5 YEAR MISSION

Danyal Saleh

Registration: 2101128

A thesis submitted for the degree of Master of Science in Computer Games

Supervisor: Dr. Richard Bartle

School of Computer Science and Electronic Engineering

University of Essex

August 2022

Acknowledgements

I would like to acknowledge the assistance of my supervisor, Dr. Richard Bartle & other academic staff who have helped me achieve the full potential of this project.

Abstract

5 Year Mission, also referred to as 5YM, is a colony simulation game targeted toward understanding the decision-making process of an individual while playing a game. There are various different intentions behind a game, evoking different emotions which dictate gameplay, but this project will focus on games that attempt to form an attachment between the player and in-game entities. Colony simulation games are able to create a connection with the player while also providing the infrastructure for the player to make independent decisions, between prioritizing attachments or progression in the game. Research into an individual's decision-making process in games has been scarcely documented compared to in the real world, hence this project aims to add to this field which is ripe for investigation. 5YM was created with C# in Unity utilizing various algorithms such as Moore's Neighbours and Perlin noise for the map generation, followed by a grid system for an A* search algorithm integrated with Unity's Rigid body system for pathfinding. The objective of the game is to aid a group of stranded colonists to reach their home planet by directing them to do tasks as an overlord, collecting the necessary resources for the final 5 Year Mission home. The game presents various scenarios in which the player is typically given two options, attachment or progression of the game where their decisions are recorded to be analysed for the research purpose of this project.

Table of Contents

Literary Review

Game development begins for various motivations such as creative outlets, profit generation, education, and pure entertainment. No matter what the motivation, in order to achieve the games potential, it must attract an audience resonating with the targeted player base. There is no one method to attract an audience, with different genres of games resonating with each of their target audiences differently. To be able to resonate with an audience the game must invoke some emotions within each individual creating a type of attachment, foreshadowing this project's aims.

When observing individuals interacting with games, their approach towards identical scenarios have varying methods, an example of this would be in *Dishonoured*[1], an RPG (role-playing game) which provides the player with the infrastructure for a stealthy approach to a given scenario. This allows the player the choice between playing the game with a stealthy approach or not, resulting in the player having increased control over a game, making decisions based on their own motivations and emotions rather than playing the game in a monotone way. This is not to say that games that do not give the same type of freedom to a player do not invoke emotions, but have a different method to resonate with their player base. *Super Mario Bros. 2*[2] is a platformer game that does not provide as much decision-making freedom for how to approach the game but can evoke different emotions to create a connection between the player and the game, hence it has one of the largest players bases with variations of the game being released containing the same memorable characters. Considering this, games providing the infrastructure for the player to make decisions based on their own motivations and emotions will be the main focus of this study.

This project intends to explore what emotions are evoked with games that provide decision-making freedom to the player, how these emotions influence the individual's decisions, and whether the motivation behind decisions is attachment or progression based. Attachment is where parts of the game become personal to the player, such as when creating a custom character or even naming an entity in the game. This may cause the player to prioritize the well-being of their attachment over taking increased risks for progression in the game. This indicates a change in a player's approach at times, where more preparation is done to ensure the risk is reduced to a minimum. In some cases, it may be intended to have certain interactions/mechanics in a game to create attachments/emotions for the player, providing an infrastructure for this certain type of play style. For progression, the player is more motivated to prioritize advancing in the game and is more willing to take more risks whether it be to challenge themselves or try to complete the game. In this, the player's only goal is to complete the game and does not consider attachments highly unless there are benefits for progression. Another aspect to consider with progression is understanding the risk involved as that would further influence the decision-making of individuals.

While different genres of games have different approaches to creating attachment in games by providing the infrastructures between attachment and progression, this project will focus on colony simulation games, such as *Rimworld* or *Dwarf Fortress*, which tend to have the most balanced infrastructure to choose between attachment and progression towards how to reach the end goal. This would allow the players the most freedom in making choices between attachment and progression on their own accord. If the player is not pushed to play a certain way, what is the dominant decision-making approach that players take? With all of this in mind, this project aims to answer the research question:

Does an individual prioritize attachment or progression in their decision-making process when the infrastructure for each is provided?

This research aims to provide an insight into what individuals naturally prioritize in games, allowing for a greater understanding of what features in games create higher levels of attachment, in line with their target audiences.

Research

Research surrounding games and their influence on individuals' decision-making process has had less exploration when compared with other areas of computer science making it ripe for investigation. There has been research that can relate to this project regarding attachment, giving insight into how games create attachment. Character-avatar archetypes generated from narrative analysis can be re-analysed with dimensions of character attachment to highlight agency and intimacy[3] a study has found. The study aims to suggest a more integrative approach to aid the understanding of the gaming experience. Unlike most forms of entertainment, video games can reach another level of immersive digital environments by requiring players to interact through characters. The study outlines character attachment as:

- Identification with one's avatar
- Suspension of disbelief in accepting the avatars reality
- Control over an avatar's action
- Responsibility for an avatar's needs

Multiple previous studies collected players reporting on enjoyable video games where the results showed that the most enjoyable reports were where players had more control over their avatar. The study then categorized character attachment to relationship archetypes, putting the archetypes against character attachment as seen in the table below.

	Avatar as Object	Avatar as Me	Avatar as Symbiote	Avatar as Other
Identification (I am that avatar)	**Low** My avatar is a digital form.	**High** My avatar is *me* in digital form.	**Mid** My avatar is a part of me.	**Low** My avatar is its own being.
Suspension of Disbelief (Accepts Digital World as Real One)	**Low** The environment is a space of competition.	**Mid** I appropriate the world to fit my own view of it.	**Mid** I am able to visit my avatar's world.	**High** My avatar lives in a digital world with its own norms.
Sense of Control (Physical)	**High** My avatar is a tool for mastery of in-game challenges.	**Mid** My avatar is my social surrogate to accomplish my social play goals.	**Mid** My avatar and I use each other to accomplish negotiated goals.	**Low** I am a tool for my avatar; it tells me how to control it to accomplish its goals.
Sense of Care & Responsibility (Affective)	**Low** My avatar has no needs.	**Mid** My avatar *is me* - it needs what I need.	**Mid** My avatar and I know each other's needs.	**High** I help my avatar get the things it needs in his/her world.

Fig 1.1 Conceptual classification player-avatar archetypes[1]

The study demonstrates that the two divergent approaches to the player-avatar relationship, one looking at intimacy and agency between player and avatar, and the other looking at their psychological merging, are compatible with both perspectives.

Another paper discusses a reliable metric for measuring character attachment which in this case is defined as "the connection felt by a video game player toward a video game character" [4]. This would provide an outline to follow when creating the feeling of attachment in a game. This study mentions a scale that looks at four major factors of a game, self-esteem, addiction, game enjoyment, and time spent playing games. There are questions about the interaction between individuals and games, as even though there is no "felt" connection there is an actual tangible connection between the gamer and a fully functional controllable character. The paper states that character attachment is not a new concept, but has been present in audience-character interaction theories as an "individual's willingness to accept the world of the character as real." The paper breaks down character attachment into feelings of:

- Friendship
- Identification with the character
- Individual willing to suspend disbelief
- Feeling responsible for the game character
- Feel in control of the character's actions

The study surveyed 572 individuals (256 males and 312 females) and was a 95-item questionnaire dedicated to asking questions based on the above breakdown of character attachment. The study found relationships between character attachment and the RPG genre, which is a very character-orientated style of games, correlating with their theories. Further, the study showed that RPG gamers who score higher in character attachment have higher motivations for fantasy, diversion, and social interaction. In conclusion, the paper details a valid method in explaining what attracts people to the RPG genre and can be seen as an outlet between self-esteem and gaming. Even though the paper was orientated toward RPGs, the same concepts can be translated for colony simulation games.

Similar Games

Simulation games are a genre of games that is designed to mimic scenarios that one may encounter in the real world. This may also include scenarios that have been vastly exaggerated or in different world completely, this usually depends on the developers take and approach to making a simulation game. The main purpose of games like this is to provide knowledge over time relating to the type of simulation. These types of games provide a "simulated environment which contain a mixture of skills, chances, and strategies to simulate an aspect of reality"[5].

Within the genre of simulation games, the games have largely different objectives evoking different emotions from each other. An example of this is *Goat Simulator*[6] (GS) compared to *Flight Simulator*[7] (FS), where GS aims to be a game for comedic relief while FS aims to educate the user and also provides a way for real pilots to practice their skills for flying. With this in mind, Colony Simulation games seem to provide a simulation of a survival experience but from the perspective of an 'overlord.' This sub-section of simulation games aims to challenge a user's management skills where players are in charge of setting orders, setting schedules, organizing resources, and ensuring the survival of their colony. *Rimworld*[8] is a sci-fi colony simulation driven by an intelligent AI storyteller, currently being the top-rated game under Steam's Colony Sim category. The game provides infrastructure for managing a colonist's moods, needs, wounds illnesses, and addictions with each colonist being unique with various traits and conditions. Further, the game provides a large-scale survival experience creating an interactive world in which colonists must do tasks to ensure their survival, such as building shelter, mining ore, defending from raids, and crafting and recruiting new colony members. With the infrastructure the game provides such as the unique colonists, forming a connection with colony members is common, especially the ones which whom the player beings their journey. The player's experience with the game is set by the AI storyteller, which essentially is choosing the difficulty of the game but also dictates the types of events that occur during the game. The player must take risks throughout the game when different scenarios occur, allowing players to progress towards finishing the game by constructing a ship to take off.

The infrastructure this game provides allows for creating attachment with colonists in the game so as to delay progression the of the game, also having to take risks for the sake of progression in the game. This balance makes the decision-making process of individuals very personal, which would aid to answer this paper's research question.

Language

For the decision of what language to use for this project, the programming style would need to be established, followed by a way to represent the game between 2D, 3D, and text-based games. There are many languages that have the capability to make games but I believe the main factor would be the use of object-orientated programming[9] also known as OOP. OOP orders and maintains code, which is beneficial for any program to avoid writing thousands of extra lines of code. OOP does this by organizing code into objects which hold the state and behaviour of the object, indicating which characteristic or behaviour to do respectively. Languages like python that do not use OOP differ as they are interpreter languages making which for game development is slow and limited by things like speed and memory consumption compared to languages like C++ which use OOP.

When it comes to the representation of the game, text-based games would provide minimal visuals which would allow for creating the game more straightforward, but without visual elements, the game may not be able to create as deep of an attachment between the player and the game. With this in mind, between creating a 2D & 3D game, 3D is not necessary to be able to achieve the aspects that allow a game to create an attachment with the player hence 2D would be most preferable. Creating a functional colony simulation game within the provided time frame will be a challenge, hence using a game engine would be most advisable. This avoids spending time on the backbone of the game, such as the physics, allowing for more time dedicated to providing the features necessary for an enjoyable and balanced experience between progression and attachment for the player.

Unity provides a user-friendly interface with the main added benefit of being able to easily connect scripts to objects. The platform is very effective while rendering 2D or 3D scenes allowing for experimentation and changing of variables without having to re-access scripts once created. This, along with various other features provides a smooth workflow allowing for better time management also when attempting to program different features. The visual aids that Unity provides help understand the complexity of doing something in 3D and further can make 2D game creation more streamlined by providing the tile map tool. Unity also creates simplicity when it comes to making a game cross-platform compatible which is very important for the current climate of gaming where it is becoming an expectation. Other game engines can achieve the goal such as Unreal Engine and Godot, but I believe Unity will be able to allow me to make appropriate progress on creating the final product and further allow for live testing while providing visual progress as motivation.

Methodology

There are two main methodologies that this project could utilise, Agile or Waterfall. The Waterfall methodology follows a strict sequential order. This entails that during the development of this project the previous stage would need to be completed in order to move onto the next, for example the to be able to move onto development the previous stage of design would need to be completed fully and successfully. As waterfall is a sequential process, it is easy to manage specific deliverables to be met before moving onto the next stage of development. Smaller projects in particular benefit most from the Waterfall method as it has less complicated deliverables aiding in faster delivery of a project.

Agile methodology operates off the premise that the development and testing of the project occur simultaneously, creating a continuous cycle of development. This methodology allows for more communication between the stages of development creating more open communication between the developer and client. This process differs from Waterfall as Agile is a client-focused incremental process over sequential, allowing for the client to be more involved with the development process reducing their risk. The methodology relies on self-motivation and self-organization to ensure the quality of the project, therefore more personal effort is put into the final product.

These methodologies do have their downfalls, with Waterfall not being as useful for long-term or large projects with the requirements of the project being determined from the start as it is a sequential process. A sequential system means backtracking from the current stage during development is less flexible, making changes to the project more challenging. This becomes especially obvious in the testing stage of development where issues that are found may require a change in the design plan. Agile, while allowing for more flexibility, also entails that changes to any stage impacts all stages of development. Agile requires the developer to be as involved as much as possible with each stage of development to ensure that the outcome of the project stays clear, which leads to more time consumption especially with consulting with the client in general.

With these differences in mind, Agile was the chosen methodology for this project. Game development requires flexibility as numerous issues will arise throughout development, which may lead to solutions that differ from decisions made in previous stages. The added flexibility in the project management for healthy changes as progress is made through development. Agile introduces a mindset where the product is tailored towards the end-users, which will be reflected in the development of this project through frequent meetings with a supervisor. Further, regular meetings will aid in ensuring that the project's goals are clear through any developmental changes.

Planning

The planning stage is dedicated to laying a firm foundation for further stages of development. This foundation will account for any technical, or logistical issues that may arise allowing to set appropriate deadlines. To this end, a project vision is determined to be able to identify all the requirements for this project. Tasks will need to be done in an appropriate order, hence the dependencies between different systems will be analysed to identify the order, optimizing progress while decreasing chances to backtrack. Once requirements are identified they can be placed in a backlog as per Agile standard, to be worked on as iterations progress throughout the project.

Project Vision:

- Create a colony simulation game with reference to a 5 Year Mission theme.
- Create & record scenarios of attachment vs progression for analysis.
- Implement features that promote attachment.
- Create a balance between the benefits of progression vs attachment so as to not overly favour either.
- The game must provide some benefit to the player.

Project Roadmap:

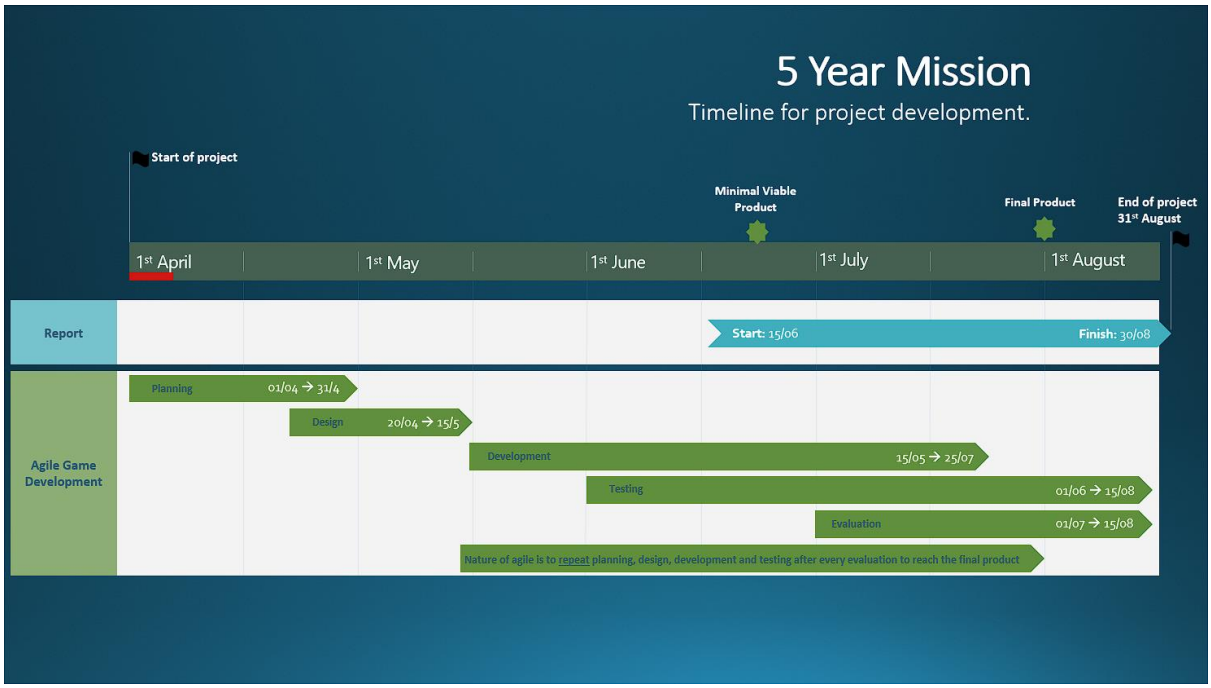


Fig 2.1 Timeline displaying planned project development

Design

The agile design process entails the use of the incremental and iterative methods. Work from the initial iteration is improved upon in following iterations throughout development of this project. This will be reflected by dividing related functionality or mechanics of the game into separate sections for a more organised approach. These sections are worked on in order of dependency, where each completed section would be considered a milestone towards the finished product. To design the 5-year mission game, it will be a type of colony simulator with the main features focused on:

- Decision making
- Problem-solving
- Balance
- Player attachment
- Progression
- Player usability

Development

This is where the implementation of the design documentation occurs, bringing ideas to life through coding. This will be the most time-consuming, where the scope is affected most as some tasks may require more resources than initially thought. Fortunately, the agile methodology aids this with each iteration being based on the progress of the last. This process will be broken down into sections to work on different components, making smaller pictures to build into the big picture of the finished game. The steps and progress of this will be recorded in Gitlab and Jira to ensure accountability and progression on tasks.

Testing

The testing section will objectively evaluate that the product is running as intended and has met all previously mentioned criteria. Testing will also be occurring throughout development to identify any bugs or issues that need to be addressed in another iteration of development. Initially, user test cases will be developed to test every feature implemented, evaluating the level of robustness of the finished product. Following a user's completion of the game, surveys will be distributed be able to assess the game in its final state and receive feedback on the product. Further to be able to evaluate the project's research question, the surveys will also include questions dedicated to assessing how an individual approaches games, and how they view attachment/progression in games.

Evaluation

Once all previous phases are complete, the final stage is to review the results from the project testing stage and evaluate the research question against it. The data received from the player's decisions throughout the game will be checked against the survey data to see if a player's intentions line up with their decisions. Further sections of the survey will also provide insight into which areas the game delivered and which areas fell short. Improvements, changes, and future work will also be further discussed towards addressing any issues with the management of the project.

Design

The design stage of this project can be divided into two parts[10], the design of what the player sees and experiences, and the design of the architecture which dictates what the player sees. The latter is more deeply discussed in the Implementation section of this paper. As mentioned, the design of this game will focus on ensuring:

- Decision making
  - ► Providing scenarios where the player has to make choices that influence the game.
- Problem-solving
  - ► Creating a challenge for the player.
- Balance
  - ► A balance between benefits of attachment vs progression.
- Player attachment
  - ► Features that encourage attachment between the player and game.
- Progression
  - ► A clear way to demonstrate progression through the game.
- Player usability
  - ► How understandable the game is while playing.

These six objectives will set a clear objective through the development of the project, providing scenarios that will aid in gathering research data and also provide an enjoyable experience to play.

Design Brief:

To be able to answer the question of an individual's decision-making process as per the research question, the game would preferably be of the colony simulation genre for reasons previously discussed. As colony simulation games are large-scale games with numerous systems, such as those for colonist control, and high replayability ceilings the created game will need to focus on implementing a balance of features between attachment and progression. Attachment in 5YM will be influenced by the colonists they control, hence interacting with them is essential to create a balance of features. The colonists will need to have systems that allow for the differentiation between colonists, traits, and personalization by the player with color choice and naming. As the title of this project is a homage to Star Trek, the game was created to take place in space with the following setting synopsis:

"You are an overlord and your colony has crashed into a distant asteroid on the way back to their home planet. You must scavenge enough resources to be able to send your colony on a final 5-Year Mission back to their home planet!"

The game will require the player to organize their colony and collect resources to go on missions to nearby areas to be able to progress in the game. To progress the player will need to craft and upgrade their ship to travel to further missions, allowing them to eventually reach the final 5-year mission which is the colony's journey back to their home planet. Each type of mission will have its challenges by giving the player scenarios in which they will have to decide between taking a risk or not, chancing reward or loss. This decision-making process would be recorded. The visuals of 5YM would need to be achievable in the project time frame, hence a low-resolution aesthetic revolving around ASCII to add depth was decided for the game.

Sitemap:

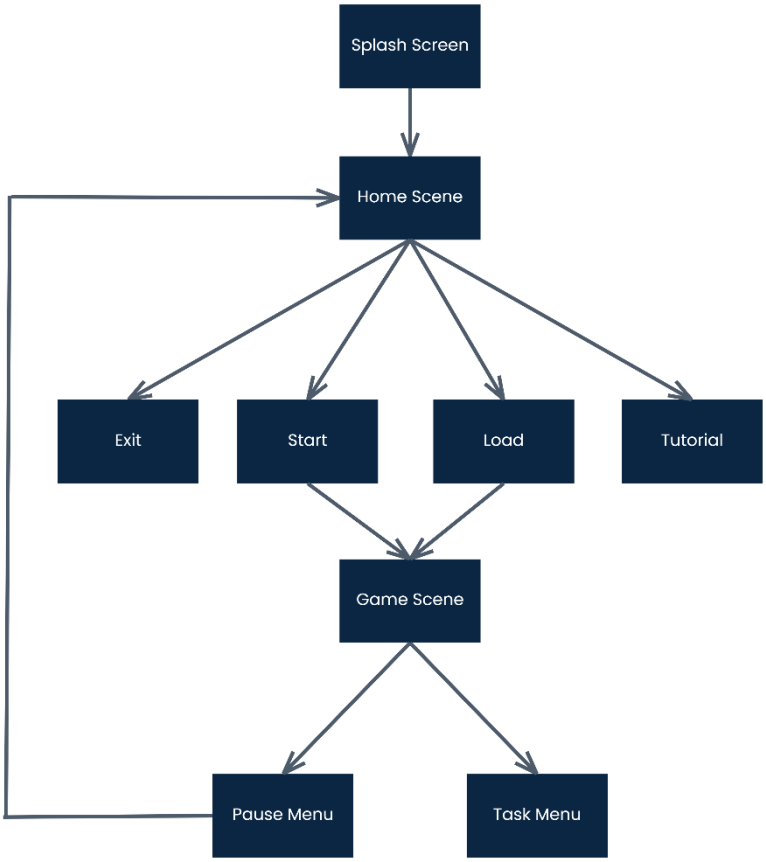


Fig 3.1 Sitemap of 5YM game

- Splash Screen: Unity has a standard splash screen when using its engine.
- Home Screen: This section will display buttons available for the player to navigate to.
- Exit: This button will quit the game.
- Start: This section will open a colonist selection menu to start a new game.
- Load: This section will display previously saved game files to start from.
- Tutorial: This section will display information on how to play the game.
- Game Scene: This scene will be where the player will be able to play and interact.
- Pause Menu: This menu will pause the game scene and offer to save and quit.
- Task Menu: This menu will give the player information, customisation, and tasks for colonists.

Wireframes:

Home Scene:

Start = Button to go to colonist selection.

Load = Button to go to save selection.

Tutorial = Button to go to tutorial page.

Exit = Quit game.

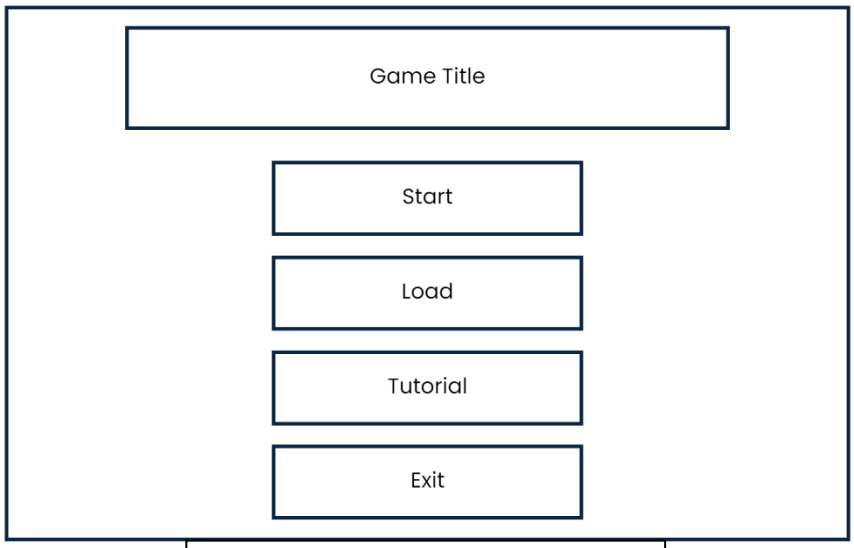


Fig 3.2 Wireframe of Home Scene

Start Menu:

Reroll = Button to randomly chooses trait values for colonists.

Back = Button to return to Home Scene.

Embark = Button to proceed to Game Scene.

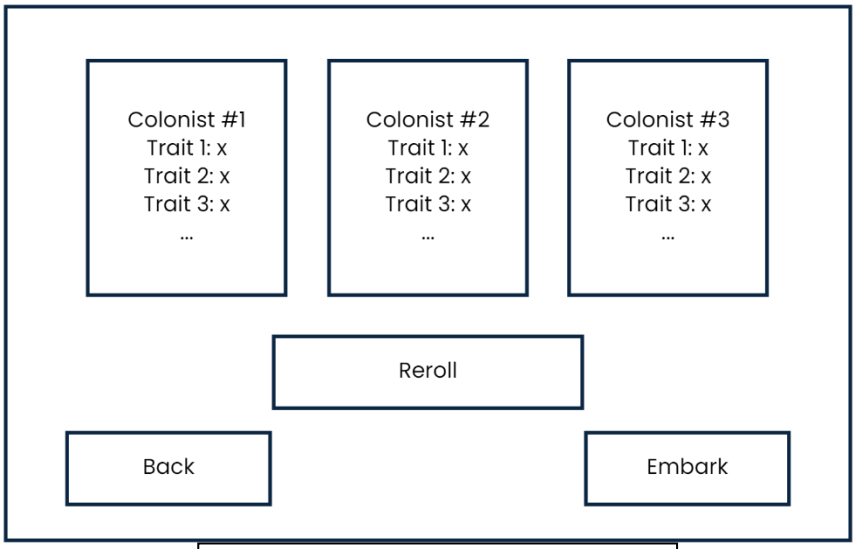


Fig 3.3 Wireframe of Start Menu

Load Menu:

Saves = Dropdown displaying all previous saves to choose from.

Load = Button to proceed to *Game Scene*.

Back = Button to return to Home Scene.

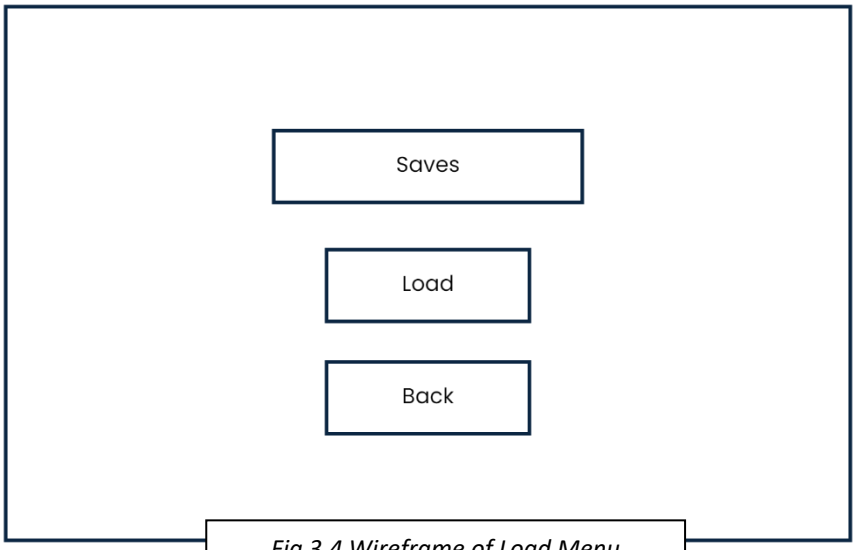


Fig 3.4 Wireframe of Load Menu

Tutorial Menu:

Mechanic Image = Image to be able to relate the intructions.

Mechanic Instructions = Text providing information to the player.

Back = Button to return to Home Scene.

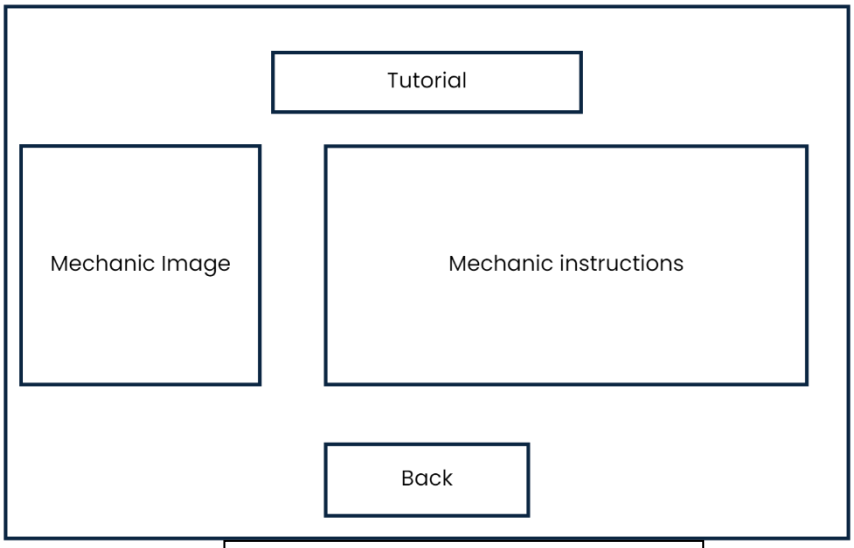


Fig 3.5 Wireframe of Tutorial Menu

Game Scene:

Resources UI = Text displaying how many resources the player has.

Building/Mining UI = Buttons to select type of build.

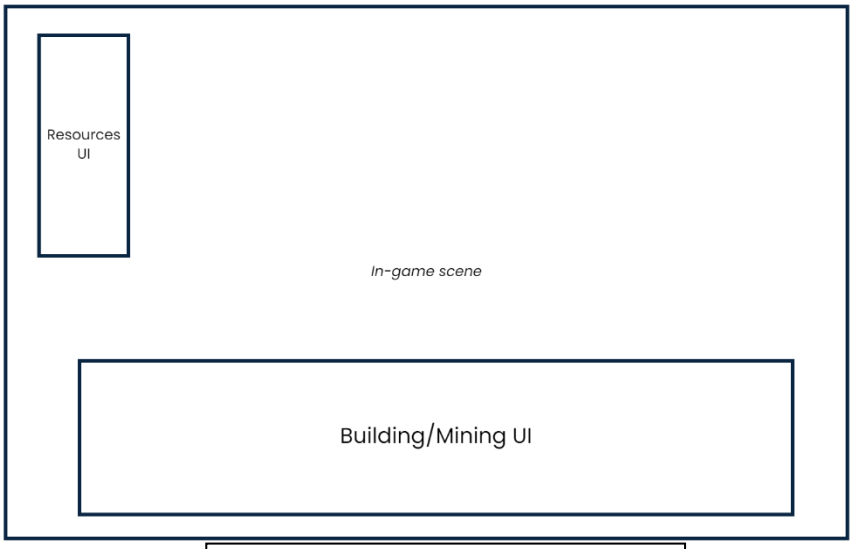


Fig 3.6 Wireframe of Game Scene

Pause Menu:

Informational Prompt = Text to warn for any loss of progress.

Back = Button to return to Game Scene.

Save&Quit = Button to return to Home Scene and also save the game.

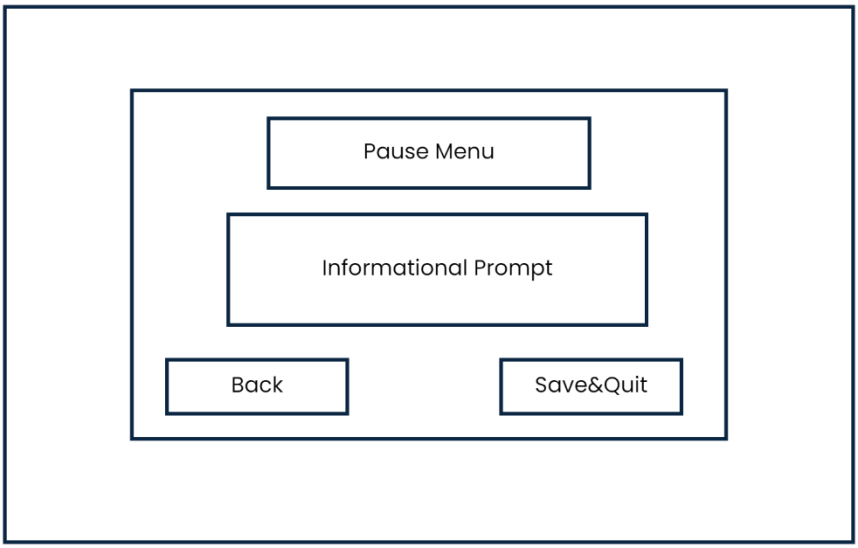


Fig 3.7 Wireframe of Pause Menu

Colonist Menu:

Colonist Display = Image of colonist.

Trait Values = Text displaying the colonists stats.

Colonist Selection = Dropdown to choose which colonist to display.

Name = Input for a colonist's new name.

Color = Dropdown to choose a colonists color.

Save = Button to save colonist changes

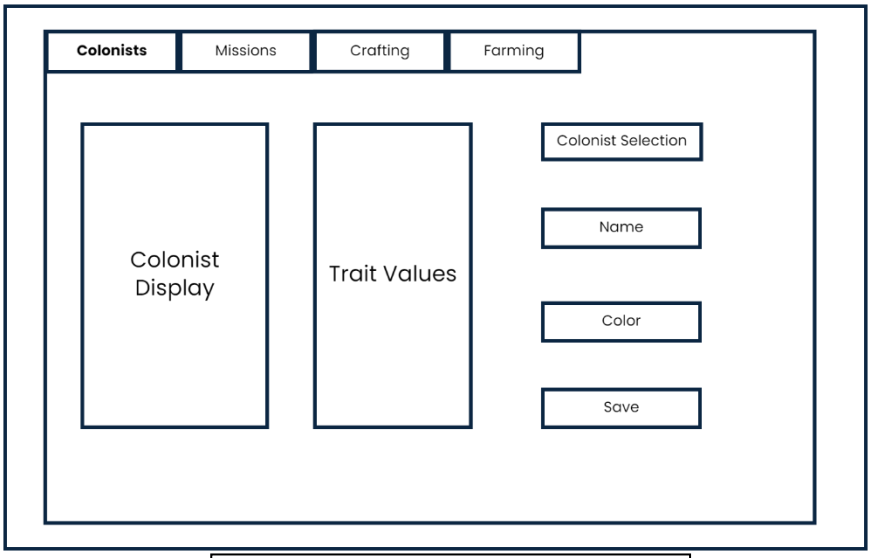


Fig 3.8 Wireframe of Colonist Menu

Missions Menu:

Mission Selection = Buttons to choose which mission.

Colonist Selection = Dropdown to choose which colonist.

Embark = Button to send the chosen colonist on a mission.

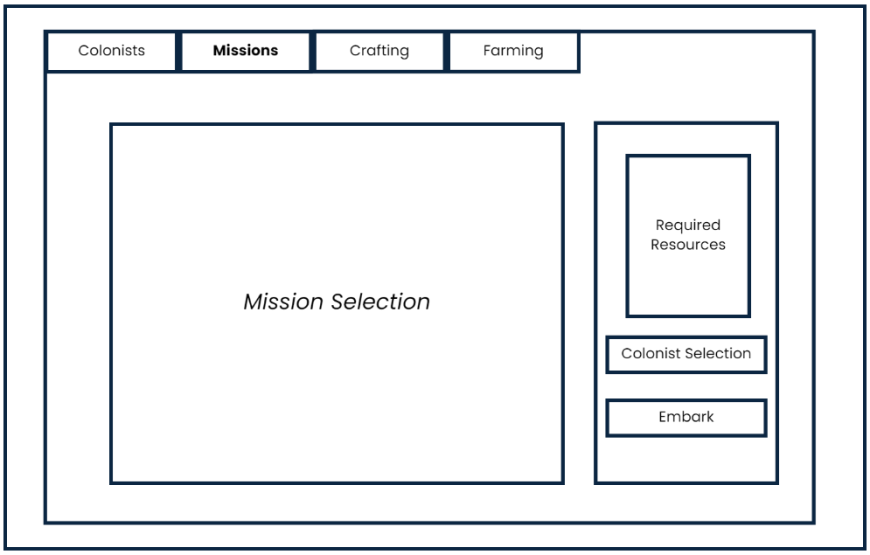


Fig 3.9 Wireframe of Mission Menu

Crafting Menu:

Ship Selection = Buttons to choose which ship to build.

Build = Create the chosen ship.

Fig 3.10 Wireframe of Crafting Menu

Farming Menu:

Farm Information = Text to display how many farms are built and how many are free to do tasks.

Colonist Selection = Dropdown to select a colonist for a farming task.

Set to Farm = Button to set the chosen colonist to start farming.

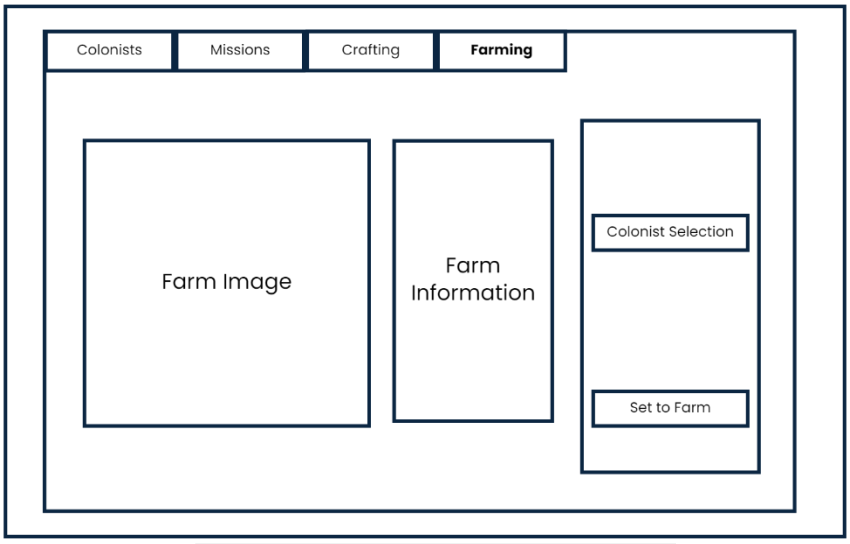


Fig 3.11 Wireframe of Farming Menu

State Machine:

A state machine was created for colonists to handle the interaction between colonists and tasks, allowing for smoother transitions between tasks and simpler checks for if a colonist is free to do a task.

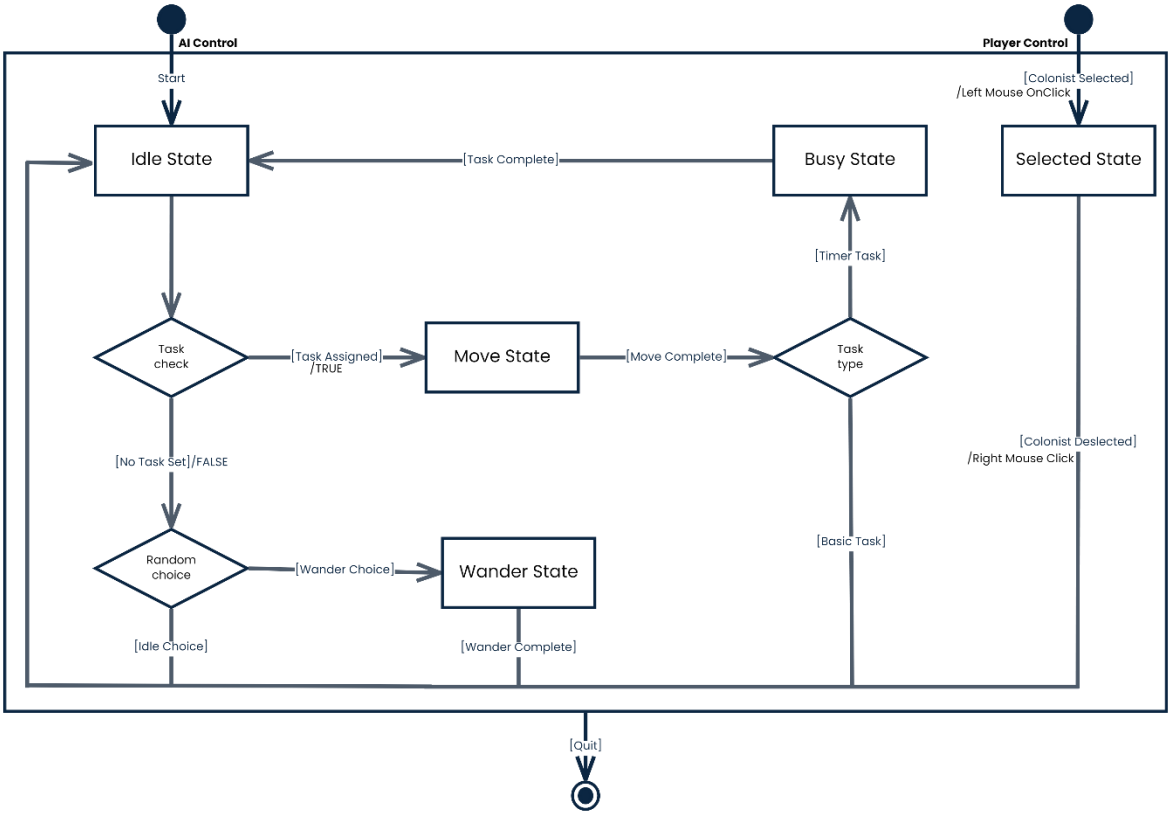


Fig 3.12 Colonist State Machine

Implementation

Map Generation:

A common theme in colony simulation games is to have an element of randomness to make each play through of the game unique, increasing the replayability of the game. To keep this theme constant through the genre and making 5YM more enjoyable, the decision was made that the world generation is randomly generated each time. To achieve this Unity's built in PerlinNoise function which is a "pseudo-random pattern of float values generated across a 2D plane"[11]. Unity's noise technique utilises two dimensions and consists of 'waves' which gradually increase or decrease across the pattern, returning float values between 0.0 and 1.0. With *Fig 4.1* as reference, the map has dark and light areas with a gradual fade between each. The value of coordinates on the map would be determined by how 'light' or 'dark' the location is. These noise values would be stored in a PerlinGrid to be checked against a constant value dictating the fill of the map, where if the noise value is higher or lower than the constant a 1 or 0 would be respectively set in a separate MapGrid. Following this a Moore neighbourhood function was utilised to smooth the map, this is where a square-shaped neighbourhood is used to define a set of cells surrounding a given cell (x,y) affection the evolution of a 2D square grid[12]. The created implementation would take a MapGrid cell and return the number of neighbouring walls, where depending on the returned value the given cell value would become a floor value (0) or wall value (1).

Fig 4.1 Perlin Noise Example

The MapGrid would now go through another 4 iterations of Perlin noise and Moore's neighbours with varying constants to be able to create a final iteration of the MapGrid, with values 0 to 4 representing different TileBase objects to be Rendered into the Tilemap, serving as the base map.

- MapGeneration.cs

To be able to create the topography of the map, which would be the layer that the player can interact with, a comparative system was utilising the MapGrid created by the MapGeneration.cs. The system would iterate through the MapGrid checking the cells values for which type of Tile to place, followed by comparing the neighbours of the current cell to return a switch value indicating which variation of the type of the tile. Implementation of this custom topography system proved to be lengthy but would provide a more visually enjoyable experience for the player with the environment reacting to changes the players would make.

- TopographyGeneration.cs

Player Usability:

The product would need to have an enjoyable and frustration-free experience for the player. To achieve this the player would need a tutorial for the game as displayed in the previous wireframes. Due to the scope of the project being large instead of creating an information panel to display game information, an instruction document was created as seen in *Appendix 1*. This proved to be a more beneficial approach as it allowed for extra time for more systematic issues to be resolved, while also providing a friendlier way for the player to be able to refer to information while playing instead of having to save and quit while in-game to access a tutorial menu.

Fig 4.2 Varying Task Menu Panels

Further, the user interface would need to be intuitive and simplistic to not overwhelm players who are not as used to the genre of colony simulation games, which typically offer a multitude of menus for game control. Having a single menu that holds multiple tabs allows for keeping all the player's control in one location, so as to not confuse or clutter the game space with multiple menus. The TaskMenu.cs would hold 4 different game objects that are activated and deactivated according to which tab the player selects to be open.

- TaskMenu.cs

Problem Solving:

While the game itself is colony simulation, 5YM also leans into elements of survival where the player would need to be conscious of their resources to be able to complete the game. The game would have a slight learning curve in understanding what is rewarded from different missions, but it is up to the player on how they spend their resources amidst their limited resource problem. A mining system was implemented creating an interactive environment for the player and also providing a limited number of resources, which the player will become more conscious of as the game goes on. The resources will initially be needed to build stations using the placement system to be able to progress in the game doing tasks. While placement or mining mode is active, the EditHandler.cs tracks the player's mouse location, displaying a green/red overlay tile to indicate if the action is valid or not (valid actions depend on the mode).

The PlayerTaskHandler.cs would refer to the EditHandler.cs Boolean when the player clicks to know if the player's request should be processed as a task. The PlayerTaskHandler handles all the interaction the player has with the game with tasks set being saved in a TupleList<Int, Vector3> known as the TaskList, containing the task type and the task location. The task type would be a number that other classes, like the BusySate.sc, would use to know what action to do. The PlayerTaskHandler.cs instantiates the ColonistTaskHandler.cs to assign tasks when TaskList is not empty. The ColonistTaskHandler.cs receives the TaskLists tuples to be added to a queue if no colonist is in the Idle/Wander state and free to do the task. If a colonist is free, it checks the task cell locations direct neighbours if they are open and chooses the shortest path, this avoids the issue of walking on the location of the task and getting stuck under a tile.

- EditHandler.cs
- PlayerTaskHandler.cs
- ColonistTaskHandler.cs

Decision Making:

This section of implementation handles the process of collecting data for the research question. When colonists are sent on missions their BusyState.cs creates TaskTimer object and once the timer is complete a UI pop-up displays the player a scenario. The player must now decide whether to take a risk and allow their colonist to proceed or return back and not risk their colonist. The scenario would be chosen from a bank randomly, with each mission location having a different bank of scenarios, sometimes considering colonists' trait values increasing the chances of a successful mission. Unsuccessful missions have varying consequences such as losing trait values or the colonist dying. These scenarios would mainly be based around chancing the colonist on the mission for a chance at progression/rewards in the game. The player's decision would then be added to a list which is then saved for later analysis when the game is saved. The research list would contain the values -1, 0, 1 and 2 indicating a risk decision, neutral decision, attachment decision and the decision to save & quit the game respectively. Recording when the player leaves the game further allows tracking to see if the player is returning to older saves because of regretting a decision they had previously made.

- BusyState.cs

When the save and quit button is clicked, the SaveSystem.cs creates a SaveObject class to hold all the important data needed to re-instantiate the player's progress in a new game session. The Save() method would proceed to collect all the relevant data from various classes, such as colonist information, and then set the relevant data in the SaveObject. The SaveObject would then be converted to a string using JsonUtility.ToJson and then sent to the FileHandler.cs class which would handle instantiating the save folder location and writing the file. Returning to a save would be handled by the LoadHandler.cs when called by the SaveSystem.cs, converting the file back to a Json SaveObject and accessing the saved information for the LoadHandler to restore to other classes and recreated the colonist game objects.

- SaveSystem.cs
- FileHandler.cs
- LoadHandler.cs

Progression:

5YM would need a clear end goal and markers for progression through the game, hence systems were created to handle tasks. All tasks require time to pass before calling and action while checks are being done for interruption. The TaskTimer.cs would receive an Action, for what method to call, a float for the time, a string for reference to the timer and created would create an object with a TaskTimerBehaviour attached to it allowing the class to check for active timers. Once the timer expired the created timer object would call the Action followed by destroying itself to not use unmercenary memory. To be able to handle tasks which were interrupted the colonist's State Machine was utilised, where the method to interrupt a colonist from doing a task would be selecting the colonist, becoming the SelectedState.cs in the SateManager.cs. With this, if the colonist's state is BustSate.cs and switches to the SelectedState.cs, a taskComplete Boolean is checked for true which only occurs when the TaskTimer object is able to call the chosen Action at the end of the timer. If taskComplete is false, the TaskTimer.StopTimer() method is called for the timer specific to the colonist, destroying the game object before it can call the Action.

- TaskTimer.cs
- BusyState.cs

Progression through the game would depend on going on Mission tasks to specific locations where the player may be rewarded with the required resources for crafting the next ship. Higher tiers in ships allow for travelling to more mission locations. Once the final ship is crafted, the player can embark on the final 5-Year Mission to complete the game. This system was implemented as it would not overcomplicate the game for users while also ensuring the player would have to make more decisions to progress in the game, leading to more data collection for the research question.

Player Attachment:

The game would need to handle the pathfinding for the colonists to move to task locations. To achieve this a grid was created with each cell representing a PathNodes.cs object in the abstract GridFrame.cs. The PathNode would store information for an A* Search algorithm such as the previous node, the fCost, gCost, and hCost allowing the Pathfinding.cs to find the optimal path. The A* search algorithm is based on Dijkstra's shorted path algorithm allowing for finding the shortest path between two nodes[13]. The PathFinding.cs would take a starting and end Vector, converted into grid locations, iterating through each neighbouring tile traveling through the lowest cost until finding the optimal path as a list of Vectors for the colonist to move through.

- GridFrame.cs
- PathNode.cs
- Pathfinding.cs

Finally, customisability was added to colonists to encourage attachment, allowing the player to be able to change the colonist's name and color through the Task Menu colonist tab. As colonists are necessary for the player to progress through the game by doing tasks, the player will have to interact with the colonists, aiming to be what create an attachment between the player and the game further allowing the player to add a personal attachment of their own to the colonists through customisation features.

Balance:

A game requires balance to have a fair experience and allow for a steady progression through the game, without balance a game may prove to be too easy or too hard at varying times. Though these concepts of fairness are important, they are not considered fundamental concepts for the definition of game balancing[14], hence another balance to consider is between attachment and progression. While the end goal of a game may be obvious, if there is no benefit or systems in place in encouraging attachment it would be easily looked over. To avoid this, 5YM created a trait system for colonists where the colonist's traits would influence the amount of time a task would take respective to the corresponding trait value, such as a colonist's 'intelligence' trait dictating the time taken to complete a Mission task. These values would be stored in a List<Int> in the ColonistGridMovement.cs to be accessed by the BusyState.cs. The higher the trait value the faster a task would be done persuading the player to favour some colonists over others, and through this preference, a system for attachment can be created. As colonists complete tasks successfully, the BusyState.cs checks the colonist's progression for a trait, increasing the value of the trait once they reach a certain threshold. This further creates a sense of time invested in individual colonists, differentiating them further from other colonists creating more depth to the attachment architecture of 5YM.

- ColonistGridMovement.cs
- BusyState.cs

Testing

Testing was dedicated to measuring the robustness of the game, collecting research data, and receiving feedback from alpha-testers.

Test Cases

Test #	Test Case Description	Test Data	Expectation	Result	Pass/Fail
1	Menu Scene buttons open correct menu buttons	Clicking the Start, Load and Exit buttons	Start = Colonist Selection Menu Load = File Selection Menu Exit = Closes application	As Expected	Pass
2	Reroll button in Colonist Selection Menu changes trait values	Clicking the reroll button	All trait label values randomly change value	As Expected	Pass
3	Starting a new game correctly assigns user inputted colonist information from Colonist Selection Menu	Embark button clicked with user inputted data: Colonist 1 =“test” + Green Colonist 2 =“test2” + Yellow Colonist 3 =“test3” + Red	Game Scene loads displaying 3 colonists coloured green, yellow and red. Opening the Task Menu displays the correct names in colonist drop downs	As Expected	Pass
4	The game correctly saves games and successfully loads previous saves	1)Creating a new game 2) Build 4 wall tiles 3) Save and Quit 4)Click the Load button from the Menu Scene 5)Select “Save_1.txt” 6)Click Depart Button	The game contains colonists with the same trait, name & color values. The map will contain 4 wall tiles in the same positions built previously.	As Expected	Pass
5	Pause Menu is able to freeze actions	While in-game: 1)Set 3 build tasks in-game 2)Pause the game with [ESC] key while colonists are busy performing tasks	All tasks will cease to progress and the pause menu will display which colonists are currently busy with tasks	As Expected	Pass
6	The player is able to control the game camera	[W], [A], [S], [D] keys pressed down at various times	The view the player sees shifts according to key press	As Expected	Pass

7	Mining mode works as intended	1) [M] key pressed to activate 2)Mouse click on surrounding reachable topography 3)[M] key pressed to deactivate	While active the players mouse position will display a color to indicate a valid or invalid action. When a valid action is clicked a colonist mines the area	As Expected	Pass
8	Placement mode works as intended	1) [SPACE] key pressed to activate 2)Mouse click on valid build location 4)[SPACE] key pressed to deactivate	While active a build UI will be available to the player to choose which station to build. When a valid action is clicked a colonist builds the correct station	As Expected	Pass
9	Resources update correctly	5 variations tasks done: 1)Building a wall 2)Mining a stone wall 3)Crafting a ship 4)Sending a colonist on a mission task 5)Assigning a colonist to farm 6)Attempt to build a ship yard with inadequate resources	1)Adequate resources = corresponding resource will be reduced by the indicated amount 2)When mining the player will gain resources. 3/4)Crafting and Mission tasks will gain and lose various resources 5)Corresponding resource gained 6)Inadequate resources = nothing will occur	As Expected	Pass
10	Select mode works as intended	While in game: 1)A chosen colonist is left-clicked 2)Left-click various open locations. 3)Right-click	1)The colonist changes color to while and is now selected 2)The colonist will move to the clicked location 3)Colonist deselected	As Expected	Pass
11	Task Queue stores unassigned tasks	Building 10 wall tiles with only 3 colonists	All 3 colonists will proceed build a wall and once their assigned task	As Expected	Pass

			is complete, one of the remaining 7 build tasks is assigned till the wall is complete		
12	Tasks are successfully interrupted	While a colonist is busy building the assigned HUB tile, left-click the colonist	A colonist changes color to grey when busy doing a task. Once left-clicked the colonist will become white and the HUB building will not be built and resources will not be affected	As Expected	Pass
13	Task Menu works as intended	1)Colonist Tab: name =“spock” + Yellow 2)Mission Tab: Colonist select = “spock” 3)Crafting Tab: Ship = “Cargo Ship” 4)Farming Tab: Colonist select = “test2”	1)When the save button is clicked, the selected colonist will have its name and color set as per the inputs 2)When the embark button is clicked, Spock would proceed to the HUB building and disappear 3)When the craft button is clicked, resources would be adjusted accordingly 4)When the assign button is clicked, Test2 would proceed to a farm and become busy	As Expected	Pass
14	Decision UI pop-up works as intended	Sending colonists on multiple missions	Missions with no decision will have a neural UI pop-up where the player has only been rewarded with resources. Missions which require a decision display Avoid and Risk buttons	As Expected	Pass
15	Colonists correctly are removed if a	Test3 (colonist), Embarked on a	The pop-up scenario will	As Expected	Pass

	mission scenario results in death	Shipwreck mission with the scenario: "Sacrifice your colonist to save an unknown number of colonists" from a Ship Wreck Mission. Risk button clicked.	display the outcome of the scenario, gaining either 1 or 2 random colonists returning back from the mission, while Test3 no longer exists		
16	Topography tiles update according to player interactions correctly	Mining a 5x5 tunnel inside a topography formation	The tile visuals around the mined tile will change according to its updated environment	As Expected	Pass
17	Colonist Pathfinding is able to decide if the path is possible, and correctly take the shortest route	25 mining and 25 building tasks were assigned at varying locations.	The colonists are able to handle most of the tasks but are unable to react to environmental changes while approaching a task location. Further, the pathfinding would not categorize two diagonal neighbours as impassable causing colonists to get stuck in their moving state.	As Expected	Fail
18	5YM ends as intended	Embark button on the final mission, "5-Year Mission" from the Missions Tab.	Pop-up indicating the game is over, allowing the player to Save&Quit to the Menu Scene.	As Expected	Pass

<u>Research Data</u>

Seven individuals participated in the alpha testing of the game. They were provided with a zip containing instructions and the 5YM executable game. During a play-through, the individuals' decisions are recorded with distinct values allowing for future analysis. These values would indicate the type of decisions a player made during their playtime and would be stored on the game's save file to be restored in future play sessions. The values ranged from -1 – 2 and represented:

- -1 → The player took a risk
- 0 → When the scenario is neutral

- 1 → The player avoided the risk
- 2 → The player saves & quits the game, indicating the end of a play session

This data was part of a SaveObject which is exported to a text file, where a new save file is made for each play session. The data was sifted through to construct a table containing statistics on each individual overall decision-making experience, reducing the multiple files of data into one location for simplicity and to more understandably identify patterns. The original data can be found here:

- Github Participant Data

Participant #	Complete game?	Save File count	Session count	Risk Decision count	Attachment Decision count	Neutral Decision count	Total count
1	Yes	26	26	30	8	76	140
2	Yes	10	10	34	5	17	66
3	No	4	3	4	1	12	20
4	No	3	1	0	0	0	1
5	No	5	3	3	1	4	11
6	No	9	7	1	0	13	21
7	Yes	22	22	21	1	217	261

Participant Feedback

Once the participants completed the play testing period they were handed out a survey to complete. The intention behind this is to collect information on the individual's experience with attachment in games generally, followed by questions directly related to 5YM allowing the participants to provide feedback and comment on their experiences. The participant's responses are linked with their gameplay data allowing for cross-analysis picking up on any connections if necessary. Below the statistical responses from individuals are displayed, refer to *Appendix 2* for the survey's longer written responses such as feedback.

Approach to Games Section:

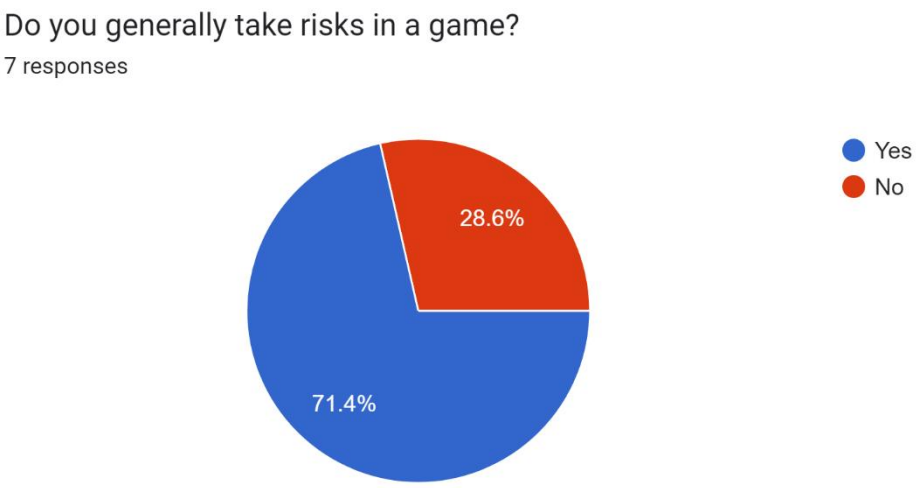


Fig 5.1

What is your typical approach in a game? Risk means you are willing to take chances for faster/further progression, or even rewards. Prepar... you do not lose progression, or things you value.

7 responses

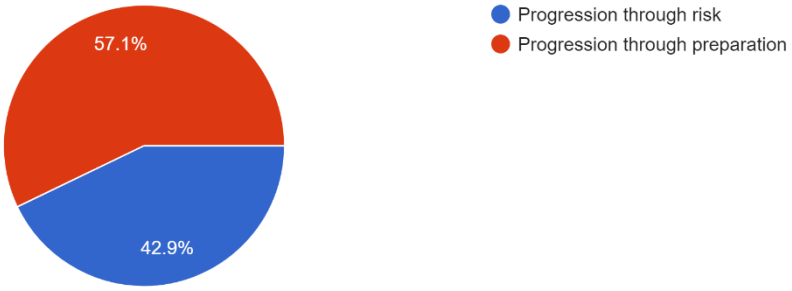


Fig 5.2

Select the the types of games you feel create attachment between the player and in-game characters.

7 responses

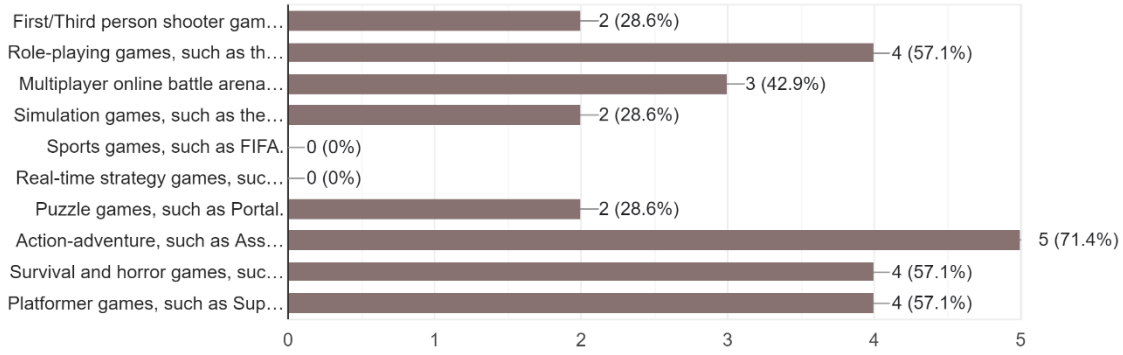


Fig 5.3

Is taking risk necessary in a game you are new at to be able to get a better understanding of the game?

7 responses

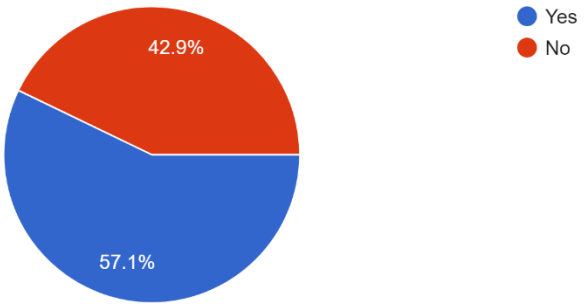


Fig 5.4

Alpha Testing Section:

Fig 5.5

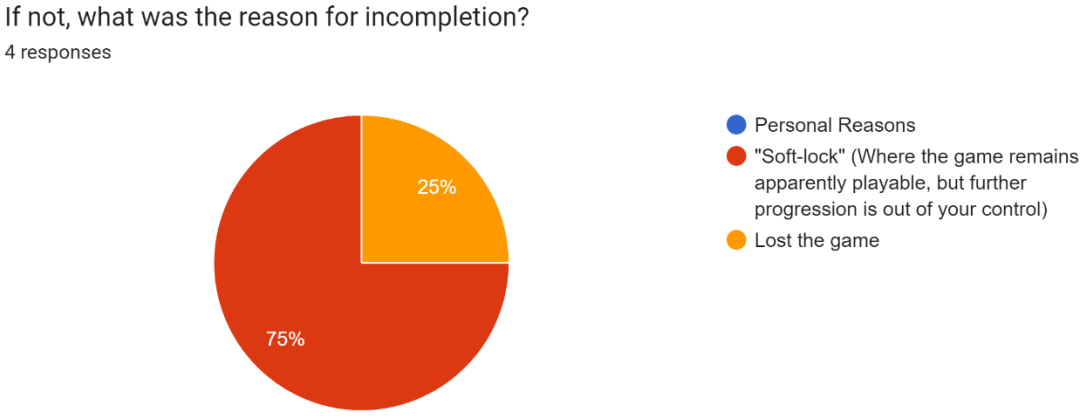


Fig 5.6

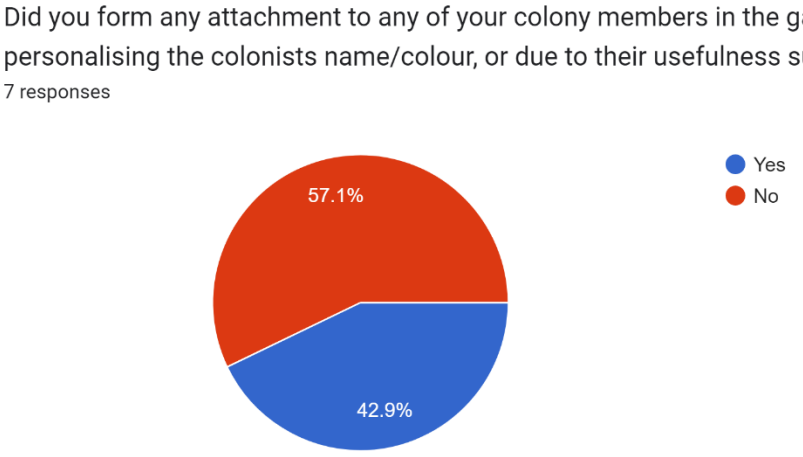


Fig 5.7

If there was more personalisation features for the colonists, would that create a deeper level of attachment?

7 responses

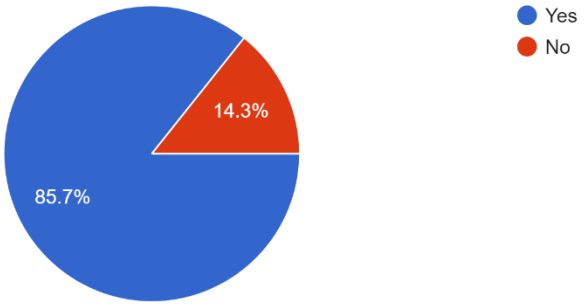


Fig 5.8

What do you think you prioritised more during your play through?

7 responses

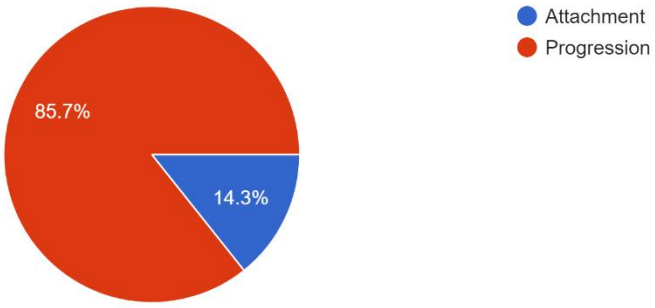


Fig 5.9

How difficult did you find the game?

7 responses

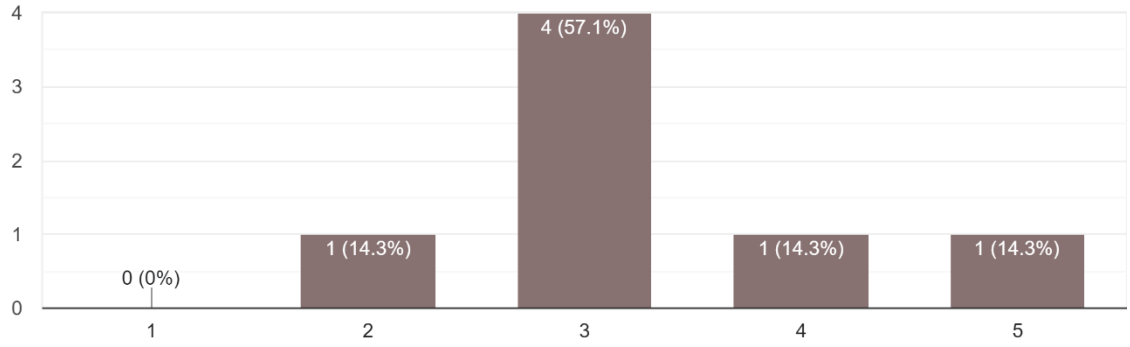


Fig 5.10 Difficulty where 1 = Easy & 5 = Hard

Regarding missions in the game, would knowing the exact probability of successfully completing a mission change the decision you make in similar scenarios without knowing the probability?

7 responses

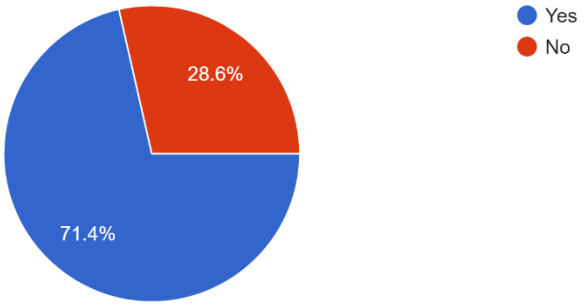


Fig 5.11

How did you find the aesthetic of the game?

7 responses

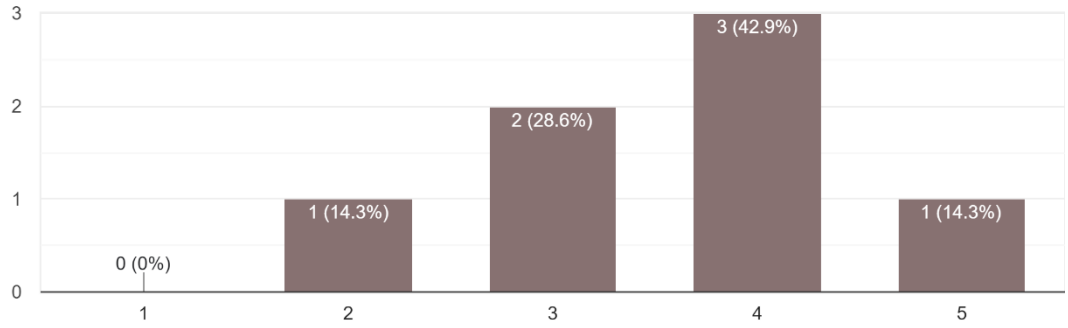


Fig 5.12 Aesthetic where 1 = Bad & 5 = Great

How easy was the game to navigate in terms of controls and UI (user interfaces)?

7 responses

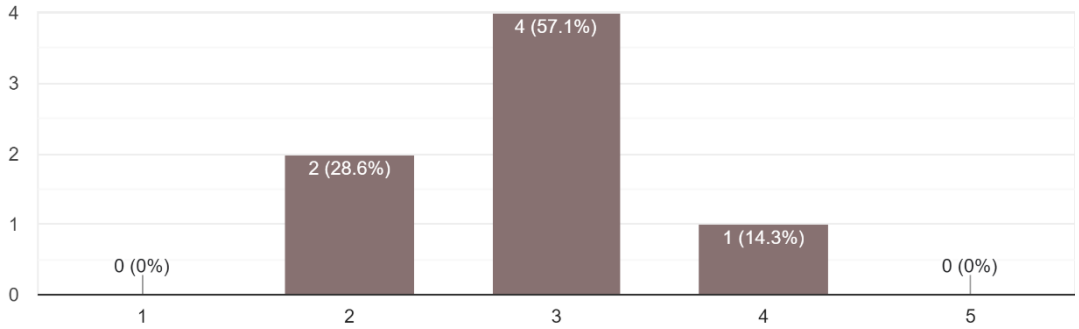


Fig 5.13 where 1 = Bad & 5 = Great

Project Management

Gitlab

Gitlab was utilized as a housing service to hold the project's latest stable code, allowing for access from any location and having a backup in case any restoration is required locally. The files on Gitlab would be updated whenever progress towards a task was completed, or if any changes were made that had yet to be pushed at the end of a session. There is a total of 26 files:

Script name:	Script function:
State.cs	This script is an abstract class for the colonist state machine
IdleState.cs	This script is one of the States in the state machine & is the default state for all colonists. This state would determine which state to change to according to Booleans.
WanderState.cs	This script is one of the States in the state machine & is randomly chosen to switch to by the *IdleState*. This state would return to *IdleState* according to a Boolean for checking if the movement is complete.
SelectedState.cs	This script is one of the States in the state machine & is forcefully switched to when a colonist is left-mouse clicked. This state indicates that the player is directly controlling the movement of this colonist, interrupting any task the colonist was doing. The colonist returns to *IdleState* once the player right-mouse clicks.
BusyState.cs	This script is one of the States in the state machine & is transitioned through the *MoveState*. This state assigns the colonist different tasks according to an index value governed by the *PlayerTaskHandler*. Mission tasks in particular have their probability and scenarios calculated here to be presented to the player. Further, rewards or losses from tasks are calculated here to handle the player's resources.
MoveState.cs	This script is one of the States in the state machine & is transitioned through the *IdleState.cs*. This state acts according to a Boolean for checking if the movement is complete and returns *BusyState* if true.
Pathfinding.cs	This class takes 2 positions (startPos & endPos) and finds a valid path by using an A* search. The colonist receives a list of *PathNodes* to follow until they reach the final node in the list.
PathNode.cs	This class is used for each object in the grid *Pathfinding* uses, holding all the information for that specific object in the grid such as if it is walkable or not.
GridFrame.cs	This class creates the grid that the *Pathfinding* class uses as the locations for the Pathfidning nodes.
MapGeneration.cs	This class uses Perlin noise & Moore's Neighbours to be able to generate a base map for the player to play in. Initially, the script creates a grid and assigns values between 0-4 (different TileBase) followed by rendering the Tiles according to the grid values. Generation is hash-seeded.
TopographyGeneration.cs	This class works off the grid created in *MapGeneration* and handles the generation and removal of tiles in the interactable layer of the game. This class handles a large

	number of cases to dictate which tile and at what orientation it is placed, allowing for the environment to visually update according to changes the player makes to the topography.
ColonistTaskHandler.cs	This class is instantiated by the *PlayerTaskHandler* to check if there is a path available for the colonist to approach the location of the task, choosing the shortest path. The class handles both standard tasks which do not need to have a specific colonist and Mission tasks which are only set when the chosen colonist is not in their *BusyState*.
ColonistGridMovement.cs	This class receives the path that the *Pathfinding* found and sets the colonists Rigidbody to follow the list of vectors. It handles the animation of the colonist and holds its trait values. This class further controls Booleans from the *MoveState*, *WanderState,* and *SelectedState* to indicate that they can transition states once again.
StateManager.cs	This class runs the state machine for each individual colonist and is able to track the colonist state as it transitions. It holds the colonist information (name & color) and is able to check if the colonist has been clicked for the *SelectedState* to be transitioned to.
TaskTimer.cs	This class is utilised by the *BusyState* for setting tasks. As tasks take time, this class would create a Timer game object that would do an action at the end of the timer (such as reward the player resources) and then destroy itself. Further the timer would be able to check if it was interrupted allowing for cancelation of tasks.
TaskMenu.cs	This class would handle the main UI of the game, dictating which game object would be visible at a given time. Further the class would handle any colonist information changes made by the player.
PauseMenu.cs	This class would pause the game by scaling time down to 0. Further, the class would check if there are any colonists busy at the moment of pausing the game, informing the player before they leave the game as live tasks are not able to be saved.
MainMenu.cs	This class would handle all of the UI in the first menu scene of the game. Button actions are also handled here with static information being saved for the game in the second scene to dictate whether it is a new game or not.
CameraController.cs	This class handles the players movement of the view as the “overlord.”
PlayerTaskHandler.cs	This class handles almost all interactions the player has with the game. The players total resources are held here, Booleans dictating whether the player is in building mode or mining mode, if there is a colonist in selected mode, what task the player has set & what locations to send colonists to for certain tasks. Lastly, the placement mode UI is also updated here.
SaveSystem.cs	This class handles creating the SaveObject to be able to save the important information that the player needs between saves. Further the class gets all the required information to

	set to the variables in the SaveObject, able to unpack the SaveObject with help from the LoadHandler when loading into a game with a save.
SpawnSystem.cs	This class is not utilised in the final version of the game, but was designed to find pockets to spawn colonists in open random locations.
LoadHandler.cs	This class handles spawning in new colonists and loading in colonists from a save, by first creating them followed by finding a colonist with no information and giving it information provided by the *SaveSystem*. Further the class sets the values of variables that would otherwise be treated as new, such as the topography, map generation and total resources.
EditHandler.cs	This class handles displaying an overlay for the player to indicate if the task is allowed or not (green or red) during edit mode or mining mode. Further sets a Boolean to indicate to the *PlayerTaskHandler* whether the task is legal or not.
FileHandler.cs	This class handles creating the save location, then getting the list of files in the Save folder, followed by loading the correct save according to its name.
MainHandler.cs	This class handles the pre-initialisation of the second scene (Awake method) by checking the static values in the *MainMenu* avoiding any Null errors with classes that rely on others. With the main value being a newGame Boolean, dictating the order of classes to be loaded or initialised (new game or continuation).

- GitHub README.md

Jira

Jira was used in a scrum style alongside the Agile methodology. Sprints were created with clear goals after regular meetings with an academic supervisor to ensure progression. Burnup charts were used to display the sprint progress as you are able to accurately see changes in tasks and their respective story points. During this project timeline, there was periods of drought in terms of progress due to an overlapping with the exam period. This may have been more of issue in other methodologies, but as Agile works off the progress of the previous stage, allowing for full use of Jira's backlog to store tasks that are incomplete, adapting the timeline of sprints as the project progresses.

Fig 6.1 Burnup chart Key

- Sprint 1:

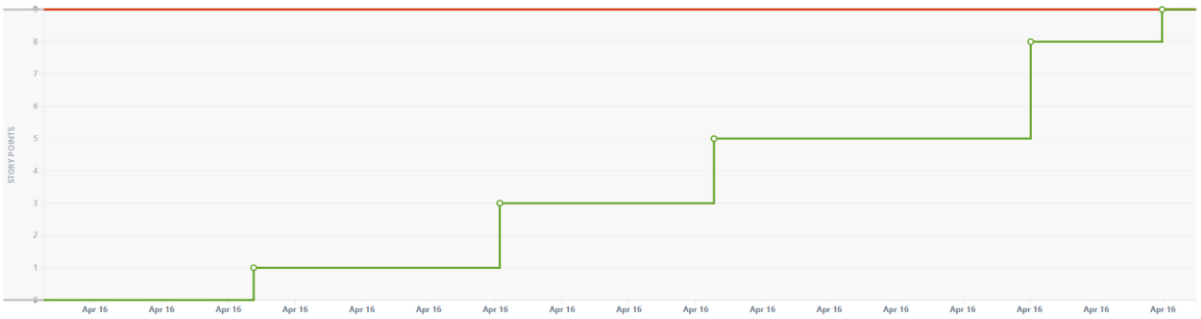


Fig 6.2 Sprint 1 Burnup chart

Date: 02/04/22 – 16/04/22

The initial sprint was dedicated towards setting up the foundations required for this 2D colony simulation game. The game would require visuals of tiles for the map generation and visuals of colonists for the player to control. This layer would utilise Perlin noise to introduce variation to create the foundation layer of the map off which the gameplay occurs.

Story Points: 9	Tasks: 5

- Sprint 2:

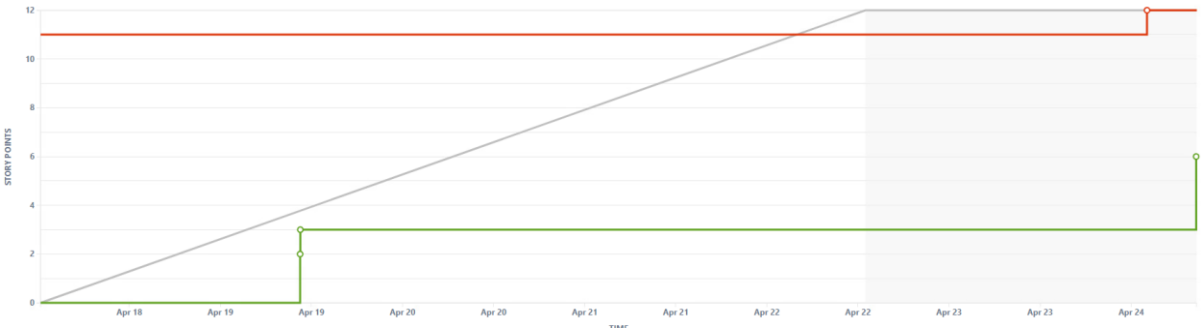


Fig 6.3 Sprint 2 Burnup chart

Date: 18/04/22 – 24/04/22

Due to this sprint overlapping with an exam period, development of the project slowed down to be able to manage more pressing deadlines causing the durations of sprints to be more erratic. The goals for this sprint centred around the topography layer which the player would be able to interact with. To achieve this the visuals of tiles for this layer's tile map would need to be created while also making the layer compatible with the foundation layer. Further for intractability with the topography basic

colony member AI would need implementation. To manage workload during the overlapping period, this sprint was split into multiple sprints with incomplete tasks moving to the backlog to be completed when there is time between exams. This part-sprint in particular was able to achieve creating visuals and generating the interactable layer on top of the foundation layer.

Story Points: 4/12	Tasks: 3/6

► Sprint 3:

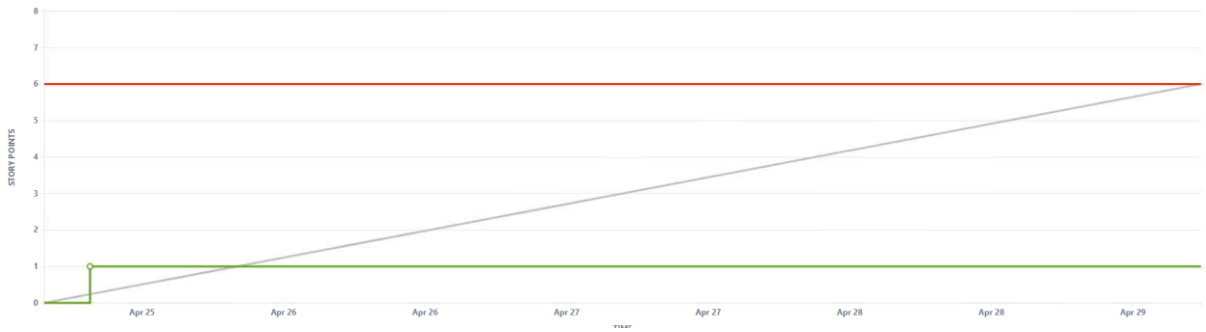


Fig 6.4 Sprint 3 Burnup chart

Date: 25/04/22 – 29/04/22

The sprint was still under the same conditions as the previous sprint. This sprint was able to achieve implementing a method for colonists to wander.

Story Points: 1/6	Tasks: 1/3

► Sprint 4:

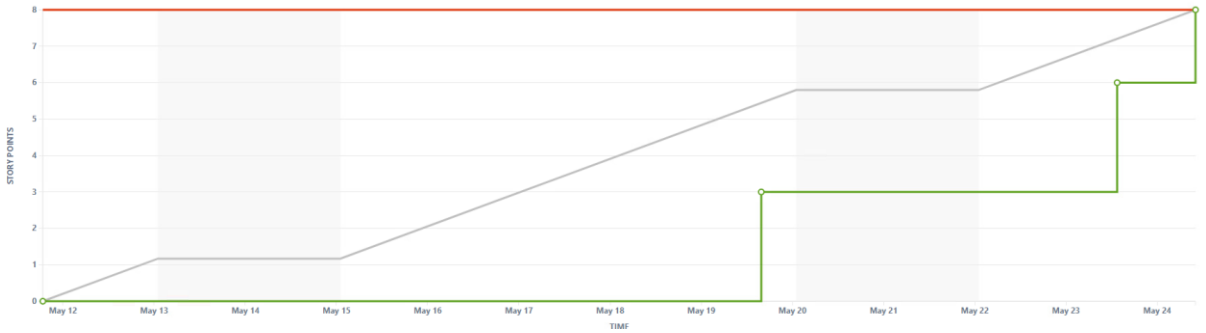


Fig 6.5 Sprint 4 Burnup chart

Date: 12/05/22 – 25/05/22

This sprint picked up after the overlapping period, picking up all incomplete tasks from the backlog from previous two sprints regarding colonist movement, while also adding a state machine for the colonist. The sprint was able to pick up the same momentum as in the initial sprint in this project. Pathfinding became a challenge but was implemented successfully using A*. This will allow for colonist pathfinding to account for the topography of the map. Foundations for colony management.

Story Points: 8	Tasks: 3

► Sprint 5:

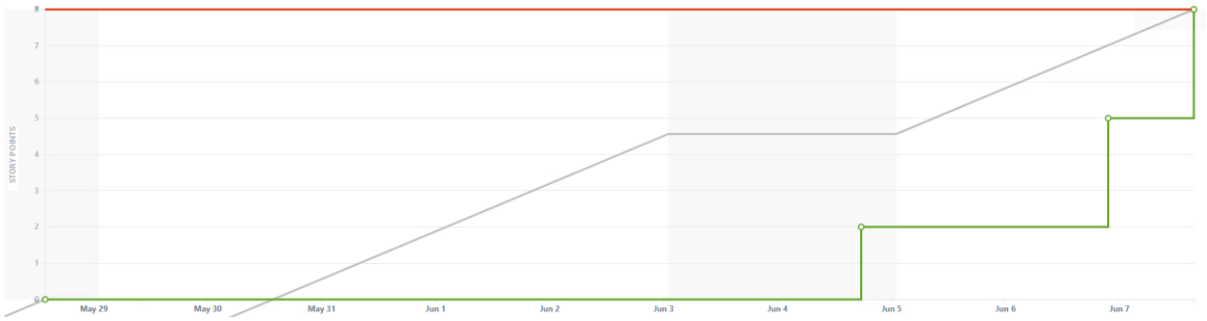


Fig 6.6 Sprint 5 Burnup chart

Date: 29/05/22 – 08/06/22

This sprint furthered the basic blocks of colony management by concentrating on interacting with the second layer, to be able to set tasks, and also create a framework for queuing tasks for individual colonists. Colonists individuality was also ensured with being able to individually select a chosen colonist.

Story Points: 8	Tasks: 3

► Sprint 6:

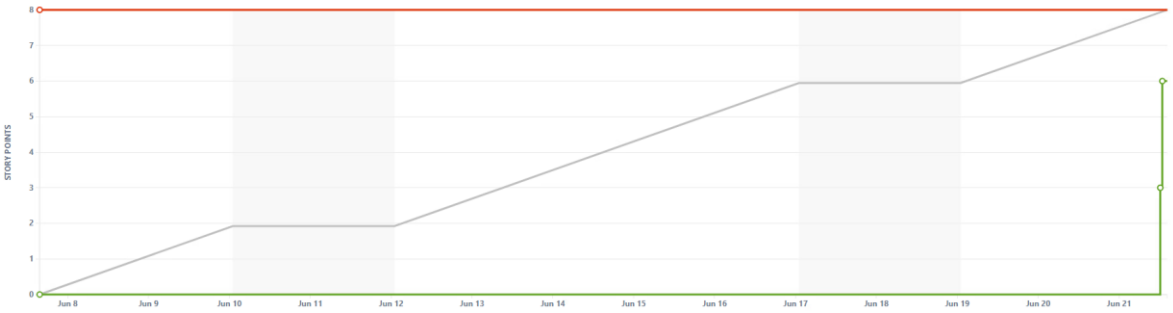


Fig 6.7 Sprint 6 Burnup chart

Date: 08/06/22 – 22/06/22

This sprint was not actively updated on Jira due to technical issues with the connection to the Virtual Lab. Time was dedicated towards the view of the player, adding controls and movement to the camera with panning limits. Further the topography tile map generation was further worked upon to add more cases for placing different tiles according to a tile's open/closed neighbours. The camera system implemented took longer than expected and a task focused on creating new colonists' states was moved to the backlog.

Story Points: 6/8	Tasks: 2/3

► Sprint 7:

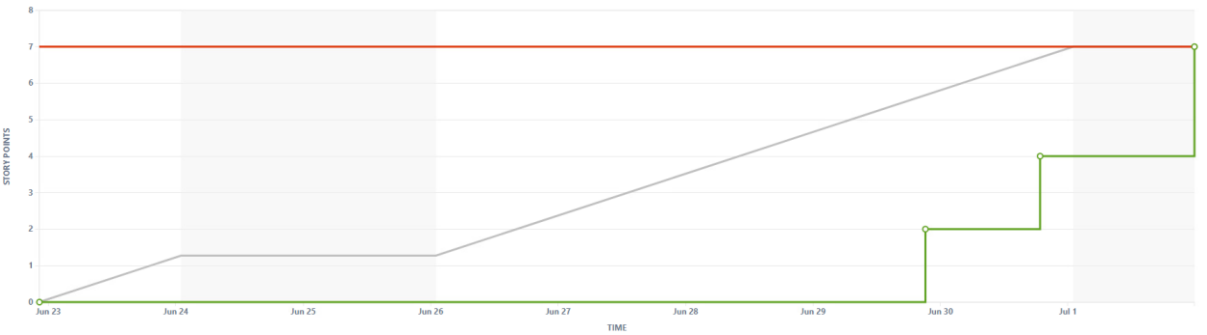


Fig 6.8 Sprint 7 Burnup chart

Date: 23/06/22 – 02/07/22

This sprint picked up the incomplete backlogged task introducing the busy and selected state for colonists. Further this sprint built on the intractability with the topography layer, adding the beginning of a placement system, by displaying whether a tile is occupied on the topography layer when hovered over. The selected state would allow the player full control of a single colonist.

Story Points: 7	Tasks: 3

- Sprint 8:

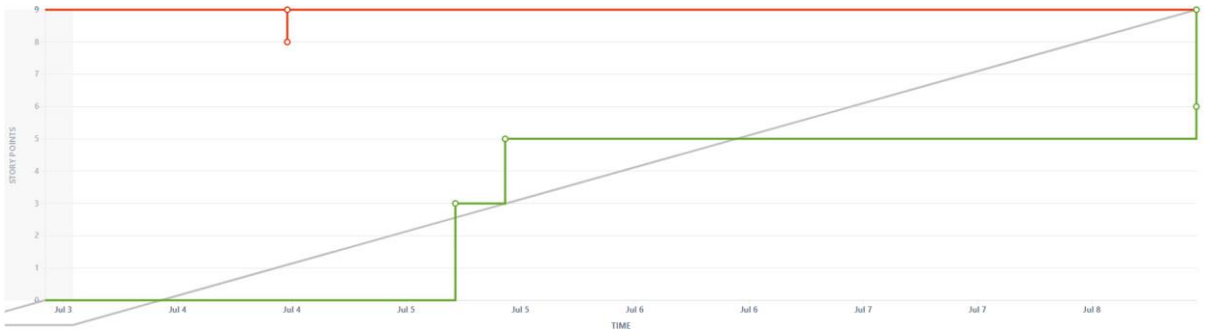


Fig 6.9 Sprint 8 Burnup chart

Date: 03/07/22 – 08/07/22

A save and load system was introduced into the game. To do this a SaveObject was created to be able to save information to a file which would then be read to load the previous games session into the current. Further traits were introduced for colonists to be included in data that is saved.

Story Points: 9	Tasks: 4

- Sprint 9:

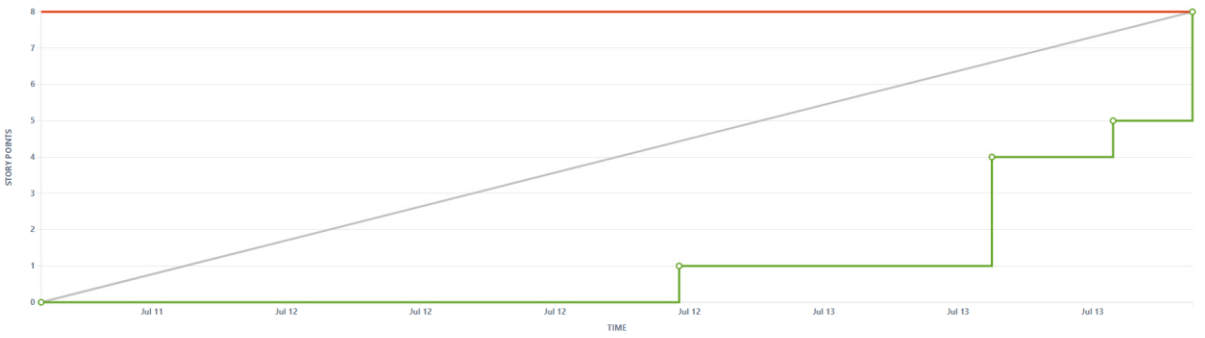


Fig 6.10 Sprint 9 Burnup chart

Date: 11/07/22 – 13/07/22

The was intended to be a very short sprint as to keep tasks in each sprint sessions somewhat related to the main objective. The main objective of this sprint was to create the menu which would be the first scene the player interacts with before going into the generated map scene. To this end, a menu was created with labels, input fields, buttons and dropdowns. This allowed the player to choose a new game, where colonists could be customised or traits could be re-rolled, or load a game from a save which would be displayed in a dropdown.

Story Points: 8	Tasks: 4

► Sprint 10:

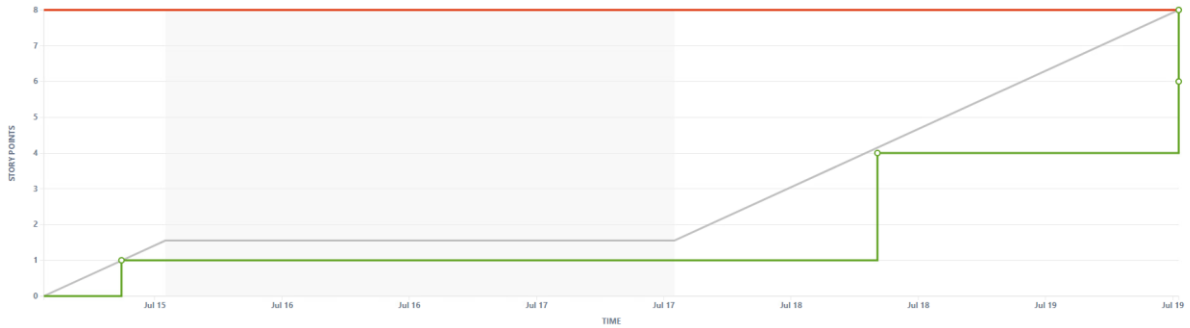


Fig 6.11 Sprint 10 Burnup chart

Date: 15/07/22 – 19/07/22

This sprint was dedicated towards tackling the mining and resource system for the game. This entailed creating a system to switch between different modes, such as mining or placement mode. Further creating a method to check if a task is able to be carried out by checking if there are any neighbour tiles available to walk to. A TaskTimer function was created to be able to govern actions when tasks are complete.

Story Points: 8	Tasks: 4

► Sprint 11:

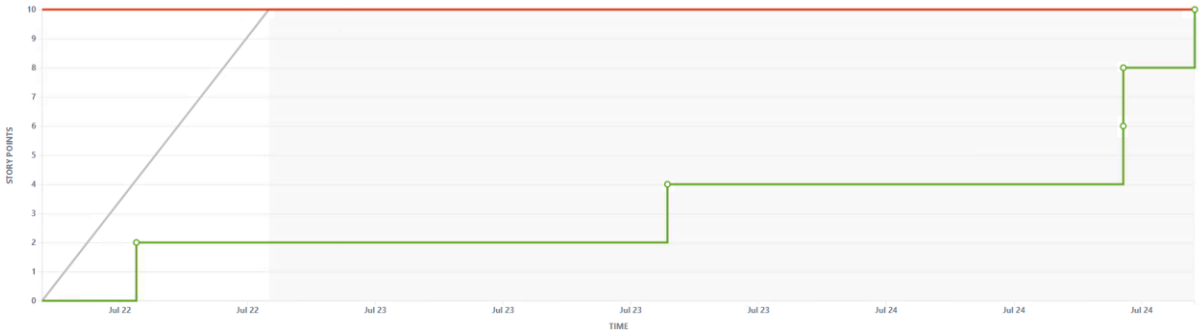


Fig 6.12 Sprint 11 Burnup chart

Date: 19/07/22 – 24/07/22

This sprint focused on refining the relationship between the mining and building modes. Sprites were created for the different required buildings in build mode, along with a UI to choose between buildings. Further due to the size of buildings being bigger than the standard time map size, functions were added to handle the placement and removal for tile sizes 2x2 & 3x3.

Story Points: 10	Tasks: 5

► Sprint 12:

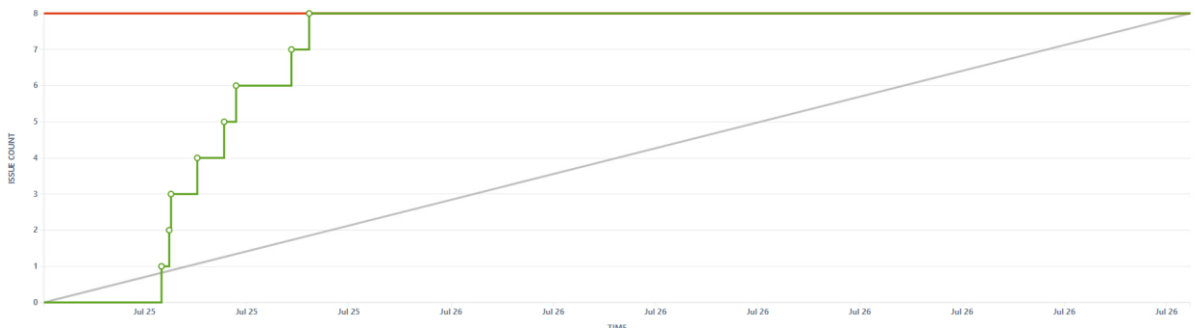


Fig 6.13 Sprint 12 Burnup chart

Date: 25/07/22 – 26/07/22

This sprint was intentionally short as it aimed to address bugs that were being noted down through development. These would include quality of life changes such as finding the shortest path to a task, removing GameObject creation bugs to reduce memory usage, queue issues, file SaveObject issues and index out of bounds issues.

Story Points: 8	Tasks: 8

► Sprint 13:

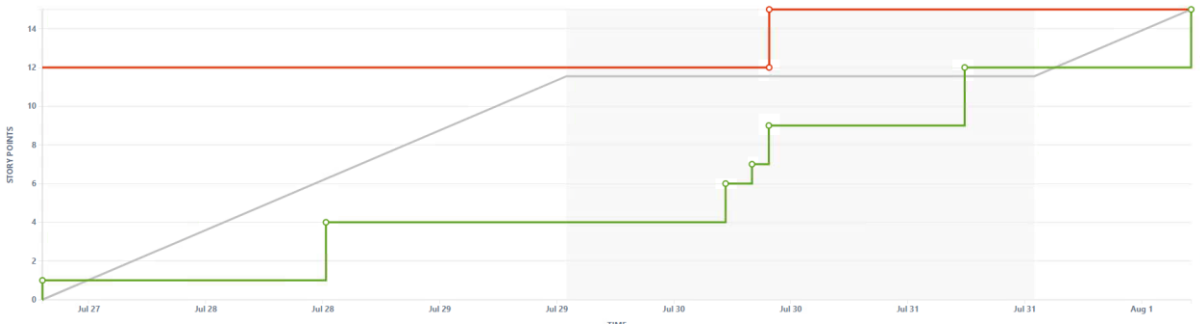


Fig 6.14 Sprint 13 Burnup chart

Date: 26/07/22 – 01/08/22

The focus for this was to create the Missions that players will send their colonists on. The implementation of this will allow for data from players decisions to be collected. Further it was time to ensure that traits are relevant to the game by making colonists traits influence the various tasks that are possible. This sprint saw to the final alteration to the topography map, ensuring that there is a visual for every 3x3 tile combination. The UI on which the players would send colonists on mission was also implemented & exporting of the game was tested. An empty queue bug was unable to be addressed during this sprint and thus was put in the backlog.

Story Points: 8	Tasks: 8

► Sprint 14:

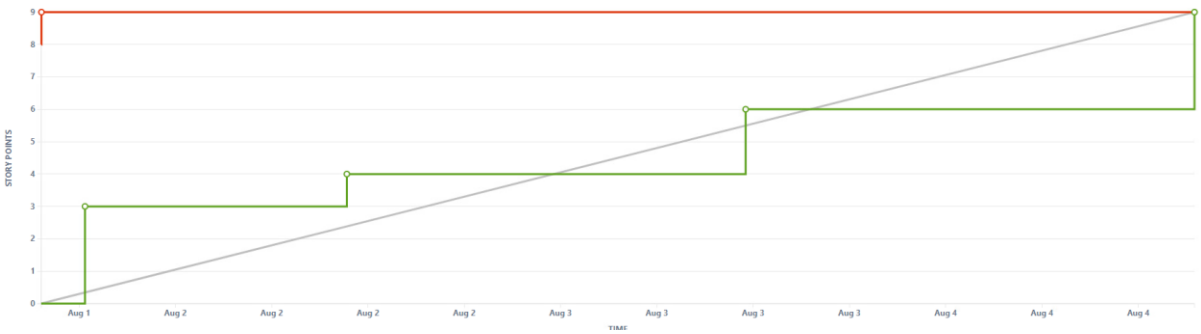


Fig 6.15 Sprint 14 Burnup chart

Date: 01/08/22 – 04/08/22

This sprint focuses on the Mission System implementation but required brainstorming to think of the scenarios to present to the player for implementation. After the scenarios were chosen, they were implemented into different banks to be picked randomly according to the chosen mission. Further a special category of Mission tasks was created to ensure that the missions were prioritised over other tasks. The UI associated with displaying scenarios to the player was implemented.

Story Points: 9	Tasks: 4

► Sprint 15:

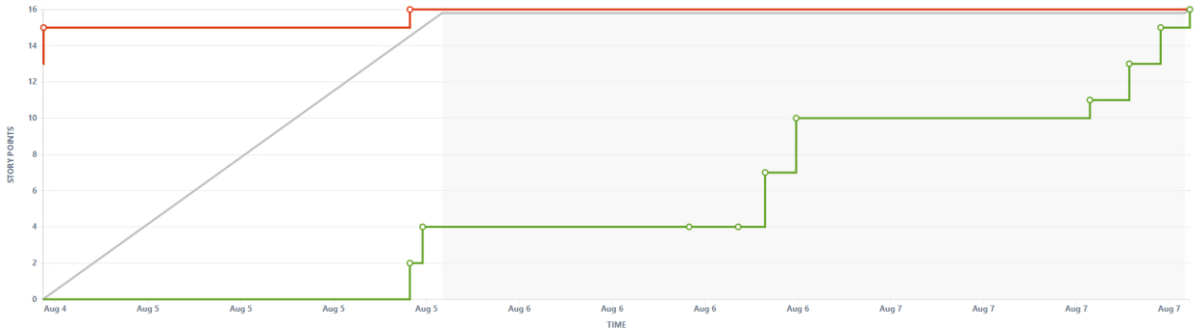


Fig 6.16 Sprint 15 Burnup chart

Date: 04/08/22 – 07/08/22

Approaching the development deadline, this sprint focused on implementing the final systems that it requires to the game. The farming task was implemented, the resource system now updated correctly with UI to display the resources a player has, task interruption was now being handled as intended, the colonist spawning system was added along with colonist customisation. Finally, decisions made were being correctly stored and saved onto the SaveObject for data analysis.

Story Points: 11	Tasks: 11

► Sprint 16:

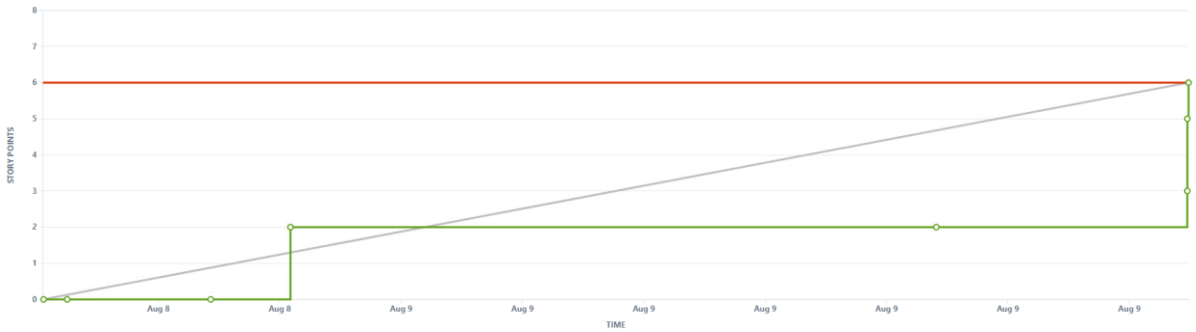


Fig 6.17 Sprint 16 Burnup chart

Date: 08/08/22 – 09/08/22

The final sprint revolved around ease of understanding of the game for the player and ensuring the game is able to be distributed as intended. A tutorial detailing all the information required to play the game was created to be sent alongside the game executable. The final issue with the pathfinding through diagonal neighbour tiles was unfortunately unable to be completed and will be addressed in the evaluation of this paper. Visual clarity was added to labels with some color-coded quality of life changes & finally basic testing to ensure each individual system was working as intended.

Story Points: 8	Tasks: 8

Evaluation

During the testing period of the game two hotfixes were released. Initially players were finding it very difficult to be able to obtain enough resources from the environment and further the rewards from the first mission were creating situations where players would get soft-locked. This is where the game seems apparently playable, but further progress is out of the player's control. To address this the initial hotfix addressed soft-lock issues and also addresses some quality of life changes:

- Removed stone gain from small Asteroid mission → Better rewards increasing resource gain
- Doubled resource gain from mining crystals → Reducing chances of soft-lock
- Added a visual color for the chosen colonist in the Colonist Tab → Increased understandability
- Increased chances for MK1 core → Faster progression for alpha-testers

The next hotfix doubled the amount of starting resources in an attempt to reduce the amount of time it was taking to be able to progress through the game. The Test Cases were able to check the robustness of each system individually, with 17/18 of the cases passed. The failed case (#17) is a known issue with two parts, the first being with the relationship between Unity's RigidBody and the PathFinding system. As colonists receive a path vector list defining the route for their bodies to take along a grid representing the Tilemap, if the colonist's movement is slightly interrupted/off-set by the interactable topography, the colonist's pathfinding may move to the next vector when their body is not actually in the correct location of the previous vector leading to the colonist getting stuck in the environment. The solution to this would be to make the colonists RigidBody not interact with the environment, but this is prevented by the second part of this test case where the pathfinding would not recognize two diagonal neighbours as impassable. Hence, the fix for the first part of this issue would allow for smoother colonist movement but allow for the second part to persist even more. To be able to combat this issue the colonists would need to have their interaction with the environment simulated and the pathfinding adjusted accordingly, but due to the timeframe of the project, it was decided to backlog the issue as it proved to be benign for conducting the intended research.

The data displayed that 4/7 of the alpha-testers (3,4,5,6) were unable to complete their games holding a combined count of 53 decisions, which is less than participant #2 holding the lowest decision count of players finishing the game. This clearly indicates that those players either got soft-locked while playing the game or did not make appropriate decisions leading them to lose the game. The data from these unfortunately are not able to provide any useful information for the research as the player had not been able to play long enough for their decisions to show any meaningful pattern, but do assist regarding the playability of the game. The remaining participants had enough data for an analysis hence, another table was created with counts converted to percentages using their total neutral, attachment and risk decision counts allowing for easier comparability in evaluation.

Participant #	Total Decisions	Risk Decision	Attachment Decision	Neutral Decision
1	114	26%	7%	67%
2	56	61%	9%	30%
7	239	9%	0%	91%

Observing the difference between the participants shows that as the participant's total decisions increase, the number of risky decisions made decreases and the number of neutral increases. The more missions a player embarks on increases the number of decisions a player has to make, hence the number of decisions displays the difference between each participant's playtime. With this in mind, the data shows that more risky decision-making is more common at the early stages of the game and the longer the participants played the safer their decisions became, sticking to neutral missions with no risk. This could be due to players taking more risks early to understand the game or taking risks without understanding what the cost could be. Players are more likely to make less risky decisions if

the exact probabilities are known, as shown in *Fig 5.11* 74.1% of people agreed with this, and I believe through experience players were understanding the risk involved more leading to more informed decisions.

Another factor to consider is the difference in progression through risk or preparation, where risk means you are willing to take chances for faster/further progression or even rewards and reparation means you take precautions to ensure you do not lose progression or things you value. Participant #7 regarding *Fig 5.2* in the form, indicated they prioritise progression through risk while having 91% of missions being embarked on neutral missions indicating prioritising preparation. A participant choosing safer choices could indicate a form of attachment towards the player's colony as a whole surviving over the attachment of individual colonists. The feedback indicated through the survey responses in *Fig 5.1 & 5.4* that risk is clearly more welcome and prioritised in games with 85.7% (*Fig 5.8*) of participants prioritising progression in 5YM. Though some players did not form any attachment with the colonists during their play-through, the players recognised how these features can encourage attachment with *Fig 5.9* showing that 85.7% of individuals would create deeper levels of attachment with colonists if there were more customization features that encouraged it.

While this is preliminary data there are patterns already appearing between the 3 participants, therefore indicating that with an increase of participants and scope for development of the game's attachment features the study would be able to collect more accurate data. This would allow for more informed decisions on *whether an individual prioritizes attachment or progression in their decision-making process when the infrastructure for each is provided.* The goals for the game were to have decision-making, problem-solving, balance, player attachment, progression and player usability. I believe I was able to deliver these goals in various forms through the implementation of the game, though the goal that proved to be the most challenging was balancing. While balance was focused on attachment vs progression, there was neglect towards the balance of the game itself with mission rewards and resource availability hence the hotfixes for the game during testing. The importance of game balancing will need higher emphasis in future projects allowing for more data collection as players do not get soft-locked.

Discussion

Overall, I believe the project has been able to meet the goals and objectives set through various implementations, providing a good insight into how games can encourage attachment and how these bonds are developed. The research has shown that attachment in games has more layers that initially predicted, especially when we consider that individuals have preferences towards different genres of games. *Fig 5.3* indicates that the group of participants found more levels of attachment in RPG or action-adventure games compared to simulation games, displaying that individuals have varying views on games that create attachment, where different genres of games have different goals for creating an attachment between the player and game as previously discussed. Hence, colony simulation games for some individuals would not create as meaningful connections of attachment as they might in role-playing games, creating a situation where player preference could heavily influence the attachment a player would experience even if there are features encouraging attachment implemented into a game. The paper was also able to explore how other forms of attachment could be seen to do with progression through risk or preparation.

Increasing the scale of this project would allow for more accurate assumptions of individual preferences while also allowing for further in-depth analysis of player decision-making between attachment and progression. For this, the game would need to address any issues/bugs present as it would go hand in hand with the player's enjoyment and create deeper connections of attachment with the player. While the system's individual components worked well as the test cases showed, this was because the order of tests was how I, the developer approached playing the game. The alpha testers would play through their eyes and encounter issues that were unnoticed because finding bugs to do with the different implemented systems overlapping in orders that were not envisioned during development. Games have to go through alpha testing to see and be able to view issues in the game from the player's perspective as no one will play exactly as you did.

The written feedback received (*Appendix 2*) indicated that individuals overall had a pleasant experience testing the game with 71.4% of people suggesting they would recommend 5YM to others (*Fig 5.11*), but also pointed out issues to be addressed. The following future work considers player feedback and developmental issues:

- User-friendly indicators/experience:
  - Highlighting tiles where tasks are queued
  - Scalable UI according to user screen size
  - Resource UI persisting during all scenes in the game
  - Sound creating a more enjoyable and bearable experience playing
- Colonists:
  - Making colonists look like something people can relate to allows for more of an attachment instead of plays regarding them as a resource.
  - More features to interact with colonists and care for them, such as needing food to keep colonists alive.
- Balancing:
  - Creating a method for renewable resources preventing cases of being soft-locked completely
  - Rather than rewards being completely random, having weighted chances that increase with the number of times a mission was done
- Bugs/Issues:
  - Embarking on multiple missions at once may remove 1 of the missions completely
  - Pathfinding as previously discussed
  - Certain buildings randomly disappear after an exact series of tasks

Appendix 1

Welcome to a Colony Simulator type game -- 5YearMission for my MSc Dissertation!

You are an overlord and your colony has crashed into a distant asteroid on the way back to their home planet! The objective is to collect enough resources to be able to send your colony on a final 5-YEAR MISSION back to their home planet.

Controls:

- Q || Opens the Task menu
- ESC || Opens the Pause menu
- SPACE || Turns on the Building menu
- M || Turns on the Mining mode
- W || Pan the camera UP
- A || Pan the camera LEFT
- S || Pan the camera DOWN
- D || Pan the camera RIGHT

Resources:

- Resources are gained by mining the area around & by sending your colonists on missions.
- Resources are displayed on the top left.
- If there are insufficient resources for a task, the task will not be completed.
- The resources are color-coded and correspond with tooltips indicating required resources.
- Mining mode (**M**) allows you to mine the environment for resources.

Building Menu: (**SPACE**)

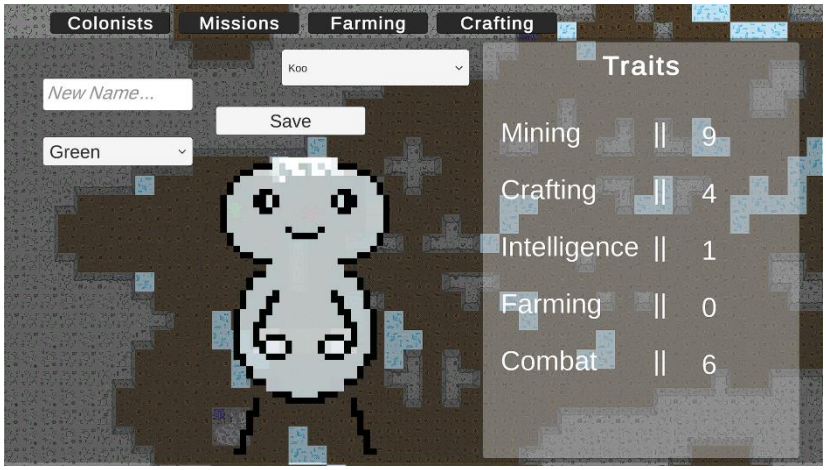
- There are 5 different building blocks (from left to right):
    1. Basic wall
    2. Research station
    3. Shipyard station
    4. Farming station
    5. HUB
- Resources required to build are displayed to the left of the UI

Task Menu: (**Q**)

► Colonist Tab

Here you are able to see your colonists' stats & are able to rename/recolor a chosen colonist.

Colonists' traits influence how long tasks take:

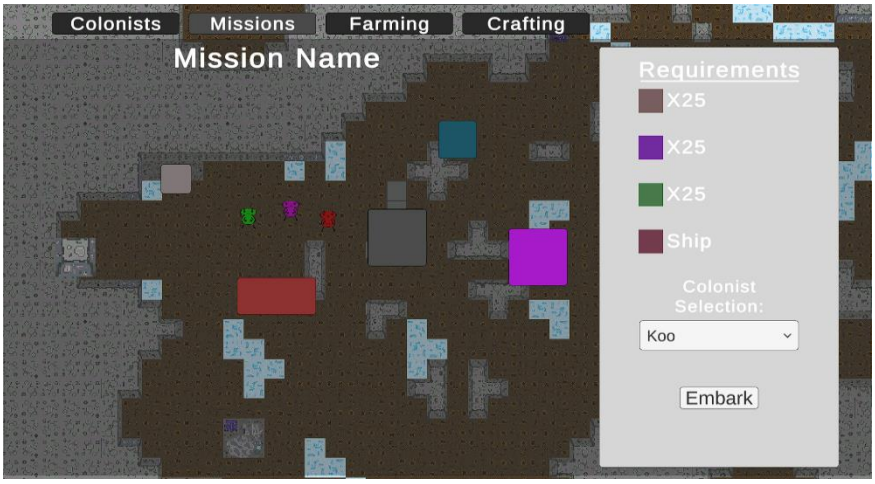


- Mining → Time to mine & resource gain from missions
- Crafting→Time to craft ships
- Intelligence→Time to reach missions
- Farming→Time to produce food
- Combat→Chance to win a fight

► Mission Tab

Here you are able to see the various missions to send your colonists on.

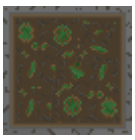
You will need to work through each mission location to get the right parts to build the ship for the final mission!



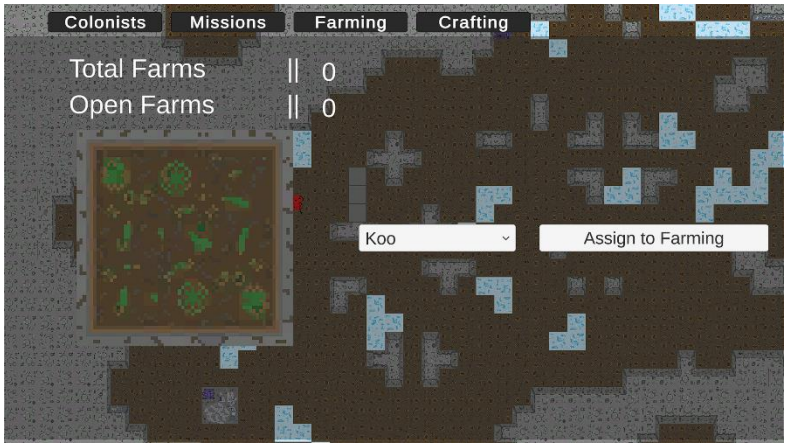
To be able to embark on missions, you will need your HUB built.

► Farming Tab

Here you can set chosen colonists to farm, producing food for missions!



You will need a Farming Station built before you can farm!



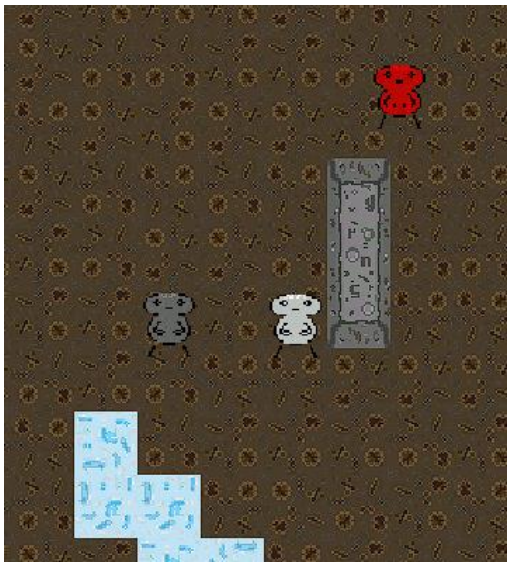
► Crafting Tab

Here you can build ships for your colonists to go on missions!

You will need the Research station built to be able to craft the ships.

You will also need a Shipyard to house your ships.

Colonist Control:



- Colonists can be taken control of by *Left Mouse Click* and become a *white* color to indicate they are **selected**.
- Colonists which are **busy** doing a task become a *grey* color.
- Tasks can be interrupted by **selecting** a colonist who is currently **busy**.

- ***Selecting*** *any colonist who seems stuck will reset them allowing any issues to be bypassed with this.*

This game records decisions you make throughout the game.

After the game's completion please find your save folder in the game's location:

/5YearMission_Data/Saves

Please forward the folder back to ds18635@essex.ac.uk

Thank you for your time Alpha Testers!

Appendix 2

What do you think these games have in common to create attachment?

7 responses

- Characterised NPCs
- they provide players with a reason to care.
- User-based choices that affect gameplay.
- well built story connecting the character to the world
- They have different characters each with their own special abilities. You then form attachment to them based on their skill set, how easy they are to play and how likely they are to win you the game.
- Character writing and standout traits
- More lore and story driven characters. More ways to connect with them as opposed to just "a character in a game".

What aspect of the game did you find most difficult?

7 responses

- Limited resources
- Not getting softlocked
- Gathering recourses.
- The clarity of progression, the mechanics needed to be introduced in a sequential manner, this is mainly due to having a lot of interconnected systems which are hard to grasp in a new game.
- Understanding when to do the first mission again in order to give me enough crystals etc to further progress and do more missions
- RNG of which materials gained from missions
- Obtaining certain materials required for new missions.

To the best of your knowledge, please identify any major bugs/issues that would **take away from the games experience/progression**:

7 responses

The ability to mine out the entire map and then lose all resources on unsuccessful missions

No renewable sources of resources (besides food)

(Pathfinding) The colonists where getting stuck between two diagonal blocks
(Balancing) A lot of resources were required in order to complete the game in the given world where resources were limited.

The type/amount of mined items needs fine tuning to reduce the chance of a soft lock or the need for grinding to proceed

The characters getting stuck when mining the blocks. Sending 2 characters on missions, only for 1 of them to "get lost" and thus the ship disappearing which hindered progression as I had to go back to mission 1 and rebuild the basic ship again

Possibly have minable resources be able to regenerate, so you have an active activity for the colonists who are not on a mission to do

Colonists becoming stuck on terrain. Also, there seemed to be a bug where certain items would disappear thus causing a significant loss of progression.

What parts of the game do you **dislike**?

7 responses

No renewable form of resource gathering, no way to track which areas are selected for mining

* softlocks
* The visual design of the colonists is somewhat unappealing to look at
* Monotony (no reason to not just mine everything out (beyond the time investment for that))
* No idea what the 'crafting' and 'intelligence' attributes for the colonists actually do

Pathfinding, Fixed Interface Scale, Task Queuing for Colonists sometimes skipped or duplicated tasks.

Only the instructions and the clarity of how to proceed with missions and their requirements

Having to tap each and every block to mine - it was long. The fact that when viewing what materials were needed for a mission, you had to physically go back to the main screen where the characters where to see how many of each material you had.

I think some of the rng could be tuned or made clearer (I believe I got certain materials a lot more from missions and was unsure if it was due to the rng or if those materials are more common)

Spending large amounts of resources on missions who's rewards weren't equivalent for the cost to obtain them.

What parts of the game do you **like**?

7 responses

The ability to develop certain skills through use in colonists

* besides the softlocks, it functions
* clear end goal for the player

Busy Indicator, Colonist Customisation, Procedural Map Generation

That there is a looot to play with, a great number of mechanics and systems implemented, it looks like it just needs more time to be developed further.

The characters were very fun - I built attachments to them and even sent out the new characters I got from the missions onto new missions because I did not want to use the characters I liked that had good skills incase I lost them through a risk. I liked the fact that the game allowed for the progression of characters, like building mining skills and getting "pulsating" intelligence. Taking a risk was always fun regardless as even if I lost a character, I managed to gain quite a few more. I loved the concept of the game in general !!!

Exploration and progression

Very good sense of progression, throughout the game. There is a clear beginning, middle and end to the game and you don't ever feel like you aren't making any progress. Consistently engaged with "obtaining more". The design is simplistic in a way that doesn't feel simplistic.

Feedback & opinions on the tested game.

Please add any other thoughts or suggestions you would like to share here.

7 responses

A highlight on areas selected for mining, renewable resource gathering (maybe at cost to food to make management more complex)

Not sure what the colonists are supposed to be. You could consider ripping off the aesthetic of 'Oxygen Not Included', make the colonists look like their own people, that might make them a bit more endearing and help encourage emotional attachment.

It would be nice if there was some risk/reward in the main game world area (maybe occasionally mining into an alien nest or something meaning that your colonists need to defend against some aliens?) instead of the current monotony.

Audio would be nice as well.

Regardless of any problems that I may have mentioned, I sincerely see this as an amazing work in the very short period in which it was built, big thumbs up!!

Exceptional game, well done!

Some ambient space-themed soundtrack (If that is in your talents to compose one)

Generally a good game. Requires some tinkering with regards to a few minor technical aspects however, it didn't take awake from the general enjoyment of the game.

Bibliography

[1] Arkane Studios. Dishonoured. Bethesda Softworks, 2012.

[2] Nintendo R&D4, and Nintendo R&D2. Super Mario Bros. 2. Nintendo, 1988.

[3] Banks, J., and N. D. Bowman. “Close Intimate Playthings? Understanding Player-Avatar Relationships As a Function of Attachment, Agency, and Intimacy”. AoIR Selected Papers of Internet Research, vol. 3, Oct. 2013, https://journals.uic.edu/ojs/index.php/spir/article/view/8498. Accessed 20 Aug. 2022.

[4] Melissa L. Lewis, René Weber, and Nicholas David Bowman.CyberPsychology & Behavior.Aug 2008.515-518.http://doi.org/10.1089/cpb.2007.0137.

[5] Baek, Joeun, et al. “Designing a Minecraft Simulation Game for Learning a Language through Knowledge Co-Construction.” Teaching, Learning, and Leading With Computer Simulations, 2020, pp. 181–208., https://doi.org/10.4018/978-1-7998-0004-0.ch007.

[6] Coffee Stain Studios. Goat Simulator. Coffee Stain Studios, 2014.

[7] Sublogic, Bruce Artwick Organization, Aces Game Studio, Dovetail Games, Asobo Studio. Microsoft Flight Simulator. Microsoft, 2020.

[8] Ludeon Studios. RimWorld. Ludeon Studios, 2018.

[9] S. Lambert, “Quick Tip: Intro to Object-Oriented Programming for Game Development,” *Game Development Envato Tuts+*, 29-Oct-2012. Available: https://gamedevelopment.tutsplus.com/tutorials/quick-tip-intro-to-object-oriented-programming-for-game-development--gamedev-1805.

[10] S. Minhas, “Agile design process,” *Medium*, 06-Aug-2020. Available: https://uxdesign.cc/agile-design-process-24be92018ad2.

[11] Technologies, Unity. “Mathf.PerlinNoise.” Unity, 2021, https://docs.unity3d.com/ScriptReference/Mathf.PerlinNoise.html.

[12] Weisstein, Eric W. "Moore Neighborhood." From MathWorld--A Wolfram Web Resource. https://mathworld.wolfram.com/MooreNeighborhood.html.

[13] Garside, Ben, and Rebecca Franks. “A* Search Algorithm.” Isaac Computer Science, 2020, https://isaaccomputerscience.org/concepts/dsa_search_a_star?examBoard=all&stage=all.

[14] Becker, Alexander, and Daniel Görlich. “What Is Game Balancing? - an Examination of Concepts.” ParadigmPlus, vol. 1, no. 1, 2020, pp. 22–41., https://doi.org/10.55969/paradigmplus.v1n1a2.