend on the camera position relative to items such as fields, forests and marketplaces. On top of this, some sounds may change relative to the day & night cycle if the mechanic is enabled.

On top of this, in an attempt to add some extra detail to the game, there will be a focus on animating as many items in the world as possible. Something as simple as a waterwheel turning in the water can provide extra depth to a game. Lastly, a focus on small particle effects such as leaves falling from a tree will ensure that the player’s civilisation feels as alive as possible.

## Audio Design

Audio design is a seemingly underappreciated part of any game. Well planned audio design can greatly increase a player’s immersion in a game environment, and furthermore, can be used to great effect to provide audio cues to the player in the event that something has happened, or that the player must take a certain action.

To take full advantage of audio design within Township, sound effects will be logically stored within the game’s code in a global fashion. This will simplify the implementation of sound effects and provide a streamline system that will allow for uniformity of specific sound effects during certain events.

Sound effects can have a 2D or 3D spatial blend. 2D blend states will be used for interface events, such as a notification or a mouse-over event on a button. However, for more advanced effects that come from the environment, such as chatter from a crowded marketplace or a windmill spinning, a 3D spatial blend will be used. This allows the effect to fade in and out as the camera passes by the object.

Background music will use the same basic foundation as sound effects; however, it will be slightly different. Storing references to the music clips in a static class will allow multiple objects to modify which piece of audio should be played during specific events. On top of this, background audio will be channelled through multiple controllers which will allow the use of specific effects, such as adding reverb and slowing the music substantially when the player pauses the game.

## MoodboardC:\Users\b00330023\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Moodboard_Banished.jpgC:\Users\b00330023\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Moodboard_Firewatch_02.jpgC:\Users\b00330023\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Moodboard_Civilization.jpgC:\Users\b00330023\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Moodboard_Anno.jpg



# Technical Design Document

## Development Approach

Many development methodologies are available to implement for a game development project. These methodologies are simply guidelines which you are advised to follow, as they can greatly increase development efficiency, allow you to detect bugs earlier and quickly iterate through development phases using a structured approach.

With no doubt, the Agile methodology is the most popular for games development projects, and as such, will be the approach taken throughout this project. Agile is an iterative approach which encourages layered implementation and a test-as-you-go strategy. This is an extremely useful approach for short-life projects as at the end of any given iteration, the product could potentially be built and released, providing freedom to remove features if time constraints become an issue.

Although this project spans 24 weeks, double the average university trimester length, it is still a relatively small timescale to plan, develop, test and deploy a fully featured 3D game. This is the main reason the Agile approach has been chosen for this project.

In contrast to the Agile approach, there are others available, such as Waterfall and V-Model. These approaches are preferable for projects which have access to a greater time frame, for example in a professional game studio.

Waterfall suggests a lineal approach to development, such that planning is complete before design, and design is complete before implementation. This is a solid foundation to follow during lengthy projects, as it almost guarantees a flawless development cycle. The downside to this is that it takes an enormous amount of time and effort to fully carry out each stage before continuing to the next and restricts the ability to add new features after design stages are complete.

V-Model suggests a similar approach to Waterfall, as it uses the same base foundation, but adds iterative testing stages. This means each stage is tested against every previous stage before continuing. This requires even more time than traditional Waterfall, and as such, is unsuitable for such a small project.

## Target Platform

The final build of Township will be for Windows (PC). The reason for this is directly correlated to the genre of the game as hardware limitations may hinder the experience on other lightweight devices due to the intense processing and graphical demand for a game of this scope.

## Hardware & Software

Creating a game from scratch requires many different pieces of hardware & software. This can be difficult, as it is important that each piece of software is updated to the appropriate version throughout the entire project, for all team members. If version control is being used to develop between multiple devices, then even more difficulty is created as software versions must persist through each device. All the hardware & software used throughout this project is listed below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Software** | | | | |
| **Name** | **Type** | | **Version** | **Use** |
| Unity | Games Engine | | 2018.2.2f1 | Provides useful libraries and API’s to increase production and efficiency during development. Also provides interactive interfaces for ease-of-use. |
| Visual Studio | I.D.E | | Comm. 2017 | Allows text editing for programming-related files such as .js or .cs, whilst providing text prediction and error logging based on the current language and engine. |
| Adobe Illustrator | Vector Images | | CS6 | Used to quickly create and modify vector graphics, allowing them to be scaled to any size and exported. |
| Adobe Photoshop | Pixel Images | | CS6 | Used to modify an image to suit a specific style or quickly create an alpha mask. |
| Audacity | Sound Editing | | 2.3.0 | Used to edit audio files, providing useful effects and modification tools. |
| Blender | Modelling | | 2.79b | Used to create 3D models and apply UV maps to existing objects. Also used to render promotional scenes. |
| Microsoft Office | Text Editor | | 365 | Used to edit text documents. |
| Github Desktop | Version Control | | 1.4.1 | Used to backup project versions and provides the ability to create branches of an existing project. |
| **Hardware** | | | | |
| **Name** | | **Use** | | |
| Keyboard | | Used to provide input to the engine and game. | | |
| Mouse | | Used to provide input to the engine and game. | | |
| Mid-Level Computer | | Runs the required software. | | |
| Monitor | | Displays the contents of the Computer. | | |

## Mechanics

### Summary

Township features multiple individual mechanics that work together to create the functionality needed for the game to run as intended. These features are created as modularly as possible, which creates an easy-to-use system for adding and removing different components to the game. Features include object placement, object inspection, citizen actions, town interaction, technological progression, trade and political decision making.

### Object Placement

With Township being predominantly in the city-building and simulation genres; object placement plays a massive role in the flow of the game. Players are able to view and filter all available buildings that can be placed in the world. Once a building has been selected, the player can place a blueprint of their selection where they please and provide a rotation for it. Once a building is placed, a job is queued in the logic of the game, and an AI citizen will begin to construct the building.

### Citizen Actions

Citizens are the centre-piece of Township. Without citizens, nothing of substance will happen. Due to this, it is crucial to program the intelligence of these characters perfectly. The AI in Township is responsive to actions taken by the player. Specific actions that the player takes can create multiple jobs for the citizens to do, which are then sorted by priority and assigned to citizens who are employed in the relevant job field.

An example of this is building placement. When the player places a new building foundation in the world, multiple jobs are created and stored in the available job queue. A job is created for each resource that is required to construct the building, and a single job is created for the construction of the building itself. Citizens that are employed as a workman then request these jobs and, once confirmed, will proceed to locate the required resources from warehouses around the world, and carry them to the construction location. After all required resources are available at the construction site, the workmen will then construct the building.

### Trade & Politics

Actively taking part in trading and politics within Township is a great way to progress the player’s civilization.

The player is able to send trade ships to different civilizations within the game to buy specific resources or to sell surplus stock. In doing this, the player’s civilization has the best chance of advancing by consuming luxury, exotic resources. On top of this, selling surplus resources provides the player will extra gold that allows them to construct bigger and better infrastructure in their town.

Furthermore, engaging in politics can grant the player extra control over their citizens. This can work in numerous ways, as the player can generate more wealth by demanding extra taxes from citizens, at the risk of making them unhappy. Restricting resource consumption allows the player to stock up on supplies, however, this can cause citizens to become malnourished. In contrast to these decisions, the player can also choose to pass laws that have a positive effect on their citizens, such as promoting events or providing free services.

## Development Environment

Unity is a development environment, available on Windows (PC), and MacOS. Used to develop many successful games, such as Hearthstone, Rust, Kerbal Space Program and Cuphead, it has become a must-have tool for any independent programmer (Dotan, 2015)[[3]](#endnote-3).

Built using a component-based foundation, Unity makes it easy for a programmer of any skill level to have a prototype up and running in a matter of days; possibly even hours. On top of this, the community for Unity is thriving with constant blog posts, documentation updates, YouTube tutorials and new versions being released every month.

Unity provide a wealth of API’s native to their engine, some of which include VR support, AR support, multiplayer, advert integration, team-based version control and cloud building. On top of this, Unity allows users to build their game to every major gaming platform, including Windows (PC), MacOS, Android, iOS, WebGL and Xbox One.

In contrast, Unity does have some major drawbacks. The most notable ones, when compared to engines such as Unreal Engine, are the lighting and shading systems. However, these only seem to be an issue out-of-the-box, and with a little knowledge a graphics programmer would be able to enhance the visuals of any Unity-made game.

Other flaws include the lack of stability in new versions, or versions in preview. The issue with this is that the most recent versions are so much better than previous releases that there is incredible incentive to update. Due to this, users may find themselves in difficult situations mid-development.

Comparing Unity to other engines such as Unreal Engine and GameMaker, Unity shines in most areas. Other engines don’t seem to offer the same features and support as Unity, and the documentation for them doesn’t seem to compare. Furthermore, Unity is very open-ended and is well-suited for many different project types, in contrast to other engines such as Unreal’s node-based FPS focus, or GameMaker’s strong focus on 2D.

Overall, Unity is the most easily-accessibly, lightweight and durable engine for any programmer. This combined with the incredible community support and previous experience are the main reasons that Unity will be used as a development environment for this project.

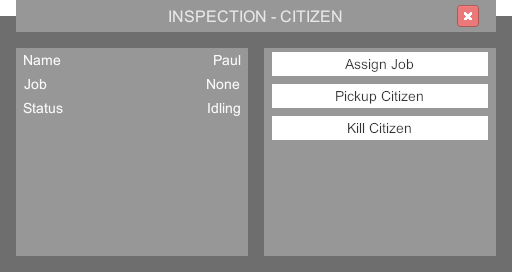
## User Interface

User Interface is the backbone of any well-made game. Without it, the user experience would be greatly diminished, as it serves as the middle-man between the game’s logic and the user’s perception of what is currently happening.

The need for well-designed UI is greatly increased in the simulation genre. This is due to the nature of the game, in that there is so much going on at the one time, that the user needs to be informed elegantly and categorically.

Interfaces in Township are designed to be as modular as possible, whilst still sticking to the same fundamental, convenient layout. On top of this, the Interface System is designed around Delegate and Call-back patterns, which allows for extensive functionality. For example, entering “Build Mode” opens the Build Panel, whilst also checking if any other conflicting modes or panels are currently active, and shuts them down.

On top of this, the Interface System is easily extensible by other objects due to the nature in which it was created. This allows different objects to display different information on the same panel. For example, when the user clicks on an object in the world to inspect it, that object will provide specific information. A house may want to display its occupants and average income, whereas a warehouse may want to display its current stock. This is easily achieved by the Interface System, with minimal setup required by the object being inspected. Below, two screenshots of the Interface System from the prototype demo are shown.



# Critical Appraisal

## Opinion

Relative to the Project Plan, I feel that the initial stage of this project has progressed rather smoothly. With each section of the project being completed on or around the given date in the plan, it’s safe to say that the plan was adhered to throughout the project.

On top of this, I feel that there has been a great amount of time and effort put into this project by myself. Considering the project team should have been 3 – 4 people in size, I am tremendously proud of what I have been able to achieve on my own thus far.

I am rather excited to carry this project on through to it’s second stage and start working on implementing the full mechanics of the game.

## Contribution

As previously mentioned, this project is being carried out as a solo venture. Therefore, every piece of contribution made to the project was made by myself.

I understand that as there are many other projects being worked on at the same time, it is impossible to give 100% of my time to a single project. However, using my best judgement, I delegated my time and resources appropriately to ensure each project received a sufficient amount of time and attention. Due to this, I am pleased with the amount of resources I assigned to this project.

## Problems & Resolutions

When analysing risks to the project during the planning stage, I highlighted the possibility of a team member becoming ill as being a significant risk to the project. As outlined, this risk is amplified with a smaller team.

Unfortunately, I became extremely ill during week 11 and most of the production for that week came to a halt. To overcome this, I made some minor adjustments to how I would approach the final few weeks of the project to ensure the project was delivered on time.

The drawback of this is that I had less time to work on the prototype and couldn’t add as many features. Due to this, some basic features that I wanted to add were scrapped from the prototype. These features included saving and loading, more buildings such as a marketplace and a tavern, a basic tutorial, and finally, some animations to make the game feel a bit more alive. Regardless of this, I feel that the prototype is still an accurate presentation of what I envision the final product to be.

## Future Implementations

As the prototype of the game has been completed, it now serves as a basic foundation to build upon throughout the next stage of the project; the implementation stage. When this stage begins, there will be a major focus on extending the existing features to add new functionality and implementing entirely new features to add more depth to the game.

A list of features that are planned to be added in the next stage of the project are outlined below.

* Extend the AI system to include specific jobs, such as a woodcutter or miner.
* Extend the AI system to allow multiple citizens to work on a single job.
* Create more objects that can be placed, such as a woodcutter’s hut, a mine and many more.
* Create a Saving & Loading system.
* Further analyse Unity’s navigation system to determine if baking can be performed asynchronously.
* Update movement logic to stop citizens getting stuck on each other when both have the same destination assigned.
* Create a simple tutorial
* Create a task/achievement system.
* Create Law system.
* Create Trading system.
* Adding amphibious vehicles to use the trading system.
* Extend Interface System to display graphs.

# References

1. WePC.com. (2018). *2018 Video Game Industry Statistics, Trends & Data - The Ultimate List*. [online] Available at: https://www.wepc.com/news/video-game-statistics/#pc-gaming [Accessed 10 Oct. 2018]. [↑](#endnote-ref-1)
2. Quantic Foundry. (2018). *Game Genre Map: The Cognitive Threshold in Strategy Games - Quantic Foundry*. [online] Available at: https://quanticfoundry.com/2016/01/20/game-genre-map-the-cognitive-threshold-in-strategy-games/ [Accessed 10 Oct. 2018]. [↑](#endnote-ref-2)
3. Dotan, G. (2015). Top 10 Unity Games Ever Made. [Blog] SoomlaBlog. Available at: http://blog.soomla.com/2015/01/top-10-unity-games-ever-made.html [Accessed 3 Oct. 2018]. [↑](#endnote-ref-3)