Project Overview

Outline.

This game project will be a barebones colony sim/management title, it will be a 2D top-down game. This will involve the player spawning with a randomly generated map and a small selection of colonists; their goal will be to keep the colonists alive. The player will be able to build structures on the map by assigning build tasks for the colonists but each task will require a number of in-game resources which they will have to collect from around the map, for example, if the player wants to build a wooden house then they will need to cut down a tree to get the raw wood. I aim to go into depth with the map generation similar to titles like dwarf fortress I aim to create a multi-level map (a map with multiple layers on top of each other). Starting with the ground level (Level 0), I would like to have layers ranging from (+3 -> -3) this will allow for the player to have more options within the game.

Aims.

I aim to work on a game with randomly generated elements, for example, the map, it will consist of different tiles overlayed on a grid. Each grid square will be a specific terrain object, e.g. a tile of dirt and so on.

I also aim to test out a number of different pathfinding algorithms using the map's grid system attempting to determine which pathfinding algorithm is most effective. Comparing the shortest path and shortest time; with the careful consideration about optimisation.

Implementation Strategy.

I plan to implement this game using C++ and SFML. I am familiar with C++ and have developed a few items using this combination and am more than confident in my abilities to program with this setup. SFML is a library which allows for the easy implementation of 2D games with useful features ranging from window management to sound implementation both useful items to speed up development.

I plan on using the agile methodology for the game implementation; wherein each week will be a sprint so, on Monday I will plan and design my tasks, Tuesday to Friday will be

dedicated to the building of the game and Saturday and Sunday will be for testing and reviewing the implementation as well as, fixing any issues which have arose.

Main Gameplay Mechanics.

- Random Map - At the beginning of the game a random map will be generated for the player.
- Colonists - These (also randomly generated to a degree) will allow for the player to manipulate the map.
- Harvest Resources - Resources will be spawned on the map and the player can use their colonists to collect these.
- Base Building - The player will be able to create custom structures and fill them with furniture to keep the colonists safe and happy.
- Random Events - There will be some random events which may occur during gameplay, mixing things up a little bit and keeping the game interesting.

Similar Titles

Dwarf Fortress - http://www.bay12games.com/dwarves/

Rimworld - https://rimworldgame.com/

Prison Architect - https://www.introversion.co.uk/prisonarchitect/

Project Plan

Development strategy

Using the **agile approach with the development of the game,** the game will be stripped down into **many different sprints** wherein **each week will be a single sprint** dedicated to the implementation of a game feature or function. At the beginning of each week, I will take a look at what is needed for the game, conduct **a phase of planning**. After **the bulk of the week will be for the implementation** of the component. Finally, the **final days of the week will be used for testing** and reviewing with **added time for any fixes**/changes that need to be made.

Milestones

This project will have **five main milestone tasks** which need to be completed; each one is important for me to achieve the objectives set for the project. **The first, a map which will be randomly generated** at the beginning of the game. **The second, implement a simple colonist** for the game, one with working AI pathfinding. **The third, implement a selection of resources** into the game and allow for the colonist to perform a basic set of tasks. **The fourth, A building system** for the game, which interacts with the multiple layers of the map. And **finally, a random events system** with a couple of different events with different outcomes.

Gantt Chart:

https://docs.google.com/spreadsheets/d/1IvRP2XnJYIO8LR09rxNJRsqJsWtvVfdTz-adm9MfdnA/edit?usp=sharing.

Ryan Hood | Student Email: p1617694x@my365.dmu.ac.uk
Email: Hoodrn03@gmail.com | Github: https://github.com/Hoodrn03

Game Design Document.

Colony Sim/Management Game.

By Ryan Hood.

Contents

Introduction: Page 2.

Story: Page 2.

Characters: Page 3 - 4.

Level and Environment Design: Page 4.

Gameplay: Page 4 - 8.

Art: Page 8.

Sound and Music: Page 8.

User Interface and Game Controls: Page 8 - 10.

Accessibility: Page 10.

Conclusion: Page 10 - 11.

Appendices: Page 11 - 12.

Ryan Hood | Student Email: p1617694x@my365.dmu.ac.uk
Email: Hoodrn03@gmail.com | Github: https://github.com/Hoodrn03

Introduction

This document will provide a detailed explanation of what will come together to create this game. Using a mixture of diagrams and textual explanation I will discuss the game concept and how it will be implemented.

The base game idea is a colony sim, or in broader terms a management sim; wherein the player will **start on a randomly generated map** (with **randomly placed resources**) and they must build various items and buildings within the map **to help their colonists survive**. Their colonists will have a **variety of wants and needs** which must be fulfilled to keep the game moving.

The aim for this game project will be to **tackle random generation of the maps**, with the added difficulty of **having multiple different levels** (layers) in the maps; it will be important to keep the maps consistent, for example not having randomly floating rocks in the upper layers with no supports below it. The game will also have the objective of experimenting with **pathfinding algorithms** determining which will be the best for this game, investigating both the **shortest path and shortest time algorithms**.

For the implementation of the game I will be **using C++ within visual studio**, but more importantly, **I will be using SFML**: a library which allows for **easy image processing** for simple graphical elements, **networking and sound** (among other things). This allows for both **ease of use as well as flexibility**, which is not always available when using other engines (Unreal Engine for example) and will help to **improve the speed of game development**.

Story

This game will not have a story per se instead it will have **a random event system**. This means that while the player is doing their own thing **at random intervals an event might affect the overall gameplay** for a short while, for example; the player is creating an expansion for one of their buildings when a raid event triggers, which places randomly generated hostile entities at one of the map edges and they will move towards the player's colony with the goals of killing and injuring the player's colonists. The raid event will be considered one of the bad events that could happen to the player, but **there will be a mixture of both good and bad events** that could be **randomly selected**.

Ryan Hood | Student Email: p1617694x@my365.dmu.ac.uk
Email: Hoodrn03@gmail.com | Github: https://github.com/Hoodrn03

The aim for this system will be to **allow for the player to create their own stories** for this game, this will not only **increase the replayability** of the game but also give the player a certain level of connection to the game.

Characters

The characters for this game will **vary between each playthrough**. The only characters which will be within the game are **the colonists which will be randomly generated** upon the games beginning. The colonists are one of if not the main concepts to the game, **the player must keep them alive and happy** in order for their colony to thrive.

When the game begins and the colonists are generated **they will start off with a few different systems**, to start with they will be given **a random name** (forename and surname) this will help with the personifying of the colonists. Each of these characters will also be **created with a few stats** (characteristics which determine how effective they work at certain tasks) consisting of:

- **Strength** - The character's raw physical power.
- **Perception** - The character's awareness of their surroundings.
- **Endurance** - The character's general resistance.
- **Charisma** - The character's social awareness.
- **Dexterity** - The character's ability to manipulate their surroundings.
- **Intelligence** - The character's intellect the ability to use and create complex systems.

The aim for the characters is to give the player something to connect to, and because the colonists are the main part of the gameplay, it is important that they are complex and interesting enough to keep the player attention. Therefore the colonists will have a certain **level of editing available** allowing for the player before the game begins to make any changes they want for the colonists, like **changing their name**, or **adjusting their stats**.

Each character will also have a few **status conditions** that **the player must manage**. The values are (But not limited to):

- **Health** - The life of the colonist, if it reaches zero the colonist will die.
- **Thirst** - The need for the character to drink.
- **Hunger** - The need for the character to eat.
- **Sleep** - The general tiredness of the character.

Ryan Hood | Student Email: p1617694x@my365.dmu.ac.uk
Email: Hoodrn03@gmail.com | Github: https://github.com/Hoodrn03

- **Stress** - The mental pressure the colonist is under.

The player must **build objects and buildings** within the game world to help them manage all of these values e.g. the player might be able to build a water pump to provide the colony with water, ext. This component to the characters will be what **pushes all of the other gameplay elements**; mainly the **base building** and the **random events**.

Level and Environment Design

At the beginning of the game a **map will be randomly constructed upon a simple grid** (probably around 200 x 200 to begin with), where each **cell on the grid will act as a tile in the map**. Each tile can spawn as one of a small selection of types ranging from stone to dirt (these tiles will determine what the cell will look like).

The map will have **a couple different levels** (or layers) which will **go both above and below the ground level**. This will allow for the player to **have a little more control** over the colony, these extra layers can also be built upon.

During the generation process, **the ground level will be the first to be generated** allowing for the upper and lower levels to be **generated with the same topography**. Each map is guaranteed to have at least **a source of water** (being either a river or a lake) **some dirt** (the main floor tile) and **some rock** formations (forming the main bulk of the lower levels and potentially mountains).

The main idea of a randomly generated map is to **add variation to the gameplay**, preventing having the same map twice. This will **help the replayability** of the game allowing for the player to **play multiple playthroughs** on many **different maps** with **different layouts** of buildings, mountains, lakes and rivers.

See **Appendix A** for more info on the basic map generation.

On each of the maps, there will also be a scattering of **resources on the maps**, which will be **building materials**, the main resource which will be spawned will be wood… During the map generation, on each dirt tile, there will be a chance that the tile will have a tree on it. (See the Gameplay section to learn more about the resources). Suffice to say during the map generation there will be resources added into the map, ranging from plants (e.g. trees) to resources inside rock (e.g. iron or gold).

Ryan Hood | Student Email: p1617694x@my365.dmu.ac.uk
Email: Hoodrn03@gmail.com | Github: https://github.com/Hoodrn03

Gameplay

This section will cover all of the gameplay elements which will create the game in great detail, explaining **how they work** and the ways they may **interact with other systems** within the game. This section will be split into **two important topics**; the **required elements**, the ones which will need to be implemented to form the game, and a **wish list** which will be completed at the end of the deadline permits it.

Required Gameplay Elements

Base building: this is one of the key gameplay elements for this type of game. This will involve the player being able to create buildings within the game world using resources from said world. Using the tile/grid system the map is generated with also allows for the easy placement of 'blocks' in the world; for example, if the player wishes to create a wooden wall in the map they will place the block onto a tile, thus forming a single wooden wall tile. This system will work for all of the building blocks, if the player wishes to build a wooden floor then they place the tile down where they want and the tile becomes that allowing for complex buildings and structures to be created easily and quickly. However, the furniture which can be placed will work slightly differently, it will be overlayed on top of the tile, while the placement will still work in a similar way (tile locked - preventing the player from placing an object outside or in the middle of a tile), the tile will still be shown below the object. E.g. if a chair would be placed on some dirt then the dirt would be shown below the chair, but both of them would still be displayed, additionally, furniture will not be placeable within walls.

Random events: The story element for this game will be delivered through a random event system, this will allow for the players to have unique experience between each playthrough, it will also differ between each player. This will allow for the player to have a special connection to this game, giving them their own story to tell; for example (https://ludeon.com/forums/index.php?topic=41809.0), this is just one place and there are many more. In terms of implementation I aim to have a list of possible events and at random intervals one of them will be selected, causing either something good for the player or something bad. It's all down to a random number generator. For more info on the potential events see the Random Events spreadsheet (https://docs.google.com/spreadsheets/d/1R_aV6u4QltcNEDkX95QkIbhnMbnDyVBAm5HdpnOkSio/edit?usp=sharing).

Ryan Hood | Student Email: p1617694x@my365.dmu.ac.uk
Email: Hoodrn03@gmail.com | Github: https://github.com/Hoodrn03

Colonists: The colonists will have a few different elements associated with them, the first of them being their skills. The skills (as mentioned above) will be the key to how well they will perform certain tasks. The second component to them will be the tasks that they perform;

- Building. After a player has designated what they want to be to be built, if the colony has the required resources then the colonist will bring those resources over to the object and begin to construct it. Each item will take a different amount of time to build added on the character will have a stat modifier that adjusts the building time.
- Medical. After one of the colonists has been injured then it will be up to one of the other colonists to tend to their wounds.
- Transporting. After a resource has been harvested from somewhere on the map, a character tasked with transporting will pick up the item and bring it to a zone marked on the map. Allowing for all of the player's resources to be stored in one place.
- Mining. This will allow for the player to assign a piece of rock somewhere in the world map and the colonist assigned to this task will break down the rock, collecting the resource and freeing up the tile which was rock previously.
- Harvesting. This is the task of collecting a resource from a plant spawned on the map. This plant could be, for example, a tree, and if the player wants some wood they would assign a harvesting order on a tree and the assigned colonist will cut the tree and collect its wood.
- Hunting (Wish List). This task represents the task of finding and killing an animal on the map and collecting the resources from it (meat and hide). The character will use their equipped weapon on the hunt.
- Taming (Wish List). This task will require one of the colonists to attempt to tame an animal using some food from the colony. If the taming attempt is successful then the animal will become a member of the colony (possibly moving and remaining in a designated zone), from where the player can then harvest different resources from different animals, for example, a chicken will lay eggs for the colony, but they can also be killed at any point for the meat and leather.
- Crafting (Wish List). The player will be able to assign a crafting task (possibly at a specific workbench) and the assigned colonist will then take the required resources and construct the item. The item could possibly be a weapon…
- Cooking (Wish List). This will allow for the assigned colonist to process raw food like meat and create a more edible food for the colony.

Ryan Hood | Student Email: p1617694x@my365.dmu.ac.uk
Email: Hoodrn03@gmail.com | Github: https://github.com/Hoodrn03

A big component for the colonist will be the way that they move around the colony, therefore a certain level of consideration for their movement must be taken. Therefore during implementation, I will be testing and comparing the way in which a path finding algorithm finds the path. After some research into pathfinding algorithms (https://en.wikipedia.org/wiki/Pathfinding) I have decided that A* will most likely be the best for this type of game, with an added consideration for the heuristics that will be used testing both the shortest path (the smallest number of notes from the start to the end) and shortest time (the shortest estimated time from start to end).

Additional Gameplay Elements (Wish List)

Ranged Combat: This will be the initial form of combat within the game, where two or more colonists will fire their weapons towards each other with the intent to injure. Each weapon will have three main stats as well as a classification. The first being its power, the damage to the colonist if hit by the projectile. The second The weapon's aiming time/fire rate, in other words how long it will take to fire the next projectile. The third, the weapon's weight, so if the weapon is heavy it will reduce the colonist's movement speed so the player may have to balance a weapons damage over its weight. And finally the classification, there will be three separate classifications;

- Close Range - Covers weapons which require the character to get far closer to the opponent to be able to deal damage.
- Long Range - Covers weapons which are able to fire from long distances allowing for reach.
- Automatic - Covers weapons of both long and short range with the added bonus of firing a volley of around three projectiles each trigger.

Melee Combat: This will work in a similar way to ranged weapons, they will have the same stats, but have different classifications. The main difference between ranged and melee is that for this form of combat, the colonist must be in an adjacent tile to their enemy. The different classifications will affect the type of damage they do, they are;

- Slashing - A blade that will cut the opponent.
- Bashing - A blunt object that will bludgeon the opponent.
- Stabbing - A sharp object that will pierce the opponent.

Animals: An additional feature that I would like to include is animals, initially starting off with small animals, such as birds but later to include larger animals. Animals like all of the other resources will spawn randomly on the map and will have two main collectable

Ryan Hood | Student Email: p1617694x@my365.dmu.ac.uk
Email: Hoodrn03@gmail.com | Github: https://github.com/Hoodrn03

resources; their meat and hide. These will be used to provide an easy source of food for the colony, but they will be tamable to allow for a slow income of additional resources.

Zoning: This will allow for the player to choose a section of tiles in the grid and assign what type of resources will be stored there. This will allow for the player to easily keep track of their resources. I aim to add additional zones into the game, allowing for better managing of colonists and colony resources. For example, an animal zone which will prevent the animals assigned to it to leave the zone, and possibly another zone which limits colonists from leaving a certain zone.

Crafting Items: I also aim to add in a basic crafting system into the game wherein the player will create an object in the map and it will give them access to recipes to create, this could be a mixture of food, weapons and various other things. The crafting will include the assignment of the item, the colonist collecting the resources required to build that item and the colonist spending time constructing the item.

Art

For this game, I aim to use a rather **simple art style**. I will use **basic blocky objects** for the furniture; using room plans for inspiration. These room plans have made **use of blocks and circles to effectively convey the information** on the page. For example, a bed is a rectangle with a rounded rectangle towards the head of the bed.

I aim to implement tiles for each of the grid cells, this would also mean that I require some **textured art for each of the tiles**, at the moment I aim to implement **three simple tiles** into the game: **water**, **dirt** and **rock**. I plan to create three similar tiles for each of the sections **allowing for slight variation**.

For the colonists I plan to **start off with white boxes** for **testing purposes**, but if time permits I would like to implement **more detailed pixel art** using **games like prison architect and rimworld** as a baseline for the game.

Sound and Music

The music in games is **extremely important**, it is used to **convey a certain feeling** while in the game. The main feeling that I want to convey whilst the player is in the game, is **a sense of calm**. The reasoning for this is that I aim for this game to be **a relaxing experience**, yes there may be some situations which might become a little stressful but this **game should ultimately be calm**.

Ryan Hood | Student Email: p1617694x@my365.dmu.ac.uk
Email: Hoodrn03@gmail.com | Github: https://github.com/Hoodrn03

User Interface and Game Controls

When it comes to UI it is important to make it both **easy to understand** and use. Therefore I plan to use **a rather simple overlay** for my UI. It will contain **large text** as well as **an image** to convey the action the button will be used for; for example, a button to assign a colonist to cut a tree will have an image of an axe on it to signify the act of cutting a tree.

I aim to include **three different sizes for the UI**, a **small**, **medium** and **large**; this will allow for the user to **adjust the user interface** to better fit with both their needs and their setup. This will require a lot of configuration to ensure all of the data is accurately displayed.

For the implementation of the UI, I aim to create **four main buttons** which will hang at **the bottom of the screen**; these buttons will have **the function of opening a list** of more buttons with a function associated with the pressed button. For example, **the Actions button will have a list of buttons for the different actions** available to the player.

The button categories will be:

- **Actions** - A list of actions the player can assign like chop wood, mine rock, ext.
- **Construction** - A list of buildable objects for the player to place on the map.
- **Colonists** - An ordered list of all of the current colonists with data relating to them, like their current task and their stats.
- **Zones** - Giving the player the ability to create, edit and delete zones within the map at will.

To find more information see **Appendix D**.

The controls for the game will be fairly simple.

To move around the map will be (WASD) **W to move up, A to move left, S to move down and D to move right**. An **alternate set** of these will be **the arrow keys**. Towards the bottom right of most keyboards. I shall also allow for the mouse to move around the map as well, so **if the mouse touches one of the edges of the window, the view will move in that direction**.

Ryan Hood | Student Email: p1617694x@my365.dmu.ac.uk
Email: Hoodrn03@gmail.com | Github: https://github.com/Hoodrn03

There will be **a zoom feature** in the game **allowing for the maps focus to be adjusted** during gameplay, this will be **controlled using the mouse wheel**, rolling it forward zooms in focusing on a smaller section of the map, but making the items in frame bigger, while rolling it back does the opposite.

Due to the map being a multi-level map, this will require **a way to traverse up and down the layers**, as well as having **a button in the UI** the **Page Up and Page down buttons will also work to change the current layer** on the map.

Most actions within the game will be controlled with **the LMB (left mouse button) it will be used to activate buttons select items and place objects**. The LBM will be the main control item for the game.

Accessibility

I aim to make the game **as accessible as possible**, this will involve me including **a number of features** into the game to achieve that. While browsing methods of increasing accessibility I found (http://gameaccessibilityguidelines.com/), this provided details **to improve accessibility**, here are a few options that can easily be developed into the game;

- To begin with, I will have **an options menu** which will remember the settings the player have selected upon startup.
- Next, I will include **an option to rebind the keys** to anything on the keyboard, as well as having **a number of keyboard shortcuts**.
- I will make the game **playable in both fullscreen and windowed**, allowing for the use of an onscreen keyboard.
- I will also **allow for the UI (user interface) to be resized**, primarily having three size options small, medium and large each with increasing font size allowing for people to easily see the UI.
- I will follow **a colour scheme which is not too garish**, ensuring similar colour are not overlayed on top of each other. Following **if I have additional time I will create a color blind mode**.
- I will **not use sound or coloured text to convey important information** within the game.
- I will have a selection of **multiple different difficulties** for this game ranging from easy to hard.
- I aim to add in **a system to both speed up and slow down the gameplay** allowing for people to take their time whilst playing.

Ryan Hood | Student Email: p1617694x@my365.dmu.ac.uk
Email: Hoodrn03@gmail.com | Github: https://github.com/Hoodrn03

- I will have **a controls menu within the options screen**, allowing for people to easily refresh on the controls.

Conclusion

And this concludes the game design document. This document covers the entire project, spanning the main gameplay items including a simple wish list for items I ideally want to add. I have also covered some additional considerations for this project including User Interface designs and accessibility items I aim to include for the game.

Following this document will be some appendices they will expand upon some items touched upon within this document and allow more information on items if required.

Appendices

Appendix A - Diagram showing the map and how the multiple levels will look for the player. It is a simple drawing consisting of straight lines and basic connotations on each square of the grid separating the different components of the map. The map shows only the minimalist detail showing the general idea of the generation without the inclusion of resources or player made buildings in the world.

Appendix B - Simple diagram of a potential building which will be constructible within the game. It shows a building with multiple walls, doors a couple windows, and some stairs. One of the main concepts for this diagram is to also propose how the multiple layers of the map should affect the options for building, as shown by the diagram there are three layers, both continuing the building from the ground level.

Appendix C - Is a collection of various mood boards which will serve to act as inspiration for the art style for the game. The final art style for the game is currently not decided and during the implementation process, the games art will be created as it is needed. There are three boards; furniture, character and texture.

Appendix D - An interface design for the game. It has multiple levels showing the hierarchy of menus within the game. This will allow for the game UI to be easily implemented. This is an initial draft and is subject to change upon a moments notice.

Appendix E - the project plan for the project, this also includes the Gantt chart linked within the document. It covers the main milestones and the implementation strategy for the project.

Ryan Hood | Student Email: p1617694x@my365.dmu.ac.uk
Email: Hoodrn03@gmail.com | Github: https://github.com/Hoodrn03

Appendix F - Bibliography for the project, It contains a small list of most if not all of the resources which I have used to create this document and any additional documents created for this project.

Appendix G - Project contract, a confirmation of the deliverables and objectives for this project, signed by both me (Ryan Hood) and my supervisor (Shengxiang Yang).

Appendix H - Ethical review form, an evaluation of my project to ensure it is an ethical project.

Appendix I - A global checklist, this will ensure my project follows De Montfort University's global guidelines.

Ryan Hood | Student Email: p1617694x@my365.dmu.ac.uk
Email: Hoodrn03@gmail.com | Github: https://github.com/Hoodrn03

Final Year Project Dissertation.

By Ryan Hood.

Contents

Introduction — Pages 2-3
- Outline — Page 2
- Main Functionality — Pages 2-4
- Aims — Page 4

Body — Pages 5-12
- Setup — Pages 5-7
- Pathfinding — Pages 7-8
- Colonist Creation and Management — Pages 8-9
- Update — Pages 8-9
- Resource Generation — Pages 9-10
- Base Building — Pages 10-11
- Literature Review — Pages 12-13

Conclusion — Pages 14-17
- Product evaluation. — Pages 14-15
- Evaluation of approach used. — Pages 15-16
- Evaluation of tools used. — Pages 16

Introduction

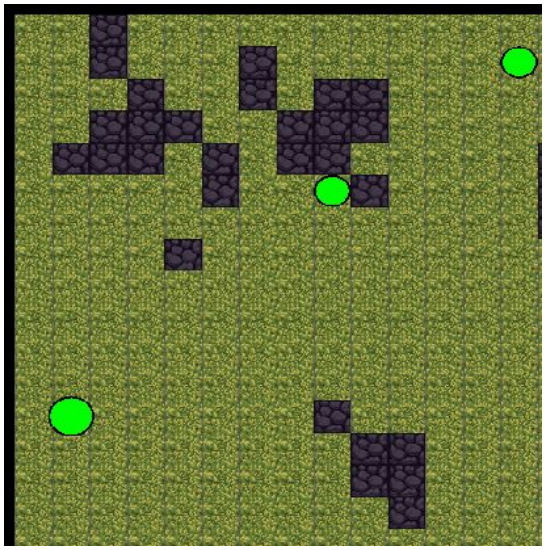
Summary.

The **project's main objective was to create a game**; the game was intended to be a **colony management sim**. Meaning the player would be given **a random map** with a scattered **assortment of resources** around the generated map and **a set number of colonists** (Pawns for the player to control). The player would then be able to **collect the resources** and use them to **assemble a base** within the game world. The main objective for the player within the game would be to **keep a number of colonists alive and happy**, they are able to do this by building objects like beds, tables, ext. within the world which will provide them with a number of different benefits. If at any point in the game all of their **colonists were to die then they would lose** the game. The game was designed to be **fairly simple** in terms of themes but have a **fairly large amount of replay value**, due to the fact that if the player were to lose the game they would be able to start anew without any repercussions.

Main Functionality.

Grid System. The game was made using a **tile-based system** creating a grid (down), the grid was created with **four different sets of data**; Cells, Columns, Rows and Layers. Columns are formed by a collection of cells, rows a collection of columns and layers are sets of grids; forming a **multi-layered map object** for use within the game. The game map is the most important component for the game, solely because **it will be used to house all of the other components** within the game.

Map Generation. Random generation within the game was a large aspect for the development. For the style of the game created it was a necessity, therefore needed to be done well. During the generation step upon startup, **there are three steps taken to form the map**. Firstly, **water is created** one of two different types of water is selected (River or lake). Secondly, **rock is placed** within the map with the potential to create large mountains or complex cave systems. Finally, the **dirt and sky are placed** within the map filling in all of the remaining cells **ensuring that the entire map is properly textured**.



Colonists. The colonists were an important feature for the game and were given plenty of consideration beforehand. They were **implemented with the A* algorithm** allowing for perfect **interaction with the grid system making movement** an easier system to create; using A* and cells allowed for creating both the open and closed sets simpler because cells having unique ids prevented double checking of cells and sped up the rate of finding paths within the game.

The colonists were also **designed with a level of requirements** to keep them going throughout the game. For example, the colonists will require sleep to keep going and prolonged sleeplessness will stop them from working and eventually they will begin to lose life and **once their health reaches zero the colonist will die** and be removed from the game. **Once there are no longer any colonists within the game the player should lose**.

The **colonists also have a set of skills** which will **help them perform specific tasks** within the game, for example, strength this is a randomly generated stat that the colonist will be assigned upon startup and it will determine the speed at which they will work a specific job. This will **allow for players who want to optimise speed** then they will be able to **assign the correct colonists to the correct job**.

Colonists stats include (but are not limited to):

- Strength
- Perception
- Endurance
- Charisma
- Dexterity
- Intelligence

Buildings. Buildings within the game are quite important, the type of game requires a certain refinement to the building system. The buildings are meant to be **the link between the colonist's needs and the resources around the map**. The list of buildings within a game like this needs to be varied enough to keep the game interesting but not too sparse as to make it intimidating.

In the game, there will be three different types of buildings:

- **Structures**; These will be used to form buildings using a pattern of other connected structures, these could be anything from a wall to a door to a window.
- **Floors**; Decorative objects used to cover the existing floor tiles, they do not replace the tile they are on, only cover them.
- **Interactable Objects**; These will be placed upon tiles and allow for certain functions to be performed by colonists while they share the same tile, for example, a bed is an interactable object and will allow for a colonist to sleep.

Resources. Within the game there will be **a selection of different resources** available to the player; they will need to use these to help keep the colonists alive. For example, **trees will spawn somewhere on the map**, if the player cuts down the tree a pile of wood will spawn in its place, finally the player will be able to **use that wood to create buildings** and other useful objects.

There will be a varied number of resources within the game and **each will be used for a specific action** there will be food available to harvest within the map and it will be used to feed the colonists. The wood and rock are specifically used to build objects within the game and can be collected by either mining at rock or chopping trees. Water can be collected and used to sustain the colonists similar to that of food. Each can be found within the map and will perform their own function. **The player must use their colonists to create/maintain these in the game** but it should be a simple task.

Draw Filter. Within a large game with **a lot of game objects** such as this was, it was important to **limit the number of objects being drawn** at a certain time, therefore I handmade **a filter** within all of the game objects to use **the size and position of the game camera** (game view) to **reduce the number of draw calls**, this was done in hopes of increasing the possible frame rate for the game.

Aims.

For this project one of the main aims were to **create a challenge**; something that was hard and took a lot of work to develop. This is one of the reasons **it was created using C++** and a **small number of external libraries** to give that extra depth for the project, everything else was to be designed and created by the developer.

Pathfinding for the colonists was a large aim for this project, to experiment with different styles and implementation strategies for pathfinding algorithms. From the offset, the goal was to **use A*** and raw euclidean distance but the refinement for this would be to **examine the differences between shortest time and shortest distance** due to the differing tiles within the game, giving the project an extra challenge.

Random generation was a key feature I was aiming to implement. I mainly **wanted to explore the process involved with generating random maps** and gain a new level of insight into the way other games approach this mechanic. The other reason I wanted to add random generation was to aid in the replayability of the game itself, thus creating an overall better game.

This project was not only a challenge to create within the somewhat limited timeframe it was also **planned to encompass most, if not all of the knowledge obtained through the three years of the course**. The project in all regards, although stressful at times was well worth the effort put into it; despite being barebones and a little faulty at times it has the benefit of showing the care and dedication the developer has placed within it. Just for that fact, the project has no better worth. And with only a little polish and just as much care, the game could become something better.

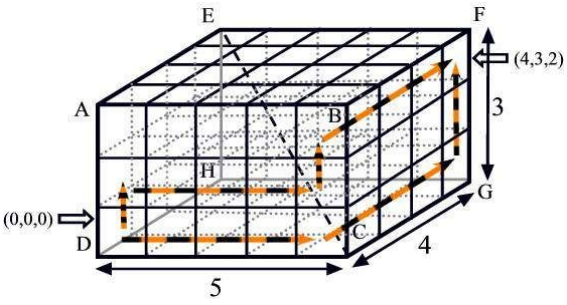
Body

The project consisted of six key development cycles; **setup**, **pathfinding**, **colonist creation** and **management, update**, **resource generation** and **base building**.

Setup

The project was **created with two main libraries SFML and TGUI** (move to the evaluation of tools used for more information on tools used), and the beginning of the project started with the setup of **the SFMl game window which will handle all of the renderings for the game**. The window requires a few variables to function but for the setup, it only needs a size (height and width) as well as a name, for this project it was taken to be called simply “Colony Sim”. Next using the size of the created window **SFML allows for the formation of an in-game view** which **works the same way a camera does**; it is made to be half the size of the actual game window upon startup.

The first part of development was dedicated to **creating a usable game map** which would at a later date hold all of the other features within the game. **Starting with a simple box** with an equal size to that of the window was made, **forming the container of the map**, using a set width and height it was simple to properly size the future cells within the box, (for example if the box was 100x100 and you wanted a 10x10 grid each cell would be sized to fit inside perfectly). Because of the game design, there was also an additional step to be made; a set of layers also needed to be created this was done by overlaying multiple grids atop each other **forming a three-dimensional map for the game**.



The grid was rather simple to form, I used **a set of for loops**, with the limiter to them being a set of **predefined columns, rows and layers** set by the developer. Passing the parameters into the setup allowed for easy testing and maintenance. Using smaller numbers was key to being able to perfect the strategy in which the creation process occurred. **The loop contains three nested for loops**; I, J and K each of them have unique functionality which culminated to form the finalized grid. **The loops work from the centre loop outward**. From **K this is where a temporary cell is initialized**, sized and positioned. At the end of the formation of that cell, the **cell is then added to a vector of cells creating a row** of cells. The loop J then uses the K loop to **form a number of rows and this populates the grid with both rows and columns**. The final step of the loop is

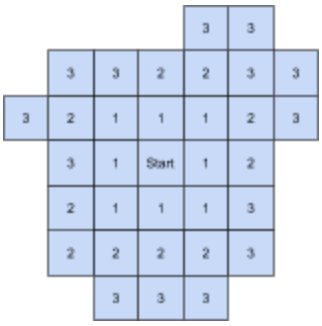
Cell	Cell	Cell	Cell	Cell	Cell

Cell	Cell	Cell	Cell
Cell	Cell	Cell	Cell
Cell	Cell	Cell	Cell
Cell	Cell	Cell	Cell

to use both J and K to form a set **number of grids so that they can be overlaid atop each** other. This forms the basis of the game map and allows for the next step of the setup functionality.

The next step for the setup stage of the game is **the random generation of the game map** and for this, a new class was created and lovingly coined “RandGen” a combination of the two words random and generator. This is a rather simple class with a **limited number of functions**, it's main and only uses being to **supply other classes with random numbers**. This class’s key method is an important and versatile one, it **requires a min and max value** (integer; whole number) and will randomly **select a value somewhere between the two**. This class is heavily inherited from throughout the program because of the substantial need for randomly generated values at numerous places from moving the colonist from one tile or another to the amount a tree object will grow in a given time. Because of the heavy use of random values within the game, it was necessary to ensure that **the numbers generated were guaranteed to be unique** therefore a seed was needed for the current program so, the developer added a simple line of code upon startup which uses the system time as **a new seed to ensure for new random numbers**.

When **generating a map** the first part is to **create water** within the game, one form this water could take would be to **create a lake** this will form **a large body of water starting at a single centre cell** and will **vary in size** between games. We start off **choosing a random cell** on what is selected to be **the ground level of the grid** (the middle layer), before beginning a random integer for the number of iterations is selected; determining the size of the lake. Then all of **the neighbours of the starting cell is added to a list of cells to potentially add to the lake**, two variables are also created to a chance to place the water tile and a degradation amount, these will both be used to **limit the number of tiles which are placed** within the game map. Next, we move into a for loop using the aforementioned number of iterations each loop we start by adding the neighbours of all of the current water tiles; ensuring no duplicates are added (to save on processing time). Then we **use the placement chance to add the tile**, by generating a number between the starting placement chance and the current placement chance, if the number is below the current placement chance the tile is added and we proceed to possibly add the next one. At the **end of the loop for the next iteration, the placement chance is decreased** to allow for gaps within the outer reaches of the water.



Generating rock within the game map is the second part of the overall map generation. The rock **works in a different fashion** to that of the water. It begins with a single for loop **iterating through the entire map grid**, each cell in the map has a certain chance to become a rock tile (excluding those already water) around 5% however, the plan to allow for complex mountainous structures **if a tile is adjacent to an existing rock tile the chance that this cell becoming**

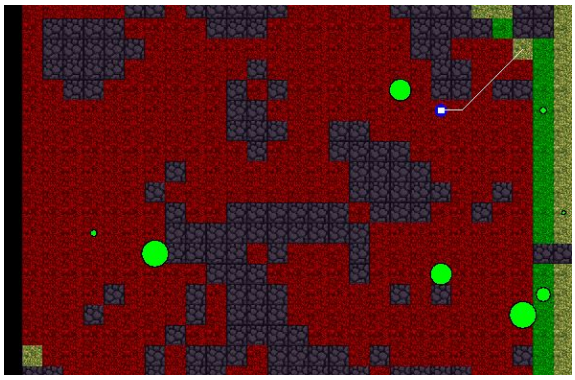
rock increases substantially to 25%. This system allows for the easy creation of large rocky areas within the game but also ensures that there are still untiled areas.

The final step of the map generation is the **dirt/sky filler**, this is rather important yet gloriously simple, **it loops through the remaining untiled cells** within the grid and depending upon the layer all cells **at and below ground level will become dirt** while those **above will become sky**. Thus forming the completed map for the game to then begin.

Pathfinding

When **finding a path between two points** the algorithm **requires two pieces of information** for its setup, **the starting cell** and **the end cell** each with their grid positions, neighbours and id. We start with a check that **the end cell doesn't have the same id as the current** (start) cell, this would mean that the algorithm has found the end cell. We then **add all of the neighbouring cells to the open set** (the list of cells to check, at the start only the starting cell) c**ross-checking the cells in the closed set** (the list of cells already checked, at the start the closed set should be empty). We then **move the current cell to the closed set** removing it from the open set, before the checking process, the **G, H and F score is then calculated for all of the cells inside the open set**.

The **G score** is **the total number of steps from the starting point and the ending point**, the algorithm uses the cells grid positions to easily calculate this, for example, if the starting point is (5, 5) and the end point is (15, 15) it would be 10 steps down the grid and 10 steps right to get to the end point. The **H score** is **the number of steps from the current cell to the endpoint**, using the same method as the G score the grid positions play a large role in the formation of the H score but this method makes use of a selection of **either diagonal or straight movement**. The main difference is that moving in a diagonal fashion would be cheaper than moving straight, diagonal movement is equivalent to 14 points opposed to 10 while moving straight therefore 14 is cheaper than 20 points. The **F score** is **the total of both G and H scores** and will be **used in selecting the next cell along the path**. Once all of the cells have had their values updated the next **cell with the lowest F score is selected to be the current cell**, immediately after the current cell is updated the cell previously the main cell becomes the parent of the new current cell. This process is repeated until the end cell is reached by the algorithm.



Next, **the tracing of the path occurs**. Starting **from the end cell the program follows the tree of parent cells** adding the cells past **into a deque** until the original starting cell is reached. Once the final path is found it is outputted from the algorithm to be used. **The deque works**

with a first in first out approach, allowing for the start cell to be at the beginning and be used first.

The second form of **water generation** within the game **required a level of pathfinding** to ensure it worked correctly, this was **creating a river** within the game map. This worked by selecting one cell on one of the sides of the map (top, bottom left or right) to be the starting point and then one cell on any of the other sides before **finding a path** between them. Next, was following the path setting **the tile for each cell inside that path into water tiles**. Following, a **width for the river was generated** and looping through all of the water tiles to **transform all tiles either side of the initial tiles into water tiles**, ending in a river through the game map.

Colonist Creation and Management

The colonists where if not the biggest part of the game and required a lot of time and consideration before implementation. We **began with some basics rendering and positioning within the game world**, this was rather simple to achieve using **SFML's rectangle shape** for it handles most of the positioning within its window object. After the primary colonist was created and placed within the game world it became apparent that multiple of these would become quite troublesome to maintain; so **a manager class was created to manage multiple versions of the colonist class** at once. And for the set up it also allowed for multiple **colonists to be assembled** with only a single integer value.

It was important to figure out how **the colonists' condition and stats** were implemented in the correct way as **to prevent multiple redesigns**, therefore the plan was to **leave them for later versions** of the game when their uses could be implemented at a similar time. For example, once a system to allow for a bed to be created is devised then there would be little to no use in the colonists requiring a sleep counter.

In the game, **the way colonists get from point A to point B** was a crux and needed to be implemented **in a way which was fairly open** to change. The way it was set up at the beginning was **using the previously created pathfinding algorithm**, yet there was little to **no interaction between the colonist and the grid** so it was hard to determine a start despite a random end could be found simply. After much trial and tribulation, we were **able to insert a pointer** (a locally stored variable which points to the location of a certain piece of data) for the **colonist's current cell** which would then be **updated every time they move**.

One of the biggest **issues for the colonists' pathfinding** was that there was a **huge framerate drop during the A* algorithm**, many different options were tried in attempts to address this issue but we found that one of the more effective methods were to **place the pathfinding for the colonists into a new thread** (a new operation which happens simultaneously to another) separate from the ongoing game loop.

The way that **the pathfinding was implemented** allowed at a later point to **easily pass a cell into the colonist** and the **path would be found** from their current position to that cell, this enabled the effortless **pathfinding from colonist to other objects** provided they also had a pointer to their current cell.

The game requires a level of detail **to look presentable**, to make this happen we require **textures and fonts** within the game. To insert these items into the game **it will require loading the files** from a local data folder, luckily **SFML helps with the loading** part. The only thing is that SFML doesn't offer an easy way to manage them, so for the project, both **font and texture managers** were a must. These **managers only purpose is to store the loaded textures and fonts** and allow them to be **easily accessed throughout the project**, the easiest solution for this would be to use **a map object** and store the loaded files with an identifier, in this case, **a string was used** so it had an easily recognisable name in the map.

Update

While there are plenty of functions within **the SFML library** which handles a lot of **mouse-related operations**, like button triggers and desktop mouse position it still **didn't encompass everything that was needed** therefore a new class was created to expand upon what was already there. Most importantly, a method to convert from **the mouse's literal position within the desktop to that of the game window**. It was also given a little extra functionality, like showing the tooltip of which tiles the player is currently hovering over **adding a little quality of life** and allowing for a selection box to be drawn, showing what the player has dragged their mouse over while holding down the LMB (Left Mouse Button).

The game required **a level of control from the player's** point of view because **the game was top-down** and **quite big** this allowed for many different ways to give the player control over the map. We first implemented **a way to move the view around the game map**, this was difficult at the beginning due to the complications around **passing information between the SFML event objects and handmade classes**, eventually settling creating **an event handler class** with it holding **local boolean values** for when a button is pressed and accessing then when it was needed. After the event handler was added it became much easier to implement other similar methods, so in addition to the **left/right** and **up/down** movement **a scroll feature** was implemented allowing for the view to be shrunk or increased with the players' mouse wheel, both of these features allow for the player to effortlessly navigate around the game map.

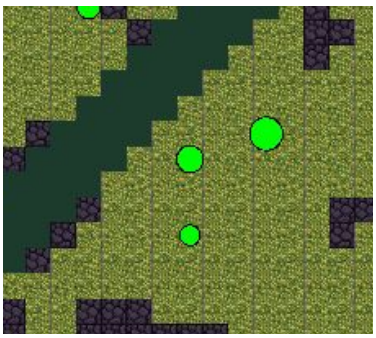
At this point, **the game had a big performance hit**, one of those issues was each frame there were **one hundred or more drawing calls**. This atop the pathfinding, the colonist management and more to come was instantly making itself apparent that this couldn't remain, therefore a plan was devised to **create a hard limit on what was drawn at a time**. So, **a draw filter was created**, it was placed within the game object class made to be inherited from whenever a game object existed within the game world, **it would be updated semi-regularly** and prevent

anything from being drawn outside the game view. This turned out to be a great step and helps to improve game performance.

The Gui (Graphical User Interface) was a stage of the implementation which came with a few worries, unsure how the implementation would turn out and required a fair amount of research into different potential libraries, finally settling on **TGUI** (https://tgui.eu/). Not only did it flawlessly **interface with SFML** which was a must it was also well documented allowing for speedy development. Within a single development sprint, a UI (user interface) was designed, mocked up and inserted into the game. This did, however, come with its fair share of performance problems which require complex debugging cycles.

Resource Generation

For the production of the game to continue there needed to be something for the player to do, of the bigger components to other similar games are base building and this works really well with the style of game being produced, therefore **resources** are a pivotal part to the implementation to structures within a game. When making a resource there are many factors to mull over, like **how are they placed** within the game world? How the **player would be able to interact with them**? **Do they drop anything** if so what? When thinking about the potential buildings the foremost building material available is **wood** so, **trees** sprang to mind and the implementation ensued.



For the **trees** in the game it was necessary to think over the interaction they would have within the game world, therefore it took a few designs to get into a working state. It began with the development of **the main body for the tree object**, inheriting from the previously established game object class it was uncomplicated to place a tree manually within the game map. Attendant to this was **the interaction between this object and the main map object**, due to the tree's real counterpart it seemed realistic to **place the tree on a dirt tile**, consequently, it required the game maps ability to find a relevant dirt tile, which proved somewhat difficult. This resulted in a single for **loop of the ground level within the grid until a dirt tile is found** and randomly selected, however, this process is simple it drastically **increases the time it takes to generate a map**, with the addition of multiple tree objects, despite this the tree objects are generated in non-uniform fashion as designed making the game map look like a more natural landscape.

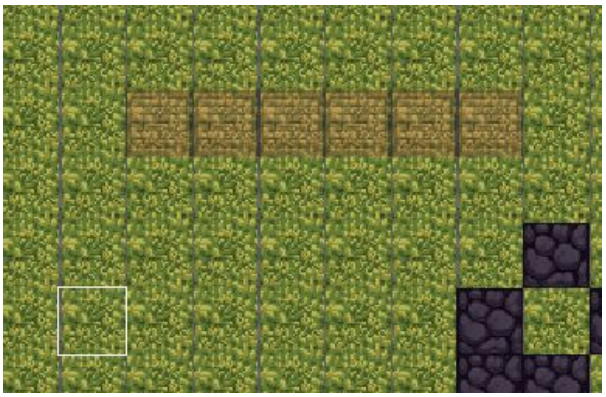
In addition to the implementation of **trees** within the game, the development of them was not finished there, **the product of those trees** was needed to allow for **the interaction between them and the player**. Then was the introduction of **woodpiles** into the project, not only would this be a clear product of cutting the trees it would also prove a clear link to the chain of a **cut tree, get wood and make a building**. At the start, the tree instantly created a **woodpile** for

testing purposes, but it was important to allow for the changes to the way that the trees themselves worked, it was important that **the trees and woodpiles have a little variation** to spice up the game world, so, the trees were **given a growth meter** allows for the **size of the trees to change slowly** during the gameplay, this would also allow for a **varied amount of wood to be dropped** when the tree is cut down, for example, a smaller tree yields less wood.

Now with the implementation of a second resource within the game, it was becoming much harder to **manage all of the resources** by themselves; thus **a manager** would need to be created, this game has **a lot of managers** already but they become useful when the need to manage an undetermined number of unique game objects within a game. A **resource manager** would slow be fairly useful to **convert a single tree object into a single woodpile** in a single function. The addition to a **resource manager would also enable a new resource** requiring similar functionality to be added to the game.

Base Building

Although the resources within the game add some extra detail within the level generation and allow for a more realistic map, however at this point they are without use and must have a form on interaction with the colonists in order to achieve their full potential. Within the colonists, **a job system was implemented**; a **simple enum** value named after the job, for the moment it remains at **only two**, an **Idle job** which is limited to only them moving to random points on the map (making use of getting random dirt tiles) and **a logging job** which when active should look for a tree and have the colonist cut the tree down. When the colonist has the logging job they will loop through **the list of trees assigned to be cut down**, select the closest one and store a pointer to it. Once the pathfinding algorithm has done the thing and **the colonist shares a cell with the tree the colonists should proceed to cut it down**; making use of their strength modifier to affect the speed at which it is removed from the game. **Once the tree is chopped down it is replaced with a pile of wood** and the colonist will move onto the next closest tree which is also marked to be cut down.



The next portion of the game development was dedicated entirely to **allowing the player to place buildings in the game world**. The first step was planning on the way the player would place the buildings within the world and it was decided to use what is currently called a placeholder or **ghost building**, this acts as a ghostly **outline of what the placeable building would look like**. It began with the implementation of a **cell-sized body which would follow the mouse cursor** while **snapping to the cells** (gridlocked) whenever the player would enter “build mode”; this would allow for the player to know exactly where they were building, but also really useful for testing.

Following this, it was important to consider how the building would later **interact with the player's colonists**, so **when a building was placed it would remain in an unbuilt state** with an int of how much work the building would take, this would allow for simple checking every few frames to see if the building has been built, if so then **the building would upgrade from a ghostly transparent version into the fully textured version**. This was to add a clear indication to the player if something was built or not.

Next was the part the **colonist would take in the building objects**, this resulted in a new **construction job**; this took inspiration from the existing logging job and followed most of its methodology. The **colonist will look for the closest building**, find a path to it and the work it each frame using one of its stats to affect the rate at which the building is constructed. This allows for the game to feel like each of the individual components created interlocked and came together to do something.

But there was one thing missing; the colonists walk to the building and make it. It was missing something and the connection to the wood pile was initialized. At the point, that the building was commissioned to be built a new was designed and assigned which would hold the amount of **wood required for the building**, so for example if the player wanted to **make a wood wall it might require the addition of 15 pieces of wood** the player must collect before they can start working. This **tied together the two separate jobs** of cutting down trees to making a structure.

Literature Review

Abstract

When making a game it is really important to design the game correctly and there are many discussions on the "correct" way to design a game. This study will look into some materials showing their opinions of the correct way to design a game with the hopes to expand knowledge on the processes of game design.

What skills do you need to design a game?

When first designing a game there are skills that are associated with it, some say that you only need one Richard Rouse in his book Game design: theory and practice (2005) suggests that focus is all you need and with that key skill you can avoid getting sidetracked *'sidetracked by technological obstacles that are thrown in your path, sidetracked by altercations between team members, or sidetracked when your publisher tells you features A, B, and C simply have to be changed'*, while other designers take a more haphazard form of thinking when it comes to the

skills they consider important for the game designer; Jessie Schell in his book The art of game design: a book of lenses (2008) claims that skills from any and every discipline are useful for designing a game, *'Almost anything that you can be good at can become a useful skill for a game designer.'* and while this is true and most skills can be utilized he still makes the critical distinction that in his opinion listening is the most important skill, *'The most important skill for a game designer is listening. Game designers must listen to many things. These can be grouped into five major categories: Team, Audience, Game, Client, and Self'*. For the most part, the industry experts seem to agree that I *' A game designer needs to possess many, many skills'*, quite aptly put by Scott Rogers in his book Level up: the guide to great video game design.

The three authors seem to agree on one thing, a developer can come from anywhere in the team, be it a programmer, artist or even an anthropologist. They remain in sync throughout their respective books advising that one of the key features of a good game designer is working with your team.

What is a game?

The three authors Jesse Schell, Richard Rouse and Scott Rogers all seem to agree on the point of what is a game. It is important to keep it simple. And I believe that Scott Rogers says it best *'Q: What is a game? A: A game is an activity that: • requires at least one player • has rules • has a victory condition. That ' s pretty much it. '*(Level up: the guide to great video game design), all of their descriptions follow that simple approach and well, its rather fitting, a game should be something uncomplicated.

What are games made of?

When it comes to the bones of what games are they are easily broken down into four key areas, aesthetics, mechanics technology and story. Each is important to form the finished game and they are all useful in their own way, it all depends upon the game which is being designed, for example in Level up: the guide to great video game design Scott Rogers mentions *'Some games need a story. Some games don't. All games need gameplay.'* which is quite true in my opinion but Jessie Schell disagrees pointing out that *' none of the elements is more important than the others.'* probably eluding to the fact that all games in one form or another will have a story regardless of intentions.

Summary

We have concluded that from the research that a game designer could be from any background and contribute to the design of a game, but the most important skills to have are to listen to others and have focused on the tasks at hand. We have also discovered that in the opinion of industry game designers games are a simple art form with only a few requirements. And what goes into a game is an equal combination of four different elements none of which should be a lower priority than any of the other elements.

Conclusion

Product evaluation.

The game at the time of writing is fully functional and it encompasses most if not all of the aims which were set at the start of the project. The game indeed fits within **the colony sim category** containing all of the major tropes; **base building**, **resource gathering** and most of all **colonists to control**. The game while at this time has neither a win or lose condition the game is still fully functional. Personally, **I think that the game is good**, a little lacklustre and it will still need a fair amount of optimization and cleaning up to make it a good gaming experience but one of the main goals I set for myself was to reach the end of the project with something I was happy with and right now. I'm at that point.

I am most proud of **the implementation of the A* algorithm** within the game, I chose to tackle **the implementation of it very early in the development cycle** and if it ended poorly it would set a bad precedent for the rest of the project and the fact that it works and **works efficiently**, makes me glad. In addition to this, the **interfacing between various classes** are at a working state which during the process of making them talk to each other became a long one, requiring many designs and redesigns, but when they finally started to interact with each other the process just slid into place, turning one of the biggest worries into one of it's bigger assets.

But, with the timeframe of the project drawing to a close, **there are many tasks which remain undone**. If I had more time the first step would be to properly add in a random generation and implementation of **the colonist's skills** with **appropriate UI** to easily show to the player what they are and what it would allow for the player to do with them. I would love to spend a week **improving the UI itself**, at the moment it appears blocky and plain, rather unfitting in the game and brings the overall aesthetics down a fair margin. Furthermore, some time to **add in more resources, and buildings** into the game would make the gameplay substantially more varied. Not to mention the **limited list of jobs for the colonists**, limited only to cutting trees, construction and idle, with more time I know I would have implemented more varied and exciting jobs into the game; jobs like planting, mining, ext.

However, in the game, there are **a few features which just were left out due** to the approaching deadline. **A fully implemented options menu**, with interactable options, this was a feature intended to improve accessibility for the game and other more critical functions popped up this got pushed to the back. **Loading and saving game maps** to a local file, this was one of the more ambition items in my development schedule, but as other tasks took priority, this was shelved to the wish list and forgotten over time, but there are still times where I would love to tackle the loading of a completely randomly generated map from a file. Most Importantly one feature I had intended to add from the offset of the project was **the random events system** wherein after a certain amount of time an event would happen to spice up the

gameplay, but not enough items from the wishlist were implemented and this feature never got the chance to see the light of day.

Evaluation of approach used.

For the basic management of the project, **I created a Gantt chart** which outlined which portions of the game would be developed at that time. However, **this required many updates** to the tasks and when the tasks are performed. Using the Gantt chart it made **the agile development strategy** much easier to prep for. The agile methodology I used was in the form of **many separate sprints**; at the start of a new task, **the implementation was designed**, then **added into the game's code** and **finally tested** to ensure it worked with everything else currently in the game. This approach was more than sufficient for the project type being that the game was designed long before the development process actually started, however, this meant that **the project was subject to change at various points** of the projects life cycle. The size of the team, being limited to only a single person meant that if there was any change needed then it would be far easier to plan what was needed knowing my own strengths and weaknesses.

Considering that the final project is in a working state, I feel confident in saying that **the planning techniques I used were ample** and the development structure and implementation strategies that I used were pleasant. **The only drawbacks** for the project were in that **not enough meetings were attended by me**, solely my fault. I am unsure of the full reasons for my absence but this weakened the project significantly, with the lack of updates and progress reports. This was no strategy, but only a setback.

I learnt a lot during the development of the game. Foremost **I practised the A* algorithm** cementing my understanding of its workings and feel more than confident in my ability to be able to replicate its functionality; not only this but **I discovered the effects of changing how the H score calculates** within the algorithm allowing for a more refined system to be created. **I was determined to generate a random map** in the game, this paid off massively, I have **a solid grasp on the way different styles of game maps could be created**, this will help in the development of plenty of games in the future which require maps, randomly generated or not. I also **learnt many new C++ operations** for example for button input within the game it required me to learn about **lambda functions**; to allow for pathfinding I learnt how to properly implement a **new threads** into the game and so many more, these will ultimately appear in more game projects I take part in making this project more than worth it for all the new knowledge.

If I had to perform the project **from start to finish a second time**, I would begin by tackling the project from the development stage more loosely, during **the initial planning stage I was far too rigid** in what had to be in the game rather than it would be cool if this was in the game, this made **the development stage of the project far tougher** for the simple point that **it was stressing** to know you have so much work to get done to make the game you've envisioned.

And another thing is that I would **take more time and consideration towards optimization**, this is a big thing with all games if they are crudely optimized then they will obviously run poorly. This was **one of the harder things for the project** for me to get right, in the end, it is still really badly optimized but **I don't have any idea how to address this issue** especially as far along as the project is.

Finally, I would **include my assigned supervisor more**, this was like **handicapping myself**. I didn't use a resource which was there for me and there were many points where if I went to a meeting or even sent an email I would learn so much more, benefitting both me and the project.

Evaluation of tools used.

The main language that I used was **C++** within the IDE (Integrated Development Environment) **visual studio**, this was the language and environment I felt most comfortable with due to **using them for most courses** throughout my three years at university. Visual studios console was really **useful for debugging** as well as just creating the code. C++ within visual studio (specifically Visual Studio 2017) is **one of the easier IDEs** I have used and would use it in future projects given the chance.

The main library I used within the game was one called **SFML**, its main functionality is allowing for **simple games to be created using C++.** It handles the **creation of a game window**, **event calls** and most importantly the **game loop** itself. I have used SFML for a few simple projects in the past, and it is in most parts easy to use. Therefore I planned to use this because I was **interested in making a game using C++** but not many game engines allow for C++ so I resorted to the library I know and enjoy working with. After this project, I feel more confident with SFML and will make more projects with this library, although maybe on a lower scale.

The **TGUI** library was a really useful one, it **links directly with the SFML** game window allowing for developers to focus more on the functionality of the widgets it allows to be created. The only problem with **TGUI is it is far more complicated** and requires a lot of reading and researching into how things work, but luckily the library has a **really comprehensive documentation page**. If I were to make another project with SFML, TGUI is an instant include due to its implementation of GUI elements into SFML.

The most important resource which was used for this project was **Github**, not only did it allow for me to **easily access my files at different locations**, but it allowed for me to **source control** the project. If I were to make a mistake I could easily roll back to previous versions, and mainly with **Github Desktop**, the interface is **easy to use**. With the addition to a **Github student** account, I am able to simply create a **private repository** keeping my code to myself. Github is a must for any and all of my projects I am a part of personal and academic.

Final statements.

I am beyond words with how happy I am to have the opportunity to take part in this final year project. It has been a hard time with many ups and downs but pushing through to get to the finish has been satisfying. I am proud of the game I have produced.

End.

Project Contract

Student Name - Ryan Hood.
P-number - P1617694X.
Programme - Computer Games Programming (G62041).
Email address - Hoodrn03@gmail.com or P1617694X@my365.dmu.ac.uk.
Project Title - Colony Sim/Management Game.
Project Proposer - Self.

Supervisor
The name, affiliation and contact details of the supervisor, if different from proposer.
Shengxiang Yang
syang@dmu.ac.uk

Introduction

In this projects **aim is to create a game** wherein a **random map** is generated for the player and using resources which have been placed on that map, they will need to keep their colonists alive. The game will include **base building** and **random events/encounters**. This project will also include a look into **suitable pathfinding algorithms**, comparing the **shortest path** and **shortest time**.

Project Background

As a gamer, I greatly enjoy playing colony sims and management games and I would for once like to try my hand at creating one of my own. I have a huge fascination with random generation and I would love to find a good method of implementing a method of map generation within a game scene.

Aim/Objectives/Deliverables

Aims: I aim to create a fully functioning game, within the colony sim/management sim category. It will include functionality including random map generation and AI pathfinding.

Objectives: a list of specific, measurable objectives, each of which is likely to result in a deliverable. They specify all the work tasks to be undertaken to meet the stated aim. They will vary from project to project, as every project is different.

The main objectives of this project will be:

- Creating a system for creating a **randomly generate a map** for the player, this will also include the **random placement of 'Resources'** within the map.
- Implement and **evaluate the suitability of pathfinding algorithms** within the context of a game.
- Create **a grid system within the map** allowing for the player **to create structures** within the game.

- Follow the **agile development plan** wherein during each sprint a plan is created, implemented and tested. **These sprints will be on a weekly basis**.
- Create models, such as **flowcharts** and **class diagrams** to describe algorithmic operations and implementation strategies.
- Create a **conclusive report** summarizing the development of the project with a **critical evaluation of the work performed**.

Deliverables: a list of your Project's deliverables with some general description. **Please list in your contract only those that apply and remove everything else.**

	Development Project
Final Submission Week 27	• Project Contract. • Ethics Form. • Project Plan Including Gantt Chart. • Global Checklist. • Literature Review. • Class Diagrams and Case Diagrams. • Interface Design. • Design Documentation. • Test Plan. • Critical Evaluation. • Software. • Appendices.
Viva Examination Week 31 - 33	• Oral examination (demo of your work)

Students on a BCS accredited course should consult the BCS checklist before completing their project contract, as it includes eight conditions that the project contract should fulfil, such as

- The contract contains an elucidation of the problem, the objectives of the project, and a risk analysis
- The contract states that the final report will contain a clear description of the stages of the life cycle undertaken
- The contract states that the final report will contain a description of how verification and validation were applied

Resources and Constraints

The software requirements for this project include; **Visual Studio**, **SFML a C++ library** which allows for the control and creation of critical game objects, for example,

a game window. **TGUI** will also be used within the project; it is a library which links to SFML and **allows for the creation of UI** (user interface) elements into the game.

The game will be **backed up using GitHub**, an online repository which is most commonly used for **source control** with programming solutions. I will be using it for that exact reason, to prevent loss of work and make it possible to try out new solutions with the added benefit of being able to revert to a previous build.

Sources of Information

The main source of information I aim to use for this project will be the **internet**, it is the most common place to find solutions for programming problems, due to the numerous **forums** wherein someone will have had the same issue at some point and already found a solution.

The other source of information I will use will be **personal contacts**, for example, **other students** who may have experienced a similar issue. Another option would be to contact a member of staff be that one of my **tutors** or my **supervisor**.

Risk Analysis

The **random generation of the game is a large part** of the project but, if for whatever reason it begins to become a roadblock for the project and I'm unable to complete it I will have to **concede the random generation and create a static map**. One where the tiles are placed manually and the map is the same every game.

If the implementation of a single pathfinding algorithm for the project eats up too much time then I may have to **forgo the implementation/comparison of any additional algorithms**.

If I am **unable to polish the game** to an industry standard then I will **have to settle for a prototype for the game**, a **proof of concept with working features** to **outline the main gameplay mechanics**.

Schedule of Activities

Having defined the tasks to be undertaken in the list of objectives, you need to prepare a Project Plan to show how you intend to carry them out: You may find it helpful to draw up a critical path diagram before drawing a Gantt chart.

For this project, I aim to split the work down into five separate activities. The first will be spent **implementing a map** and generating a random terrain for the game. The next, **a colonist system** where they will **perform basic tasks** which the player will be able to assign them. Following that, I will allow for the **player to build structures** within the map. Next, I will add a system to create **random events** in the game with **different outcomes**. Finally, an evaluation stage, filled with testing and report writing, this is when I find out if the project has

been a success or a failure. In the end, I will conduct a WWW (What Went Well) report.

Link to current Gantt Chart:

https://docs.google.com/spreadsheets/d/1IvRP2XnJYIO8LR09rxNJRsqJsWtvVfdTz-adm9MfdnA/edit#gid=0

Student _____ **Date** 22/10/18

Proposer (if other from the student and/or the supervisor) _____ **Date** _____

Supervisor _____ **Date** 22/10/18

Keep the signed copy somewhere safe: include it with your initial submission. Your supervisor will require a copy as well.

Project Plan

Development strategy

Using the **agile approach with the development of the game,** the game will be stripped down into **many different sprints** wherein **each week will be a single sprint** dedicated to the implementation of a game feature or function. At the beginning of each week, I will take a look at what is needed for the game, conduct **a phase of planning**. After **the bulk of the week will be for the implementation** of the component. Finally, the **final days of the week will be used for testing** and reviewing with **added time for any fixes**/changes that need to be made.

Milestones

This project will have **five main milestone tasks** which need to be completed; each one is important for me to achieve the objectives set for the project. **The first, a map which will be randomly generated** at the beginning of the game. **The second, implement a simple colonist** for the game, one with working AI pathfinding. **The third, implement a selection of resources** into the game and allow for the colonist to perform a basic set of tasks. **The fourth, A building system** for the game, which interacts with the multiple layers of the map. And **finally, a random events system** with a couple of different events with different outcomes.

Gantt Chart:

https://docs.google.com/spreadsheets/d/1IvRP2XnJYIO8LR09rxNJRsqJsWtvVfdTz-adm9MfdnA/edit?usp=sharing.

Ryan Hood | Student Email: p1617694x@my365.dmu.ac.uk
Email: Hoodrn03@gmail.com | Github: https://github.com/Hoodrn03

Project Timeline

Date	1/10/2018	8/10/2018	15/10/2018	22/10/2018	29/10/2018	5/11/2018	12/11/2018
Week Number	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
Task	Find a supervisor and propose the game project.	Complete any additional documents required by the supervisor based around completing the game project.		Evaluate and turn over/hand in all design documents	Setup the game project	Create a system that will randomly within the game engi	
Current Week							
Date	4/3/2019	11/3/2019	18/3/2019	25/3/2019	1/4/2019		
Week Number	Week 23	Week 24	Week 25	Week 26	Week 27		
Task	Test the buildings within the game to ensure they work correctly.	Implement random events into the game using a list of possible events each with their own functionality.			Test Events and hand in final year project.		
Current Week							

Key			
Task		Current Week	
Break		Week Ended	
Testing			

19/11/2018	26/11/2018	3/12/2018	10/12/2018	17/12/2018	24/12/2018	31/12/2018	7/1/2019	14/1/2019
Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15	Week 16
generate a map ne.	Conduct a test evaluating the map generation and make any changes to fix any bugs.	Implement some basic colonist functionality which allows for them to move around the map, using a path finding algorithm.				Test the viability of the colonists using the algorithm, perform some stress tests with a large number of colonists. Make any required changes to the colonists.	Place resources randomly within the map. which will later be used by the player.	

End Of Project Implementation.

21/1/2019	28/1/2019	4/2/2019	11/2/2019	18/2/2019	25/2/2018
Week 17	Week 18	Week 19	Week 20	Week 21	Week 22
Implement a system which allows for the colonists to perform jobs, using the previously placed resources.		Test the placement of the resources with different maps and test the collection of them.	Using the resources allow for the player to place building blocks within the map and create a building system.		

IMAT3451 FINAL YEAR PROJECT - ETHICAL REVIEW FORM

The University requires all undergraduate final year projects to undergo an ethical review and, where human research ethical issues are identified, to ensure that these issues are addressed.

For the majority of Computing Final Year Projects, the outcome will be either 'No ethical issues' or 'Minor/Major ethical issues which have been addressed'; in these cases, approval can be given by the supervisor. In the unlikely event that the outcome is 'Ethical issues that have not been addressed', the completed form will need to be forwarded to the Faculty Research Ethics Committee.

Student Name

RYAN HOOD

Programme

COMPUTER GAMES PROGRAMMING

Project Title

Colony Sim / Management game.

A brief description of the proposed activity and its objectives:

Create a game or game Prototype within the Colony Sim genere. The game will include AI Pathfinding and random map generation.

Ethical Issues Identified:
(see overleaf)

How these will be addressed:

Checklist

Has the project proposal identified any of the following research procedures?

1. Gathering information about human beings through Interviewing, Surveying, Questionnaires, Observation of human behaviour — Yes / (No)
2. Using archived data in which individuals are identifiable — Yes / (No)
3. Researching into illegal activities, activities at the margins of the law or activities that have a risk of personal injury — Yes / (No)
4. Supporting innovation that might impact on human behaviour e.g. Behavioural Studies — Yes / (No)

If 'Yes' to any of 1-4 above: have you considered the following?

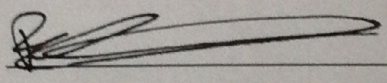
- ☐ Providing participants with full details of the objectives of the research
- ☐ Providing information appropriate for those whose first language is not English
- ☐ Voluntary participation with informed consent
- ☐ A written description of involvement
- ☐ Freedom to withdraw
- ☐ Keeping appropriate records
- ☐ Signed acknowledgement and understanding by participants
- ☐ Consideration of relevant codes of conduct/guidelines

Ethical Review Outcome

- ☑ 1. No ethical issues
- ☐ 2. Minor ethical issues which have been addressed and concerns resolved
- ☐ 3. Major ethical issues which have been addressed and concerns resolved
- ☐ 4. Ethical issues that have not been resolved/addressed

Authorisation

If the outcome is no. 3 or 4 above, this form should be forwarded to the Faculty Research Ethics Committee.

Signature of student  Date 22/10/18

Signature of supervisor Sheryl Yuen Date 22/10/18

IMAT3451 FINAL YEAR PROJECT - Global Checklist

The University requires all undergraduate final year projects students to undertake a global review of their project. Here is an International Impact Checklist for you to complete, which can be done in consultation with the project supervisor.

Student Name

RYAN HOOD

Programme

COMPUTER GAMES PROGRAMMING

Project Title

Colony Sim / Management game.

Please indicate which of these possible attributes is addressed by your undertaking of this project.

Possible Global Experience	Addressed by Project
Ability to work collaboratively: teams from a range of backgrounds and countries	
Excellent communication skills with a sensitivity to speaking with and listening to non-native English speakers	
An ability to embrace multiple perspectives and challenge thinking in a range of cultural context	
A capacity to develop new skills and behaviours according to role requirements	✓
An ability to negotiate and influence clients across the globe from different cultures	
An ability to form professional, global networks	
An openness to/respect of a range of perspectives from around the world	
Multicultural learning agility (i.e. able to learn in any culture or environment)	

Brief description of how the ticked attributes have been addressed:

This will be the first Project of this kind I have ever attempted, This will require me to manage my time and resources appropriately.

Having access to a Supervisor is new for me, having someone who requires regular updates for the Project Progress; I will need to keep track of all work completed and be able to update the Supervisor on Progression.

Signature of student [signature] Date 22/10/18

Signature of supervisor Shengxiang Yang Date 22/10/18

IMAT3451 BCS Accreditation Checklist

Student Name - Ryan Hood.
P-number - P1617694X.
Programme - Computer Games Programming (G62041).
Email address - Hoodrn03@gmail.com or P1617694X@my365.dmu.ac.uk.
Project Title - Colony Sim/Management Game.
Project Proposer - Self.

Supervisor
The name, affiliation and contact details of the supervisor, if different from proposer.
Shengxiang Yang
syang@dmu.ac.uk

BCS Accreditation

Your supervisor needs to check your contract against this list and sign if you are on a BCS accredited course. Take note of this and be sure that you mention all requirements.

This contract contains an elucidation of the problem, the objectives of the project and a risk analysis	Yes ☑	No ☐
The contract states that the project will include an in-depth investigation of the context and literature, and where appropriate, other similar products	Yes ☑	No ☐
The contract states that the final report will contain a clear description of the stages of the life cycle undertaken	Yes ☑	No ☐
The contract states that the final report will contain a description of how verification and validation were applied.	Yes ☑	No ☐
The contract states that the report will contain a description of the use of tools to support the development process	Yes ☑	No ☐
The contract states that the final report will contain a critical appraisal of the project, indicating the rationale for any design/implementation decisions, lessons learnt during the course of the project, and evaluation (with hindsight) of the project outcome and the process of its production (including a review of the plan and any deviations from it)	Yes ☑	No ☐
The contract states that there will be a description of any research hypothesis	Yes ☑	No ☐
The contract states that all research will be fully referenced	Yes ☑	No ☐

Contract is suitable for BCS Accredited Project	**Yes** ☑	**No** ☐	**Supervisor Signature**	Shengxiang Yang.

Student _____ Date 22/10/18

Proposer _____ Date _____

Supervisor Shuyuan Yang Date 22/10/18

Then keep the signed copy somewhere safe: include it with your initial submission. Your supervisor will require a copy as well.

Bibliography

I have used an image of a three-dimensional map from this person, they do not state their name or the date of publication, but I have used this image to display the style of map I have used within my final deliverable

Unknown. (Unknown). *Mazes in Higher Dimensions.* Available: http://isaac.lsu.edu/uva/120//12051.html. Last accessed 28 March 2019.

For the literature review, I have used the following three books. They were useful to gain some insight into the way industry game designers think and approach a new game project.

Jesse Schell. The art of game design: A book of lenses. 2008. Last accessed 4 April 2019.

Scott Rogers. Level up: The guide to great game design. 2010. Last accessed 4 April 2019.

Richard Rouse. Game design: Theory and practice. 2005. Last accessed 4 April 2019.

Sergey Galyonkin. (2015-2017). *Steam Spy Top By Playtime.*Available: https://steamspy.com/. Last accessed 1st October 2018.

I have used SteamSpy for some of my market research, it is a useful tool to find out how popular a game is based on how many peek daily and hourly players a game gets. https://steamspy.com/ :- It is not always reliable due to the info it displays is updated every two weeks, but it is good for getting a basic insight into a game's average player base.

MachineGunLuke. (2018). How did your colony die? Available: https://ludeon.com/forums/index.php?topic=41809.0. Last accessed 1st October 2018.

After some research into possible story elements for a colony sim game, I had options on scenarios, an overall goal, ext. I found a forum thread (https://ludeon.com/forums/index.php?topic=41809.0), where people talked about how their colony died, this is how I decided to go with a random event system because from completely random situations the most fun gameplay is possible.

Multiple. (2018). Pathfinding. Available: https://en.wikipedia.org/wiki/Pathfinding. Last accessed 1st October 2018.

I used this page to look up potential pathfinding algorithms which could be used in the game, it gave me a nice selection of potential algorithms helping me to decide upon A*.

Various Studios. (2016). Game Accessibility Guidelines. Available: http://gameaccessibilityguidelines.com/basic/. Last accessed 3rd October 2018.

I looked into ways to make games more accessible and stumbled upon this site, it was pretty useful in helping me find simple and easy ways to make my game more accessible.
