Department of Computer Science

Submitted in part fulfilment for the degree of MEng

Interactive Resource Management Simulation with AI Elements

Sazidul Hoque

30 April 2024

Supervisor: Antonio Garcia-Dominguez

ACKNOWLEDGEMENTS

I would like to express my gratitude towards my supervisor, Dr Antonio Garcia-Dominguez for your insightful feedback and expertise throughout the entire project.

I am grateful to all those who participated in my user study whose responses and perspectives have guided my evaluations. Their willingness to dedicate their time to this project is deeply appreciated.

Lastly, I couldn't have undertaken this journey without my friends and family with their unwavering support and encouragement.

STATEMENT OF ETHICS

Ethics approval from the University of York Computer Science department was granted for this project for a User Study entailing a playtesting session. All Documentation concerning this is available in the submission folder under the folder name Supporting Evidence.

All raw data has been kept anonymous for the rights and privacy of the participants. All of this information has been passed over, with informed consent forms acquired for all involved participants.

TABLE OF CONTENTS

CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

Executive summary

The motivation behind this project was driven by the recognition of artificial intelligence (AI) as a phenomenon in the gaming industry. As AI becomes increasingly ubiquitous in modern gaming, it led me to wonder what the impact of AI has towards games and how it might influence the experience of a player. By delving into the implementation of different AI techniques, this project aimed to answer these questions through developing a game with AI elements as the forefront, to isolate how it might influence player experiences.

The objectives of this project were split into two dimensions. Initially, the primary focus was on the functionality of the game and in particular the AI that was to be developed. But this needed to be evaluated on a more subjective scale, with the second primary requirement of this project reliant on the enjoyability of the game and inherently its AI features.

This project details the methods that were set in place to achieve the aforementioned objectives. Adopting an agile methodology, specifically Scrum, provided a structured framework that became vital for managing the projects iterative development progress.

Before proceeding with the implementation segment of the project, it was important to establish the product and software design associated with the game. By creating a game design document, the different mechanics and features were set out, outlining the vision of the game. Refining these requirements in order to create a minimal viable product (MVP) that served in verifying the functionality of the game.

Establishing the games engine was equally important, influencing the architectural design that organized the software components of the game in a maintainable and expandable manner.

Various evaluation methods were required to gauge the fulfilment of the different objectives set out. In terms of verifying the functionality of the game and its AI features, a combination of manual and unit testing was carried out to determine its success. Whereas determining the enjoyability of the game employed the use of a user study to evaluate the subjective experiences of the different aspects of the game.

Playtesting sessions required significant ethical considerations to be taken into account. This included providing participants with informed

consent forms and information sheets to ensure transparency and respect for their rights and privacy. Observations and post-play survey acted as additional mediums for gathering different types of data.

Key findings from the evaluation exceeded expectations in regard to the MVP requirements. While not every detail outlined in the GDD made its way into the final game, all the set-out requirements had been successfully verified through the testing methods. From the observations and feedback received it was apparent that the game resonated differently with players with valuable responses to improvements and appreciated features. The post-play survey also accumulated quantitative data, which allowed me to determine the general sentiment towards the game. Analysis of the results revealed overwhelmingly positive feedback towards the enjoyment of the game and the effectiveness of the agent-based AI incorporated.

On concluding this project, it was evaluated that AI enhances traditional gameplay mechanics, introducing another level of depth in terms of interaction and strategy. Moving forward, transitioning from a single-agent system to a multi-agent system would be intriguing. Furthermore, expanding the agent's environments to a more complex setting would encourage developing more sophisticated agent behaviour.

1 Introduction

In recent years, the introduction of artificial intelligence into the gaming industry has become something of a phenomenon. With concepts like procedural generation on whole planet like environments, to algorithms that can predict player behaviour and adapt to this knowledge.

AI in gaming has now developed to such a scale where its presence in games is now almost seen as something inherent and an expected component of modern gaming experiences. It has come to a point where there will almost always be some form of artificial intelligence subtly shaping our gameplay without necessarily recognizing it as such.

The motivation behind this project centres around wanting to understand the role of AI and how it can affect player experiences. Moreover, delving into the intricacies of its implementation, I'm hoping to uncover the ways in which it works to improve games and their enjoyment factor.

Understanding this can be a vast subject, I've decided to localise this through undertaking the process of developing a game with AI elements, narrowing this down to a level of creating some form of intelligence in the virtual realm of a game. With resource management being known for its autonomy, using this as the genre of the project will aim to paint AI as the forefront of the game. Through evaluating its autonomous capabilities, I'd like to determine whether it can replicate *its own* decision making and whether this will actually add a dimension to the players gameplay.

The game developed as part of this dissertation will serve as both a testament to the passion and enthusiasm I have towards gaming and AI, but also as a medium through which I can meaningfully engage with the topics themselves. Ultimately, this project attempts not only to entertain and engage players using AI but also to inspire and inform me of the broader applications of the subject in this industry. Through research, analysis, and reflection, this work will hopefully aim to uncover the potential of AI in gaming.

2 Background Research

The general theme of our research revolves around studying the world of resource management games, delving into their characteristics, functionalities, and the elements that make them effective. This is looked at with a larger emphasis towards how agent-based artificial intelligence (AI) is integrated into these games and its impact, on gameplay.

An investigation into existing game titles and their AI frameworks, will provide insights into techniques, strategies, and potential implementation methods. The findings from this research will be integral in laying the groundwork for discussions on introducing agent-based AI into the concepts of resource management and real-time strategy.

Through a review of existing literature and implementations, the aim will be to uncover the properties and capabilities of the different types of agent-based systems used in games. Dissecting their advantages and identifying optimal strategies will provide a means to uncover practical ways in which this mechanism can enhance the gaming experience.

Congregating all these facets of research will be vital in guiding the decisions made during the design and implementation phases of this study.

2.1 Resource management games

Resource management games are a type of interactive simulation which requires strategic thinking enabling players to efficiently leverage limited resources in order to achieve a certain goal [1]. This genre is designed to be challenging with unknowns for the player to understand, moulding their strategy through compelling decisions - at its core, tapping into our instincts for optimizing, planning and strategic decision-making.

The management of resources itself stands as a foundational component, virtually hidden within every gaming experience. Whether it involves stockpiling building materials or the conservation of a player characters' health, the prevalence of this mechanic is evident. The universality of resource allocation and utilization principles in gaming likely explains the widespread appeal of resource management and how it has evolved into a genre in its own right.

Offering a dynamic gaming experience that can be tailored to suit different player preferences further bolsters the genre's popularity. Optimizing production chains in industrial simulations such as Factorio

[2], or a more relaxed pace of managing resources on a farm in Stardew Valley [3] – there is a resource management game to suit all tastes.

This genre is notably associated with others such as strategy and simulation which have consistently been mentioned among the top genres of recent years. A 2022 survey of gamers in the United States [4] found that 41 percent of respondents played simulation games regularly. A further 40 percent of responding gamers stated that they played strategy games on a regular basis.

To fully comprehend the underlying reasons for this popularity, it is necessary to pick out the key elements that characterize the success of resource management games, sourcing that which elevates their engaging and stimulating gameplay.

2.1.1 Core gameplay mechanics

While the preceding discussion has established a broad understanding to the popularity of resource management games, it's important to consider why this is so and direct our attention to the specific dynamics that define the core gameplay mechanics unique to resource management experiences. From exploring the diverse components involved in developing successful resource management games, it's evident that we need to simplify these into fewer categories. For the purposes of this investigation these have been aggregated into four distinct elements:

- Resource Dynamics
- Strategy and Decision-Making
- Customization and Progression
- Feedback and Simulation

2.2 Important games in resource management

In order to better gauge where these mechanics originate from, it’s important to understand the perspective of games which have played a pivotal role in making resource management games as popular as they are today.

I have picked two games that are considered seminal games in the genre that have garnered numerous awards for their innovative gameplay. The Sims generally achieved a broader recognition amongst the gaming industry, with multiple editions of the game being awarded throughout the years. Of many, the original title acclaimed multiple awards at the Interactive Achievement Awards (also known as the D.I.C.E Awards) throughout the years. Notably winning “Game of the Year”, “Outstanding Achievement in Game Design” and

“Outstanding Achievement in Game Play Engineering” at the Third D.I.C.E Awards in 2000 [5].

In the same event, Age of Empires fell closely behind, with their title “Age of Empires II: The Age of Kings” achieving nominations for the previously mentioned awards alongside winning a couple of their own. Alongside accumulating the most nominations, they were also able to win “Computer Game of the Year”, “PC Strategy Game of the Year” and “Outstanding Achievement in Character or Story Development” [5].

Throughout the years, these two games have cemented their position as iconic titles, specifically in the strategy-simulation genre, serving as ideal models for understanding the intricacies of the core mechanics aforementioned.

2.2.1 Age of Empires

Age of Empires [6], a game where you must build a thriving empire as you expand through the ages [7], defeating any rival empires. The Age of Empires franchise developed by Ensemble studios, with their first game released in 1997, and several games and expansions in between then and their latest base game Age of Empires IV. Players start with a small settlement and must gather and manage four different resources [8], gathered from their surroundings: being wood from trees and forests, food from mills, farms and animals, stone, and gold from mines and so on. Advance through different ages, players unlock new technologies, units and buildings that provide strategic advantages – with all of this available for the player to decide the fate and outcome of their empire through their own *strategic decision-making*.

This game franchise was opted for as a game that best portrays our mentioned resource management mechanics, as it has not only revolutionized RTS gaming but also established a benchmark for resource management displaying the critical role of resource allocation in shaping strategic gameplay experiences.

In Age of Empires, each resource has its own unique gathering method, with *resource availability* being generally dictated by the players game-plan. By assigning villagers to resource points, like mills or forests, players can optimize their resource production according to their needs, thereby adding a level of automation to their gameplay. Providing this level of automation encourages players to think long-term, directing their focus towards advancement.

With a procedurally generated world, the player can gain strategic advantages through mechanics like Scouting. Uncovering locational

information about resource deposits or enemy settlements and armies, especially within the fog of war format, serves as an essential *feedback* mechanism for upcoming conquests. Meanwhile other visual cues such as resource stockpiles and population counts offer more immediate *feedback*, allowing players to fine-tine their ongoing approach.

Players constantly have to decide how to allocate resources. Between economic development and military expansion, weighing the benefits of investing in technology upgrades versus building military units to expand their army. Whether these upgrades may be improved armour for specific military units or advanced mills to bolster the production of food. It's evident how the player has multiple facets in which to play the game with a clear *progression* path allowing players to specialize in particular areas based on their strategic preferences.

The *strategic* depth of the resource management mechanics enhanced by its *customizability* provides substantial replay value, encouraging players to experiment with different strategies and refine their approach over time.

2.2.2 The Sims

The Sims is a popular life simulation game series that allows players to create and control virtual characters and live their lives from an overseeing perspective. The first iteration of The Sims was developed by Maxis and published by Electronic Arts in 2000. With the help of designer, Will Wright after his SimCity [9] series, this game followed a similar complex where instead of having players control the city, they are in control of the residents of the city. This franchise has had a large share of success with many expansions and game being introduced over the years - their latest and fourth major title The Sims 4 releasing in 2014.

Dedicated to the casual gamer, players have the freedom to create customize this game to their liking. From their (virtual characters are known as Sims) physical appearance to their personality traits, aspirations and life goals, the players are able to craft unique personalities and stories for their Sims.

Although The Sims may not seem like it belongs in the resource management genre, but this feature is crucially present throughout. This game stands as a pivotal title within this genre because it uniquely places players in the role of not just an observer but a controller of managing the lives of these characters. Unlike traditional resource management games where the focus is often economic or strategic elements, The Sims provides a unique perspective where all decisions matter, placing equal importance on the management of time,

relationships, and personal development, transforming mundane tasks like eating, sleeping, and socializing into captivating gameplay experiences.

Players need to evaluate all these different choices, for example, buying different levels of facilities that can affect the needs of Sims where a more comfortable bed would provide a better-quality sleep, or buying fitness equipment for an athletic sim would align with their aspirations. These *needs dynamically change* over time influencing their behaviours and interactions. With these Sims exhibiting *autonomous behaviour*, they're able to act based on these needs, for example, if a Sim becomes hungry, they will seek out food or if they're tired, they will prioritize sleep.

However, it still remains important for players to retain control over their actions and override them to steer Sims towards their own desired outcomes. Queueing up multiple actions for Sims to perform in sequence, give players opportunities to *strategically plan* in advance. Whilst this allows players to observe and focus their attention elsewhere, Sims will keep players in the loop, alerting them if they have unfulfilled needs or when significant events occur such as job promotions or relationship milestones. Players can approach this *visual and audible feedback* as means to assess the effectiveness of their current decisions and adjust their *strategies* accordingly.

This may vary from game to game, as players can personalize their gameplay experiences through the customizable features of their Sims. With multiple personality traits and life goals to set, there is immense potential for both short and long-term progression paths and end-goals, contributing to the enjoyment factor of the game.

2.3 Benefits of AI in resource management games

While the use of AI is universal in gaming, its significance is particularly vital for the resource management genre. There can be a discussion as to how AI can be involved in the development process of the game, however this is not the projects interests. Instead, we will look into how AI can improve the game quality and experience.

2.3.1 Micromanagement

Micromanagement is a core factor in this genre of games, with the player being required to oversee all forms of things from the production and gathering of resources to more strategic elements such as research and expansion. This can get tedious and overwhelming to the point where it loses its enjoyment factor. AI introduces the facet of autonomy which can alleviate this burden of excessive control, freeing up the player to focus on higher-level gameplay through strategy and decision-making. Whether this might be through assisting players by

automating routine tasks, or through the addition of intelligent agents who may be autonomous in nature.

This is evident in “The Sims” where the AI are known to perform tasks on their own. The way this is accommodated is through the different objects in the environment, advertising what they offer to the Sim, for example, a bed may offer 20 points to energy. So, when a sims’ needs become critical, they can assess their surroundings and prioritize interactions with those objects accordingly in order to meet their specific needs or desires.

This technique is represented by Fuzzy State Machines [10] (FuSM), introducing degrees of uncertainty which aim to shape the behaviour of the sim to be more flexible – this would mean needs like “hunger” or “happiness” would be replaced with terms such as “very hungry” or “slightly happy”. These fuzzy states are then used to define the conditions for transitioning between states or executing actions within the state machine. This “free will” aspect of sims significantly contributes towards the realism and depth of this game, mirroring the unpredictability and spontaneity of human-like behaviour.

2.3.2 Unpredictability

Predictability is a killer when it comes to the resource management genre, diminishing the challenge and excitement that comes with it. AI-driven procedural generation can prevent this through creating diverse and unique game worlds, and with it represent distribution of resources. Adapting in responses to changes in a game’s parameters, opting to do this in real-time rather than rely on pre-programmed responses, allow players to encounter new experiences with each playthrough. A computer-controlled entity would therefore need to be able to deal with many aspects of human level intelligence [11] – allowing the player to feel immersed and engaged through the sophisticated strategic behaviours that come with unpredictability.

When playing Age of Empires, it’s hard to not notice the intelligence of the enemy civilizations, with their strategies and tactics adapting to the changing world state and decisions made by the player.

In the early days, the behaviour of the enemy AI was determined through a mass number of “if-then statements” [12], where if a condition was met a certain action would take place, resembling the appearance and description of Decision Trees. Although, Marius “Promi” Beck, the artificial intelligence lead developer at forgotten empires, suggested this behaviour was actually based a rule-based system [12], governed by sets of predefined rules and algorithms.

Age of Empires have also used Finite State Machines (FSMs) [13] in conjunction with this to represent behaviours of different entities, by defining a finite set of states and transitions between them, allowing them to react to changing game conditions and perform actions appropriate to the current state. Pathfinding algorithms allow them to move autonomously based on where they need to go, through calculating the most efficient routes, taking into consideration different obstacles or other entities.

The various examples showcased techniques such as decision trees or finite and fuzzy state machines, all revolving around agent-based AI principles. It is evident that agent-based AI form the focal point of our investigation and thus the following section of this background research will be dedicated towards this.

2.4 Agent-based AI in resource management games

2.4.1 What are agents?

Agents are an integral aspect of artificial intelligence (AI) with their abilities in analysing data and utilizing it to formulate reactions to scenarios. As Franklin and Graesser [14] have articulated, agents are a “system situated within and part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future” – denoting a core concept of AI, the need to automate intelligent decision making.

This conjunctively aligns with Michael Woolridges’ [15] definition of an agent, where “a computer system is capable of independent action on behalf of its user or owner… figuring out for itself what it needs to do in order to satisfy its design objectives, rather than having to be told explicitly what to do”.

Considering both the ideologies represented by Franklin and Graesser [14], and Woolridge [15] we can better establish that an intelligent agent can understand the goals it is assigned and utilizes the information available to it to achieve those goals *on its* own .

2.4.2 The current state of AI in games

In the modern era, games with agent-based AI are widespread, with elements of this concept existing in all types of games ranging from those which may involve a single agent to a multi-agent-based system. Whilst a single agent may try to achieve a desired goal based on precomposed algorithms, a multi agent system would be composed of a coalition of agents that can interact with one another and its environment, together describing a much more complex system.

Deep learning frameworks such as neural networks have made drastic advances in developing agents. Using reinforcement learning in order to train itself in order to make extremely advanced evaluations of positions, with “AlphaZero” being a prime example. Although learning agents are a significant advancement in the realm of agent-based AI, our research will avoid machine learning in order to limit the scope of the project.

2.5 Commonly used techniques of agent-based AI

As discussed previously it is important that AI needs to adapt to the changes in its environment. Having that human-like level of understanding for our agents, allows them to intelligently make decisions or at least appear to do so. There are several agent-based techniques for performing decision-making, so we will attempt to understand the more commonly used approaches in games.

2.5.1 Decision trees

Decision trees are perhaps the simplest way to structure the decision-making process of agents in games. Known for their simplicity, this method can be associated can be best resembled by “if then” statements, where if a condition is met, proceed with an action. This can be extended with further conditions to represent a complex behaviour structured as a tree-like flowchart.

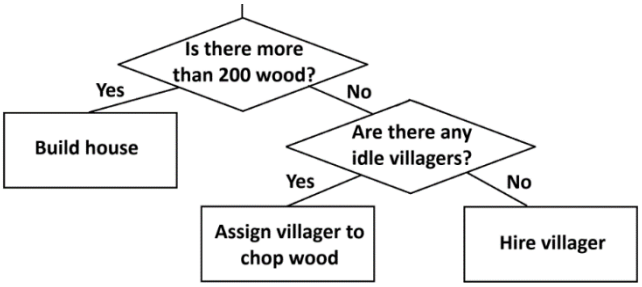


Figure 1: *Example of a Decision Tree in the context of an ‘Age of Empires’ Enemy AI*

Evaluated from root to leaf, decision trees begin with a single root, which branch out into various subtrees which consequently may branch out into other subtrees eventually terminating in leaves [13]. Here the roots and subtree nodes represent conditions, and the leaves represent corresponding actions.

The simplicity of decision trees is a notable advantage in this specific technique. However, as decision trees grow deeper, this sacrifices their simplicity and interpretability. This trade-off between depth and simplicity can represent its challenges, where the developer must effectively balance the complexity of the trees whilst ensuring they remain manageable.

2.5.2 Finite state machines

Integrating the concept of “states” with decision trees and incorporating these values alongside changes in the game environment [16] offers a pathway to achieve more sophisticated behaviour. Finite State Machines (FSM) split this behaviour into logical states where each state represents a distinct behavioural tendency [17]. Each state may transition to another state through different conditional(s) where the agents’ behaviour is determined through its current state – as it can only remain in one state at a time. These conditions may be based on the current game state, changes in the game environment or actions of other entities in the game.

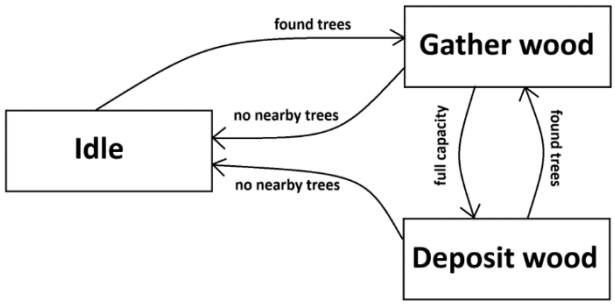


Figure 2: *Example of an FSM in the context of an ‘Age of Empires’ Lumberjack*

The presence of states provides a structured framework for organizing and managing the agents’ behaviour – deconstructing complex behaviours into smaller, more manageable states. This modularity and flexibility make it easy to add more states and modes accordingly however this comes with its limitations. As the number of states and transitions in an FSM grows, managing and maintaining the state space can become increasingly challenging. It can reach a point where the clarity offered by the FSM pattern is nullified by the sheer volume of states, making it more difficult to comprehend or adapt and reuse.

2.5.3 Behaviour trees

The issue of scalability is generally tackled through the use of behaviour trees where the organizing of behaviours is done in a hierarchical manner [18], allowing developers to create sophisticated behaviours by combining and nesting smaller behaviour trees. This modular structure makes it simpler to add, remove or adjust individual behaviours without affecting the whole system [19]. A behaviour tree is represented as a polytree (or a directed tree) following the general root, child, parent, and leaf node structure [20]. Here the leaf nodes are generally called execution nodes [20] determining the state of the agent, whereas the nodes prior, that direct the tree to these states are generally called control flow nodes [20]

Execution nodes have 2 different types: Action nodes and Condition nodes - conditions representing a question and actions representing a task to be performed. To better understand how these, it's important to determine the behaviours represented by the 3 different types of control flow nodes:

In Sequence nodes, all its children are executed sequentially [19], representing a behaviour where all tasks/conditions would need to be successfully completed in a specific order.

Fallback (also known as Selector [19]) nodes will only execute the first child that is successful, more suited for implementing behaviours where there are different choices, allowing the agent to try different actions until one succeeds.

And Parallel nodes execute all of its children simultaneously, determining success through an initialised success threshold [20] - a more particular behaviour dependent on the success of a subset of actions.

The possibilities from combining these different control nodes offer a flexible framework for shaping AI decision-making pathways to handle complex behaviours effectively.

We can see the different type of nodes in a behaviour tree in action below:

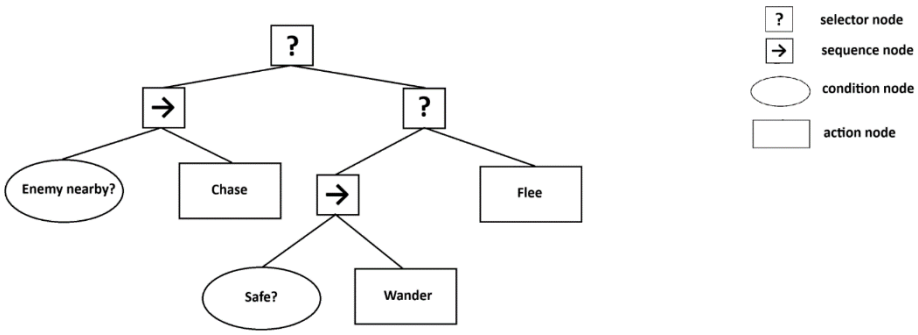


Figure 3: *Example of a Behaviour Tree representing a basic Enemy AI*

Examining the manners of how a behaviour tree works in action, reveal how they represent major aspects of both paradigms, decision trees and FSMs'. By integrating the profound conditional evaluation of decision trees with the structured state-based behaviour of FSMs', behaviour trees provide a powerful and flexible framework for modelling complex and adaptive AI behaviour in games. While this remains true, their hierarchy of nodes in this framework requires careful planning and considerations of the interactions between different level of the tree, which can become overwhelming if being utilized for a simpler implementation.

3 Project Management

3.1 Agile Workflow

Iteration stands as a fundamental principle within game development processes [21]. Abrahamsson and Ronkainen [22] explore the philosophy behind the principles of the agile manifesto [23], displaying how it revolves around iterative development, with a focal attention towards the following areas: [22] strong interactivity between teams; simplifying the work to be done; adapting to changing requirements and applying those changes incrementally.

Although not all these apply to this project specifically, these core characteristics of agile, encourage its suitability for the process of game development. Despite the emphasis on cohesion and co-operation between team-members, principles concerning this can be disregarded considering this is an individual project.

Agile approaches aim to “deliver working software frequently” [23] avoiding unnecessary complexity. Encouraging a culture of continuous feedback and improvement, with each iteration allowing for the opportunity to refine the product over time. Applying this to a shorter timescale [23], allows me to be able to make necessary adjustments early in the development process. With the potential for uncertainty and unknowns throughout the projects’ timeline - Agile processes embrace the possibility of change, avoiding investing time on features that may not be used. Offering a flexible approach that “welcome changing requirements” [23], readily accommodating variations in project scope and priorities.

Approaches with the Waterfall concept were considered, which traditionally follow a more linear progression from one phase to the next. While this structure is effective in organizing the projects progression, with each phase having to be completed before moving onto the next, it becomes difficult to accommodate for changing circumstances - particularly within the short timescale available.

3.1.1 Scrum

Agile offers a plethora of methodologies such as Scrum, Kanban, and Extreme Programming (XP), each tailored to address specific needs and project requirements. [24]

Scrum was opted for over others due to its structured approach to project management, which aligns well with the iterative and adaptive nature of game development. The structured framework in Scrum is ideal in organizing the projects workload where self-accountability is encouraged. Its flexibility and adaptive nature allow for continuous improvement through concepts such as retrospectives and reviews

[25], where we can re-evaluate the priorities of the remaining time available, as well as understand what went well and can be improved. Other key features of Scrum involve estimating tasks to understand their complexity, as well as to aid in planning the tasks to be done.

Team-related concepts of Scrum such as daily standups, retrospectives and reviews were not followed as they were deemed unnecessary. An adaptation of this was applied where personal reflections were considered each iteration to improve self-productivity.

Kanban, another agile approach has all its tasks and features for the project set out. Its goal, more aligned towards working on the most prioritized task at any time, eventually completing all the defined jobs. [26] This varied to the sprint structure defined within Scrum which generally provides a more organized view, easily visualizing what needs to get done within a time period. Although Kanban is a viable approach, a time-based structure like Scrum better suits the constraints of this project.

3.1.2 Sprints

Sprints are a core concept in agile methodologies but more specifically, in Scrum - breaking down the development process into feasible chunks of time, typically lasting up to four weeks [26]. In this project, these sprints will generally take groups of related tasks or features that build on the previous iteration. Aligning with agiles' principles of delivering functional improvements, we focus on collating features that are set to add value or a tangible improvement to the game – the goal being we have a playable / improved portion of the game every sprint. This is further reinforced through the practice of continuous integration and delivery (refer to 3.2), ensuring that code changes are integrated and tested with every sprint.

Working in sprints allow us to be flexible when required. If certain features or mechanics aren't working as expected, they will be adjusted or removed in subsequent sprints. Similarly, if new ideas or features emerge, they can be incorporated to future sprints accordingly. This structure will reduce the likelihood of major problems later on, ultimately leading to a more polished and successful game.

The implementation phase of the project was organized across an eight-week period, structured into two-week sprints. The length of each iteration cycle ensured sufficient time to deliver working software by the end of the sprint whilst also allowing for possible refinement. Sprint planning was conducted shortly before each block – considering tasks that were left over from previous blocks. Appendix A illustrates how sprints in the development cycle were laid out.

3.1.3 Trello

Trello is a visual tool that can be used to manage any type of project, workflow, or task tracking [27]. Trello became essential in applying and visualising my workflow. Boards represented each sprint, where tasks were broken down into manageable chunks and prioritized accordingly. These boards were structured to reflect the key elements of Scrum with distinct columns representing backlog, sprint planning, in-progress tasks, and completed items. Each task was represented as a card which I could move across the aforementioned columns Features like labels, due dates, and comments gave me an outlook which I could use to relay information for future sprints. Making use of a backlog meant tasks that weren't completed within the sprint could be added back to the top of the backlog, prioritizing their completion before others. Appendix A visualises these sprints as Trello boards.

3.2 Version Control

It was important to consider revisiting or referencing previous changes. Consequently, it was essential to use version control systems to effectively manage and track changes made to the game's codebase and assets throughout the iterative development cycles.

Git offers a robust and flexible platform, known for its reliable backup system. Data loss can be prevented by regularly committing changes and pushing them to a remote repository. Git's versioning and history features allow for tracking the evolution of the game over time, providing a medium to revert to previous versions if necessary. This became apparent when experimenting with a dynamic camera feature that led to problems with jittering. Reverting to a previous stable version was seamless, avoiding setback and preserving the progress made. For better practice, branches were created whenever a new feature was being introduced, acting as a safety net, separating any unfinished code from the main game.

Using GitHub, which hosts the Git repositories, there is access to many workflows through GitHub Actions which are essentially sets of automated processes that can be triggered upon various events. Among these Continuous Integration (CI) and Continuous Deployment (CD) proved to be crucial for the stability and robustness of this project. With CI, was able to ensure that every code change pushed to the repository is automatically tested, allowing for early detection of bugs, and ensuring the overall integrity of the codebase. Additionally, CD automated the deployment process, following the Agile manifesto to seamlessly deliver an updated functional game at the end of each sprint. Appendix B illustrates the CI/CD workflows in GitHub Actions.

4 Design and The Game

4.1 Product Design

A game design document (GDD) is fundamental to game design serving as a blueprint for the game itself – outlining its concepts, mechanics, story and more. Essential details such as the targeted platform(s) and the intended audience will provide crucial insights into the game's market positioning and development requirements.

Documenting our ideas, requirements and expectations will help convey this information effectively, establishing a clear vision for the development team - ensuring that everyone is on the same page regarding the game's direction and goals. Although this project be worked on as an individual developer, a GDD will be an important reference guide in remaining focused on the goals at hand.

4.1.1 Game Design Document

A well-crafted GDD will provide a roadmap to work more effectively and efficiently, with a clear understanding of what needs to be implemented and how different elements of the game should function. Not only does this avoid potential development issues like "scope creep", but also makes dividing tasks into sprints (refer to 3.1.2) much easier to comprehend, with prioritized features established by the GDD. The format and organization of the GDD of this project has been heavily shaped by studying the GDDs of commercial successes such as Diablo [28] and Grand Theft Auto [29], aiming to adopt their effective practices.

The full example of this document can be found in Appendix C. The general structure entails the following sections: Overview, Foundational specifications, Story and Narrative, Gameplay, Aesthetic Design.

4.1.2 Minimal Viable Product

The following section will outline the Minimal Viable Product (MVP) for the game described by the GDD in Appendix C. This will serve as the foundational and minimal version of the game that will encapsulate all the requirements for this project.

The MVP represents the core elements and functionality necessary to deliver a playable and enjoyable experience to players whilst also serving as a framework for future development and expansion. This is readily apparent in additional features available outside of the MVP, for example, repair robots or random storm events.

In the context of “Mission Spudnik: Grow or Die”, the MVP will include essential feature such as resource gathering, robot management, and progression mechanics such as research and upgrades. This will provide players with a solid foundation for gathering and managing resources whilst ultimately working towards the end goal of building a spacecraft to escape the hazardous planet, as specified in the “Escape” mechanic of the “Gameplay” section of the GDD. While the GDD paints a picture of the game’s potential, the MVP offers a grounded perspective which will also be simpler to refine iteratively based on player feedback. Hence, sprints will prioritize assigning requirements within the MVP before other features within the GDD.

In specific, the minimal requirements involve:

Table 1: *Functional Requirements of the MVP*

ID	Functional Requirement
R1	30+ minutes of gameplay
R2	Deploy at least 3 types of robots
R3	Assign robots to different jobs
R4	Players and robots can collect potatoes and scrap
R5	Robots can plant and harvest potatoes
R6	Research upgrades for robots
R7	Research upgrades for resources
R8	The game can be won by building a rocket
R9	The game can be lost by running out of potatoes or player cannot escape in time

Table 2*: Non-functional Requirements for the MVP*

ID	Non-functional Requirement
R10	Compatible with Windows and Linux on PC
R11	User Interface (UI) should be intuitive and user-friendly

4.2 Software Design

4.2.1 Game Engine Selection

Choosing Godot for this indie project was a natural choice, given its reputation for being a lightweight, free to use engine, compared to the more established but heavier options like Unity. With dedicated engines for both 2D and 3D development, there would be no concern for confusing overlap in the learning process. The main contributing factor coincided with Godot’s specialization in 2D game development, aligning perfectly with the top-down 2D pixel art style outlined in the GDD. With their pixel-based measurement system in particular being praised by Fat Gem Games co-founder Shane Sicienski [30] – providing an edge when it comes to creating pixel art games.

Despite still being up-and-coming, Godot's recent surge in popularity in the indie game market ranked as the third most used engine on itch.io [31], proving it was a solid foundation to build this project on.

4.2.2 Architectural Design Pattern

A Hierarchical architecture pattern was deemed appropriate for this project. This generally comprises of multiple levels of components connected in such ways in which to reflect the control and communication relationships between them [32]. With a tree like structure, the higher nodes representing parents and the lower nodes representing children [32]. Appendix D visualizes this architecture.

Entity Component Systems were also considered; however, they are not inherently part of Godots' architecture or design philosophy. Godot follows a different paradigm based on scenes made up of nodes, each of which can contain properties, scripts, or further child nodes. Considering this, the decision choice of a hierarchical architecture pattern in Godot lends itself well in accommodating building complex game objects and systems through composition and inheritance.

By structuring my game's elements into a hierarchical scene tree, it can be easily translated into the codebase. Firstly, Godots' scene tree inherently provides a natural way to organize the different components of the game, whether that's entities, environments, or user interfaces. This hierarchical organization not only makes it easier to manage and navigate the scene tree as its complexity increases but also promotes modularity and code reusability. For instance, I'm able to encapsulate specific functionalities into individual nodes, making it straightforward to re-use them across different scenes or instances.

Godot continues to encourage this hierarchical structure with its built-in signalling system. With nodes enveloping a parent-child relationship, signals provide a facet of seamless communication and interactivity between nodes. This could be crucial for the properties of agent-based ai where communication will be detrimental to its logic. Additionally, Godots' support for dynamic instantiation and manipulation of nodes at runtime aligns perfectly with the demands of the game, where it will be an integral feature to actively generate instances, for example, newly harvested potatoes being added as a child of a larger container of potatoes.

There will also be situations where nodes or scenes need to be accessible from any part of the game, and this can get tedious if there is a need to pass references between them. The Singleton design pattern alleviates this problem, ensuring only one instance of this node or scene will exist throughout the entire runtime of the application.

Godot makes this possible with its autoload feature, allowing singleton instances to act as centralized managers for those functionalities, for example, the functionality for a pause menu can be isolated within a single scene extending its modularity. This should all however be used cautiously and appropriately, as it can introduce strong dependencies between components which may reduce testability in the future [33].

4.2.3 Agent-Based AI

With the inherent hierarchical structure apparent in Behaviour Trees, this choice would seamlessly complement the previous design choices made in 4.2. With this in mind, we can benefit from the complex adaptive behaviours associated with behaviour trees, without needing to concern ourselves with the restructuring required in integrating this tree-like structure into the codebase. Contributing to the general maintainability of this project, its compatibility with Godot's structure, ensure any future behaviour modifications can be easily incorporated.

4.3 The Game

4.3.1 Basic Game Mechanics

The foundational elements of the game mechanics involved a player character, complete with movement controls and collisions. A rocky map layout was designed, with a central hub providing a framework for player exploration and interaction.

People can engage with resource-related mechanisms described in the "Gameplay" section of the GDD (see Appendix C), specifically R4 & R5 of the MVP which involves harvesting potatoes and collecting scrap. Potatoes were programmed to grow on designated farming plots, while scrap items were set to randomly spawn in a predetermined portion of the map at set time intervals.

4.3.2 UI and Tech Tree System

The user interface (UI) was vital in enhancing the user experience (R11 of the MVP) of the game. Developing features such as a day night cycle allowed to introduce the concept of time in the game which was necessary for establishing the losing condition (R9 of the MVP) described by "The Storm" section of our GDD. Clear feedback on game progression was provided through displaying key information such as resource counts and upkeep, aiming to streamline player interaction.

The Tech Tree system was a primary feature of the game given R6 & R7 of the MVP. Using the hierarchical structure, (refer to 4.2.2) the different forms of the Tech Tree were easily split into separate workbenches, with each workbench sharing a common composition

only differing in the associated UI to be displayed. To aid the player in their decision-making, a predicted upkeep feature helped provide insights into the future maintenance costs of any purchases made.

Maintaining safe synchronization was applied to frequently referred notions such as time management, global values like potato count or upkeep, interaction management, and also pausing and resuming gameplay - were all influenced by the design choice of the singleton pattern (refer to 4.2.2)

4.3.3 Agent Behaviour and Assignment

Robotic entities and their agent-based AI was a pivotal aspect of the game as it constituted a significant portion of the game's core functionality, covering R6 & R7 of the MVP.

With the "Robot Types" section of the GDD detailing 5 different robot types, the "Planter", "Harvester" and "Salvager" were prioritized to align with R2 of the MVP, trying to accommodate a whole game experience. This was extended to accommodate the "Collector" robot as it shared similar interactions with the existing "Scavenger" robot. This solidified the simulation aspect that the agents were trying to emulate, amplifying the automation elements of the game. Not only did this offer a great benefactor for player engagement but also directly reinforced the *feedback and simulation* core mechanic (see 2.1.1).

In regard to the behaviour of the agents, I utilized a plugin named Beehave [34] for implementing the behaviour trees, designed for specific use with Godot. With Beehave covering the entire logic specified in 2.5.3, the process of creating different conditions and actions was straightforward – making these components relatively versatile in order to allow them to be re-used across different behaviours. Throughout the implementation process, debugging was a crucial aspect of creating these behaviours considering they were operating without any user input. Beehave's real-time visualization feature allowed me to analyse the behaviour tree execution (refer to Appendix E), aiding in the identification and resolution of unexpected issues. With the AI logic being assigned as a child of its robot, it was able to benefit from the present hierarchical structure, inheriting all of its parent's components, such as navigation or audio players – seamlessly integrating a robots decision making with its functionality.

Interacting with the robots was a crucial feature as suggested by R3 of the MVP. Allowing players to assign them to specific plots or areas cemented the robots as functional and responsive agents. This drastically improved the level of player involvement whilst developing the strategic dimension of the game. This was enhanced visually by

adding an illuminated circle around the selected robot, coupled with an information panel to display status updates of the selected robot.

Idle areas were introduced as resting areas for the robots to return to when inactive. These were further developed into visual indicators of the number of robots present in each plot, feeding more information to the player to further deepen their *strategy and decision-making*.

4.3.4 Balance and Ending Conditions

With numerous interacting components in the game, it was necessary to balance these. This meant fine-tuning the different properties of robots, or the costs and effects of certain upgrades within the Tech Tree System so that they fit together to promote an enjoyable and engaging experience for the player. This was a significant part of the game itself as it strongly correlated with three of our core gameplay mechanics being: *resource dynamics*, *strategy and decision-making,* and *Customization and Progression*.

This is also tied into R8 and R9 of our MVP, where balancing the game would lead to suitable win and lose scenarios. A Rocket Pad was introduced to address the win scenario, whereas Upkeep addressed the lose scenario. Balancing was also key in determining the game length which also acted as an additional scenario for losing the game. This was set to 20 sols, lasting a total of 40 minutes, meeting the 30+ minutes required by R1 of our MVP without becoming repetitive.

4.3.5 Asset Management

In managing assets for this project, both freely available resources and personally created assets were utilized, balancing between time-cost and flexibility. I opted to use free sound and font related resources, primarily because creating instances of these, required specialized skills and equipment that I didn’t have access to. On top of this, there was no need to compromise on the quality, given the wide availability of resources that fit the requirements. These were appropriately accredited in the main menu, accounting for the different licencing methods mentioned on their pages.

When it came to visual assets, free assets did not always align with the chosen artistic style. Hence, I opted to create my own using Aseprite [35]. This alleviated any possibilities of inconsistencies from varying themes or size requirements. Aseprite was much better suited for this project over more well-known graphics editors such as Adobe Photoshop or GIMP, due to its focus on 2D pixel art. Ranked the highest in Slants latest options for the best pixel art / sprite editors [36] made it an ideal choice considering the pixel art style outlined in the GDD.

5 Evaluation

An evaluation was undertaken to gauge the implementation's effectiveness in achieving the project objectives. Simultaneously this process served as an initiative to discover areas of improvement, identify underlying issues, and generally allow me to gain insights into enhancing the overall project management process.

The following section is divided into two segments with the initial segment focusing on verification. This revolves around determining whether it complies with the set requirements or specifications [37]. The final segment will concentrate on validation, assessing whether the implementation is fit for its intended purpose and its suitability in meeting the needs [37] of the project's audience.

5.1 Verification

In the context of this project, determining this verification is entirely dependent with its compliance of the requirements defined by the MVP in 4.1.2. To establish this, multiple forms of testing are conducted to ensure the game's features and operations behave as intended, examining the implementation's code and functionalities against these predefined requirements.

The following sections delve into the specifics of these testing methodologies, including unit testing and manual testing. Combining multiple approaches achieves a high level of confidence in the functional quality and reliability of the game, each playing a crucial role in verifying the implementation's conformity to the project's objectives and MVP specifications.

5.1.1 Unit Testing

With the extensive workload involved in Manual Testing and User Testing, it is evident that automation can reduce this burden on both developers and human testers [38]. Unit testing was prioritized amongst others due to its granular approach benefitting in validating the requirements mentioned in the MVP. Alongside this, this form of testing seamlessly integrated with established Godot testing frameworks, such as GUT [37] or GdUnit4 [38].

Using predefined sequence of actions, we were able to check against expected outputs or results in our classes and methods. Testing individual components through automation allowed me to reliably ensure correct behaviour for scenarios in which otherwise would be tiresome or difficult to recreate. The automation of these tests also provides the security of identifying subtle defects that may be easily missed out on if manually evaluated.

GdUnit4, being a testing framework designed specifically for the latest version of Godot, allowed for easily writing tests in GDScript itself. Its extensive assertion framework aided in establishing a suite of simple tests that covered the methods involved in the requirements. Executing these tests through the testing tool provided detailed information regarding any test failures of errors encountered (refer to Appendix F). This not only allowed for efficient identification and debugging of issues within the game code but also benefitted the test writing process itself.

Although it would be ideal to procure some form of automatic testing for all logic in the game, this isn't always entirely possible. Project constraints such as deadlines were to be considered. Alongside this, situations where units of code rely on external components may present challenges that hinder the feasibility of achieving complete test coverage through unit testing. With these considerations in mind, unit tests were primarily focused on validating the methods involved in meeting the functional requirements outlined in the MVP. For requirements unsuited for automated testing, manual testing was employed. Further details on manual testing procedures can be found in Section 5.1.2.

A report detailing the outcomes of implemented unit tests and their integration in CI is available in Appendix F, while the original tests themselves can be found in the "Unit_Tests" folder of the codebase.

5.1.2 Manual Testing

As mentioned previously, it wasn't always possible to verify the implementation through the chosen form of automated testing. Henceforth, Manual Testing was incorporated in addition to this to account for the uncovered game logic. Generally, these scenarios were considered outside the MVP, aimed at verifying supplemental mechanics within the game to ensure they are fully functional in regard to the intended gameplay.

This was integral in replicating gameplay scenarios that required real-time interactions within the game from the player. Sensory cues, for instance, those that are visual like selecting a robot, or an audible cue when picking up items. Manually verifying these interactions ensure that they respond as expected, which would've been difficult and rather impractical to capture in automated tests – especially considering factors such as responsiveness tend to require subjective feedback.

This altered from the approach of the automated tests, with the manual approach aiming to satisfy a seamless gameplay experience, particularly from the player's perspective. On top of this, the flexibility

afforded by manually evaluating the gameplay, allowed for uncovering unexpected issues or bugs that may not have been anticipated when designing the Unit Tests. The main priority of this approach however, remained on fulfilling the MVP requirements missed out by the limitations presented by Unit Testing.

5.1.3 Code Coverage

Code Coverage is a useful metric in assessing the effectiveness of the verification techniques used. However, with Godot being a relatively newly developed game engine, there is a gap in regard to dedicated code coverage tools. Attempts at integrating renditions developed by the community resulted in little success, primarily because the most recent 4.2.1 version of Godot was being used in the project. The lack of available solutions indicates the challenges in using an emerging engine like Godot, perhaps highlighting an oversight in the design process (refer to section 4.2.1). Having this feature would undoubtedly streamline the testing and evaluation process, providing a reliable benchmark in assessing the quality of the project.

Even without relying on code coverage tools, we can still critique how well we've tested our codebase. With the large number of dependencies present throughout various nodes, it’s clear to see how this has hindered the possibility of code coverage through our automated tests. It does feel as though this could’ve been avoided if there was a more iterative approach to testing, where it was implemented earlier and more frequently, being more conscious about introducing new dependencies. If more time was afforded for the project, it would be optimal to develop a more robust testing framework with additional techniques such as Integration Testing. Testing the interaction between nodes would help alleviate the shortcomings in testing from the presence of dependencies, ensuring a more thorough verification of the game’s functionalities.

Beyond the specified requirements, although manual testing comprised the majority of the testing, it still remained a valuable tool with its usage extending to validating the effectiveness of the existing unit tests. It is always important to consider whether we’ve adequately accounted for enough edge cases or failure points in the unit tests. Although this wasn’t conclusive, observing that very few bugs were encountered when performing the manual inspections served as a useful indicator of the reliability of our automated testing procedures.

5.1.4 Completion of Requirements

This section will reference the table located in Appendix G, which assesses how each requirement was tested, along with the outcomes indicating whether they were successfully met or not.

From this evaluation, it's evident we have fulfilled the MVP requirements set out in the design stage of this project, with each requirement being addressed and tested to ensure its compliance.

Alternative verification approaches separate to those aforementioned were also necessary. For instance, validating the functionality and integration of the game across different platforms (such as Linux and Windows) was automated but didn't fall within the scope of traditional unit tests. Instead verifying these builds were directly incorporated into the Continuous Integration pipeline using GitHub Actions which served as a more practical approach.

Moving forward, rigorous testing will be placed at the forefront of the development process to uphold and exceed the standard of reliability established throughout this project.

While not every detail outlined in the GDD made its way into the final product, the game has exceeded expectations in regard to representing the outlined core objectives. Our primary aim was to create a resource management game with agent-based AI, and in that regard that has been achieved admirably. This does however extend beyond mere functionality. The following section on Validation explores how the game mechanics and agent behaviour collectively impact the gameplay experience.

5.2 Validation

This phase of the evaluation will scrutinize the games' alignment with the experience of the target audience described in the GDD. This will delve into various facets of user interaction and satisfaction particularly emphasizing player enjoyment and their interactions with the agent-based AI aspect of the game.

To determine this, a series of playtesting sessions were conducted with viable candidates in order to gather first hand feedback and observations regarding the players' experiences with the game.

Closely observing player interactions bolstered verification, allowing for identifying areas of strength and areas for improvement. This extends to validation and the player experience, ultimately deducing possible refinements to enhance the overall experience and ensure it resonates authentically with the target audience. Exploring this further with direct feedback through asking follow-up questions, these

sessions aimed to validate the game against the envisioned experience outlined in the GDD.

In order to uphold the integrity of this evaluation process, when engaging with real participants, it is inherent to address the ethical considerations concerning the study.

5.2.1 Ethics Considerations

Ethical approval from the University of York Computer Science Department was required before proceeding with the user study. In completing this application, I was able to re-visit the aim of this project where I could better determine the conditions and expectations of what I wanted to achieve from this evaluation process.

Many ethical considerations were taken into account, with a considerable effort in prioritizing the well-being and rights of participants. Concerns of data privacy were alleviated with explicit details of what data was to be collected and how this would be stored and used. Ensuring full anonymity within collected data was crucial in backing this up, with minimal demographic information being gathered beforehand. Psychological safety and risks were amongst other factors that were considered although this was established to be of minimal threat to participants.

An information sheet and informed consent form were produced to effectively communicate all relevant details to the involved participants, ensuring transparency and ample opportunity for clarification both before and after the study. Maintaining these open channels of communication was necessary in enabling participants to voice concerns and withdraw from the study if needed, without facing any consequences.

5.2.2 Participant Recruitment

Once approval was received from the University, I was able to continue on to recruiting participants for playtesting. The target audience specified in the GDD mentioned young adults (13+), although with ethical concerns in mind and difficulties with garnering such an audience, participants aged over 18 were enlisted. Through student communication channels, participants within the computer science cohort were targeted.

Through student communication channels, participants within the computer science cohort were primarily targeted. Initially, I sought to recruit individuals who were passionate about automation and gaming, aiming to engage both hardcore and casual gamers alike. However, I encountered difficulties in finding a diverse audience, as the cohort predominantly consisted of more experienced gamers. In retrospect, I

recognize the importance of diversifying my outreach strategies, potentially by engaging with alternative communities or utilizing multiple platforms such as online gaming forums.

From these efforts I was able to accumulate a relatively modest sample size of five participants. Whilst acknowledging that this may not be extensive, it was deemed sufficient to evaluate the game's overall player experience and achieve a representative view of its enjoyability.

5.2.3 Overview of the study

This study was structured to help in evaluating the user experience and effectiveness of an interactive resource management simulation game featuring agent-based AI. Participants would be asked to play the game in which they would have the opportunity to explore its mechanics, interact with AI agents, and manage resources to accomplish in-game goals.

During the gameplay sessions, qualitative data was collected through observations and user feedback through a post-play survey. I was able to assess participants' perceptions of the game's usability, engagement, and overall experience, as well as their satisfaction with the resource management mechanics and AI interactions.

5.2.4 Observations

As described in 6.2, direct observations were a major aspect of these playtesting sessions. From the five participants that were monitored, it became apparent the different decision making and strategic behaviours they had in regard to the various mechanics and scenarios. Key observations can be found in Appendix H.

It was made clear how there are diverse ways in which players approach challenges, and generally express their own creativity within the game world. Where some players may generally take time to note the differences between the various upgrades, others would be hasty with these decisions, and it was apparent how these different approaches would reflect in the outcome of their gameplay. Similarly, there was a key difference in how optimising the robot composition in each plot would prove to be beneficial in comparison to paying less attention to this. These observations reflected how crucial balancing the game was in terms of managing the intended difficulty and feasibility of the gameplay. This also did not fail to highlight the forefront importance of *strategy and decision making* (see 2.1.1).

Another highlight from observing these sessions came from being able to capture real-time reactions and emotional responses. Witnessing the players' enjoyment of watching their robots in action offered a

glimpse of how it may have influenced their satisfaction with that feature of the game. Although it's likely this will be further elaborated through the post-survey questions, it still presented a different dimension that gave an alternative outlook to their experience.

These sessions provided useful perspectives in identifying bugs and exploits that weren't as apparent during individual testing in 6.1. From minor inconveniences like misspelt words to more concerning issues like prices of purchases not matching their displayed costs. It exposed exploits like how players would collect scrap using their player character in the early stages of the game. Considering this wasn't the intended way to collect scrap, it was proposed that an oxygen meter would be put in place to restrict the player to leaving the hub for an extended period of time. Although due to time constraints within the project timeline, this wasn't implemented for the final game but is something that would be prioritised following the conclusion of the project.

5.2.5 Post-play survey

Gathering user feedback is equally as important as recording these observations. Appendix I displays a post-play survey which acted as a medium in reinforcing the insights gained from observing. With the information acquired, I was able to better gauge what worked well, what didn't, and what improvements were needed, with focused queries in mind to pinpoint those concerning areas.

Generally, the user interface and controls were well received, though there were common inconveniences that players encountered such as the short delay between buying robots. Many players recommended removing or shortening this timer, as it disrupted the flow of gameplay and felt unnecessary. While initially intended to slow down the pace of the game, upon reflection, this feature was removed as it became clear how this design choice did not align with the game's quick tempo. Another frequently mentioned issue regarding the user interface related to navigating between the scrap and potato menus within the various shops and research stations. Lack of information was apparent, hence addressing this by clarifying its accessibility in the provided info section and making the buttons themselves much more prominent. Refer to Appendix J for an outlook on the changes made.

Despite these challenges, players praised several aspects of the game. The top-down aesthetic and audio design in particular complemented the game's intensity, further contributing to their immersion in the gameplay. Players appreciated resource mechanics such as the predicted upkeep when making a purchase, as it provided valuable information that allowed them to plan their decision making towards managing their economy. Stemming from the core principles

in 2.1.1, this design choice positively aligned with the players' expectations of *strategic thinking* when allocating their resources.

The feedback regarding the robot info panel was particularly insightful. While players found it useful for understanding what the agents were currently doing, they also expressed a desire for further information on the progress of their tasks. This prompted me to incorporate a dynamic progression bar that offered visual feedback on the completion status of its task at hand. Being unable to select multiple robots at once was another frequently mentioned issue – and while it was considered a high-priority improvement, time constraints ultimately prevented it from being included in the final game.

Figure 4: *Dynamic progression bars to show completion status.*

The agent-based ai had a larger impact than anticipated with several mentions with how it positively enhanced their gaming experience. Feedback showed that players appreciated how the agents would autonomously carry out tasks without the need for constant intervention, allowing them to focus on finding the optimal balance instead of micromanaging every aspect. One player even noted the addictive nature of the game, describing the buzz that came from automating potato production, experimenting with different agents, subsequently upgrading them to facilitate further expansion.

To complement this qualitative feedback, there were several questions which necessitated Likert scale responses that offered a means of capturing opinions through quantitative data. With these questions being focused on concepts like user-friendliness, difficulty, effectiveness of the agents and the overall enjoyment factor itself - I was able to garner a representative overview of the strengths and weaknesses in the game design.

The feedback portrayed in Figure 1 gave a broader understanding of the User experience, highlighting aspects such as using the game controls, navigating the user interface, and game difficulty. While the feedback may vary between positive and negative, it pinpoints that players generally encountered challenges or struggled to engage effectively with the user interface. It was apparent that this area is where improvements were most needed, prioritizing this for the remaining timeframe of the project.

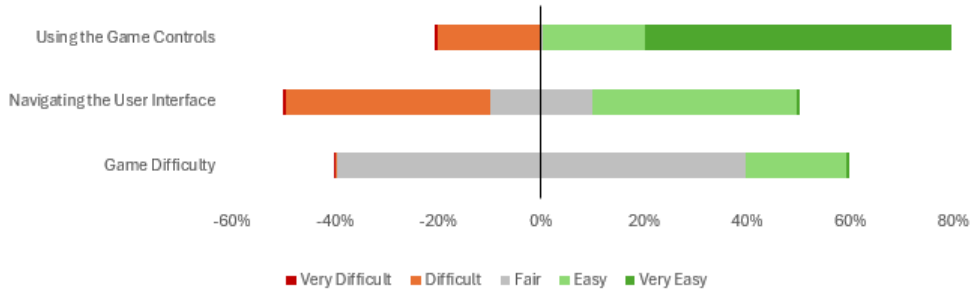


Figure 5: *User Feedback Survey: Evaluating Game Experience Across Controls, UI, and Difficulty*

The overwhelmingly positive feedback depicted in Figure 2 directly correlates with one of the major aims of the project, which was to create an engaging and enjoyable game experience. More so the enthusiastic response to the agents align perfectly with my intial motivation behind incorporating them as a major feature of the game. This positive reception only serves to strengthen the effectiveness of the design choices made throughout the development process.

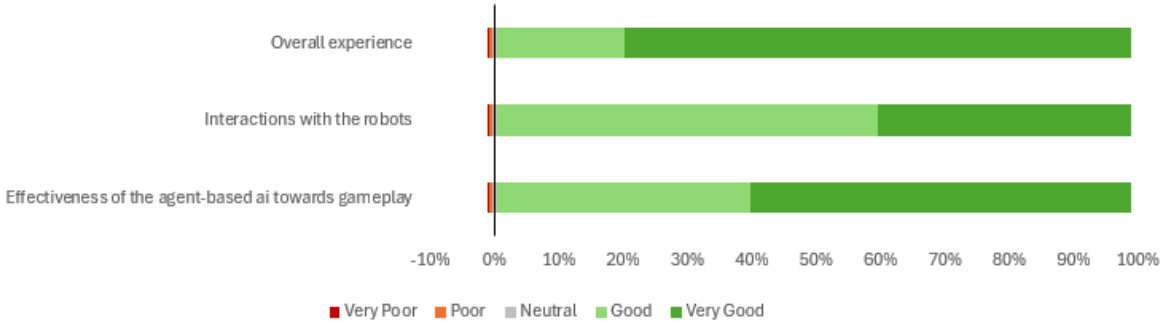


Figure 6: *User Feedback Survey: Evaluating Game Experience across Overall Enjoyment and Interactions with Agent-Based AI*

6 Conclusion

Understanding that games require multiple mechanics for them to be enjoyable, there was an initial concern that the other game mechanics might overshadow the AI elements. However, from all the feedback from the sessions, I've established an alternative perspective. With the focus being on AI behaviour, autonomous intelligent decision-making serves to uplift other mechanics rather than detract from them. This finding was empowered by the overwhelming positive feedback regarding the enjoyment of the game, in particular, the effectiveness of the agent-based AI towards the gameplay.

This project has been enlightening in regard to revealing the transformative potential that AI brings to games. Taking a basic game premise of resource collection, elevating this to another level where players are challenged to account for the decisions made by intelligent entities, adding layers of complexity and depth to the gaming experience.

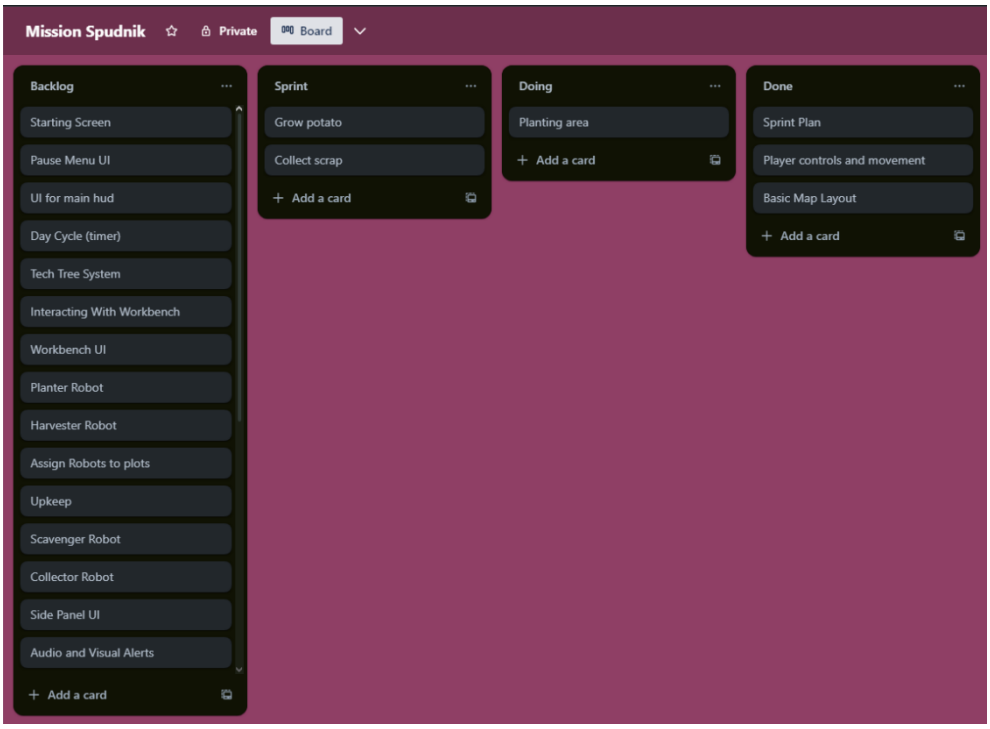
At the start of this project, I aimed to make a game that was enjoyable that showcased intelligent decision making. With the research undertaken, I've gained a deeper understanding of what intelligence in agents is. In addition to exceeding the projects functional requirements, I believe the final game exemplifies sophisticated AI behaviours in the robots, elevating the resource mechanics that were established in the beginning of this project.

There's no denying that the constraints of this project have presented limitations within the capabilities of the developed AI behaviours. If anything, these limitations have highlighted areas for improvement, particularly in transitioning from a single-agent system to a multi-agent system. Moving forward, the next steps would expand the agent's environment to a more complex setting, where they can evaluate a broader range of information and make intelligent decisions accordingly. The experience gained from this project have provided a solid foundation on which I aim to further advance in the realm of AI within gaming.

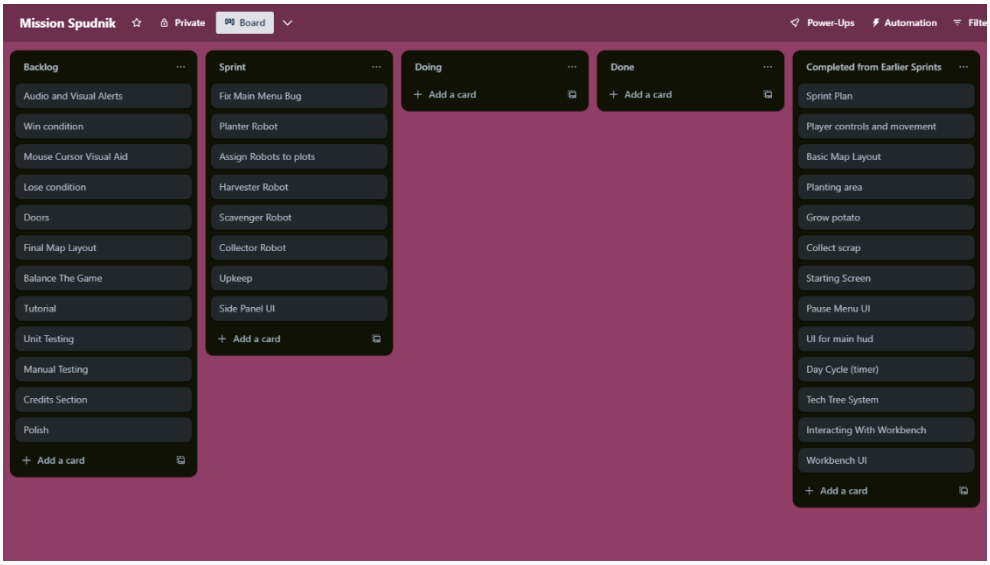
Through this journey it has become clear that AI has the power to not only enhance traditional gameplay mechanics (such as resource management) but also introduce an entirely new dimension of interaction and strategy. This paves the way for future innovations in game design, with AI establishing its position as a principal component in revolutionizing gameplay experiences.

Appendix A

Sprint Trello Board



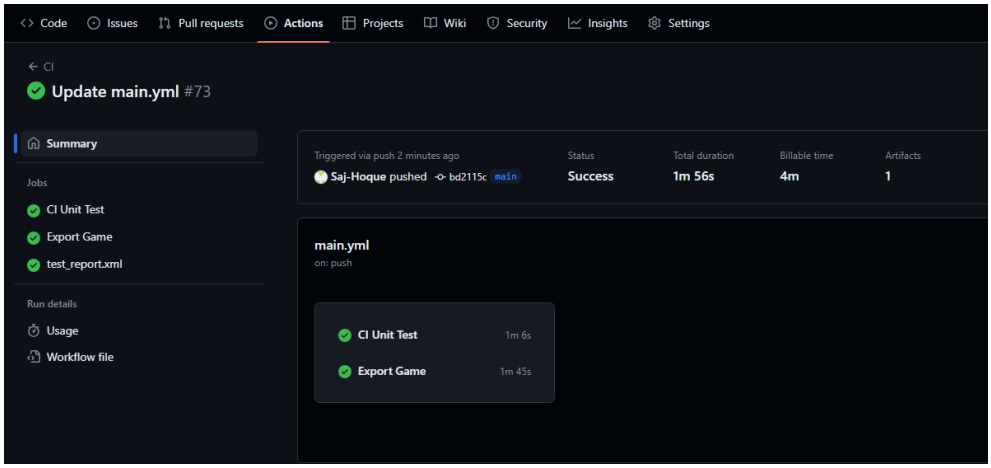
Appendix Figure 1: Initial Layout and Tasks on Trello Board for Sprint 1 in progress.



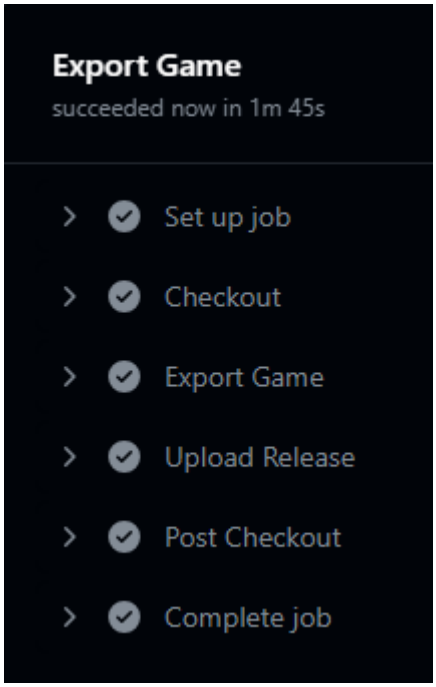
Appendix Figure 2: Tasks on Trello Board for Sprint 3 after Sprint Planning – Bringing incomplete task "Fix Main Menu Bug" from the last sprint into the current sprint for resolution.

Appendix B

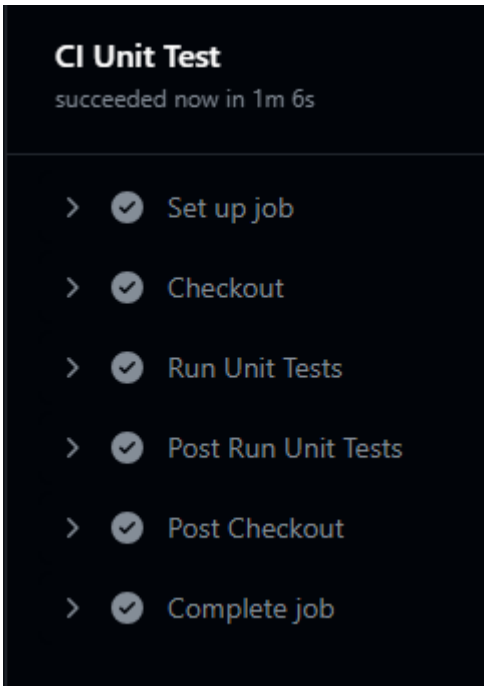
CI/CD Workflows in GitHub Actions



Appendix Figure 3: Full CI/CD Workflow - Validates Unit Tests and Game Build.



Appendix Figure 4: Building Game Workflow



Appendix Figure 5: Unit Test Workflow

Appendix C

Game Design Document

GAME DESIGN DOCUMENT (GDD)

Mission Spudnik: Grow or Die

Overview

"Mission Spudnik: Grow or Die" is a fusion of strategy and resource management with management as its core focus. The player will play as a lone engineer, tasked with managing and overseeing robots (agent-based AI) where the main goal is to grow potatoes and collect scrap. Managing these resources to not only survive but to expand their production through research and upgrades to ultimately construct a rocket and escape the planet.

At the heart of this game is its agent-based AI system, which brings your potato farm to life through the form of automation, with each agent responding to changes in the farm environment. The player will be able to employ robots in order to automate their potato growth, researching upgrades for different type of robots, enhancing their farming techniques and performance, bolstering your potato production. Strategically controlling the variance, positioning and number of robots to best optimise the rate of production whilst supervising the upkeep that is associated with this ensuring they do not run out of resources. The sophisticated AI not only adds depth to the farming experience but also introduces a layer of strategic complexity.

With time being a major factor to consider, survival hinges on the players ability to allocate resources efficiently. With upgrades being essential to progression, they will be the root focus for the player. Although with these facilities coming with an additional constant cost (e.g. increased potato fuel usage) that will hamper the players production, the player will have to consider trade-offs and make important decisions to maintain their survival.

Specifications

Game Title: Mission Spudnik: Grow or Die

Platform: PC (Windows, Linux)

Genre: Resource Management, Real-time Strategy, Simulation

Game Style: Top down player character perspective, Space-themed agriculture simulation

Target Audience: Caters to both casual and hardcore gamers, aimed towards teenagers and young adults (13+)

Intended Duration: 30+ minutes

Controls: Keyboard + Mouse

Story and Narrative

"Mission Spudnik: Grow or Die" is set in a science fiction-based world on the harsh climates of the foreign planet. You find yourself stranded on Mars as part of a secret mission aboard the Soviet Sputnik rocket. Your teams' mission was to test the viability of Martian soil for agricultural purposes. However, your mission takes an unexpected turn when a violent storm destroys your spacecraft, killing your fellow team members and leaving you stranded and alone.

As a skilled robotics engineer, who knows very little about botany, your mission is to quickly hone your planting skills and leverage your prior experience of automation to find a way to escape back home quickly before the storm consumes you!

With time running out you must learn to harness the power of the potato, construct a plant-powered empire and fuel your expedition home. With the goal of returning home to Earth driving you forward, you embark on a daring mission to launch the homemade rocket and defy the odds of interplanetary survival.

Will you succeed in your quest to escape? Survival hinges on the simple yet *starch* choice... ***Grow*** or ***Die***.

Gameplay

Resources

Potatoes serve as the primary resource in "Mission Spudnik: Grow or Die" playing a pivotal role in the players' survival and progression throughout the game. As a versatile crop, potatoes fulfil various functions essential for success. Grown in the planting areas of the space hub, this resource must be maintained throughout the game, with potatoes being utilized as a fuel source for upkeeping the managed robots, as well as the food source of the player character itself – This usage value is likely to be updated along the games' timeline through player decisions and hence it is integral to keep this resource count above this threshold. Strategy will be important when balancing between managing these limits and researching upgrades to bolster production.

Scrap is a valuable resource serving as a key component for building and repairing structures (i.e. extra plotting areas) within the space hub and crafting additional robots, with this resource being major for progression in the game. This material is salvaged, scattered across the Martian landscape. Although this may be easy to retrieve nearer the base, this becomes more difficult to accumulate as this limited resource will take time to replenish, prompting the player to automate this process through robots.

Robot Types

Robots should be able to perform their tasks considering the changes in the environment around them representing their agent-based characteristics. Ideally, they will avoid performing the **same** job as another of the same robot type in the area (For instance, two planter robots will not try and plant on the same plot)

Planter – Designed to plant potatoes onto empty planting areas. Unable to harvest or carry potatoes.

Harvester- Solely harvests potatoes from planting areas with **fully** cultivated potatoes. Unable to plant or carry potatoes.

Collector – Specialised in storage capacity. These can be assigned to plots, to retrieve any harvested potatoes in the area.

Salvager – Trained in speed, these robots are deployed to locate and retrieve loose scrap in the vast Martian lands.

Repair – Ensures robots and space hub are restored from possible degradation. (Through wear and tear or storm events)

Job Assignment

In "Mission Spudnik: Grow or Die", players have the ability to strategically assign various robots under their command to the different plotting areas within the space hub in order to optimize their colony's efficiency. Players can easily manage and direct their robotic workforce by left clicking the desired robot and right clicking the appropriate working area. The assignment of robots is not static; therefore, players have the flexibility to dynamically adjust this based on changing circumstances or player priorities.

Tech-tree

Players have access to a comprehensive tech tree that allows them to research and unlock upgrades for both their robotic workforce and upgrades for soil management and agricultural facilities. This tech tree serves as a progression system, enabling players to customize and enhance their colony's capabilities in order to advance through the game.

A variety of upgrades for the different types of robots available in the game allow players to tailor the composition of their robotic workforce to suit their specific needs and playstyle. These upgrades may include efficiency, capacity, speed, or power consumption – all offering a wide range of customizability to suit their strategic objectives.

Allowing players to improve potato yields and optimize farming efficiency provide an alternate dimension through which they are able to expand their production rates. Examples include Soil enrichment like fertilizers or Irrigation Systems, for consistent water supply – factors to enhance potato growth and yield.

With multiple levels for each upgrade category, this can be extended to also account for the late game.

"The Storm"

From the beginning of the game, the player is informed with the knowledge that a devastating storm will strike the Martian colony in 20 sols (Martian days), which serves as a looming threat that adds a sense of urgency and challenge to the players' experience. Hence, the player must carefully manage their time and resources to achieve their long-term goals of survival and escape. With only 20 sols until the storms' arrival, players face a pressing time constraint that forces them to upgrade their technology, encouraging progression within the game.

The storm is not just a game ending feature but also a dynamic gameplay element. As the storm approaches, its intensity increases, with random events occurring that can degrade the conditions of the space hub and associated robots.

Escape

The ultimate winning objective is for the player to construct a spacecraft "The Spudnik", which will allow the player to escape the planet before the storm strikes and destroys the space hub and everything in its vicinity. This requires the player to gather a significant quantity of resources, potatoes, and scrap to build this spacecraft which is only possible through strategic progression. The games' progression system is carefully designed to guide players towards this goal in incorporating upgrades from the tech-tree. This is imposed through the addition of a time constraint, motivating players to stay focused and proactive in their efforts to achieve this long-term objective whilst balancing short term survival needs.

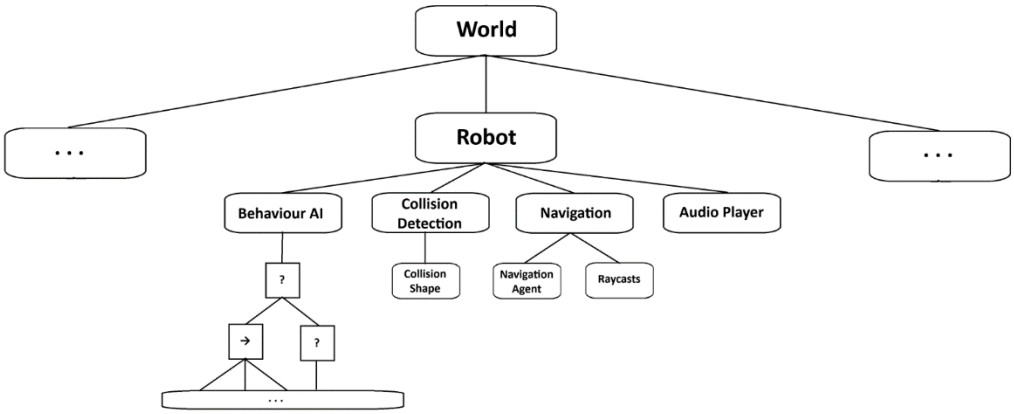
Audiovisual Design

"Mission Spudnik: Grow or Die" will adopt a top-down 2D pixel art style presenting a Martian landscape with sci-fi aesthetics. Environmental focus situated around the space hub, as majority of the game is represented indoors with vibrant and dynamic visuals to portray potato growth in planting areas.

Subtle background music to enhance the atmosphere whilst keeping the player immersed. Audible and Visual feedback will be provided to the players to acknowledge successful actions, i.e. assigning a robot to a task. Dynamic sound effects, such as planting of a potato, will provide auditory cues to further deepen the players connection in the game world.

Appendix D

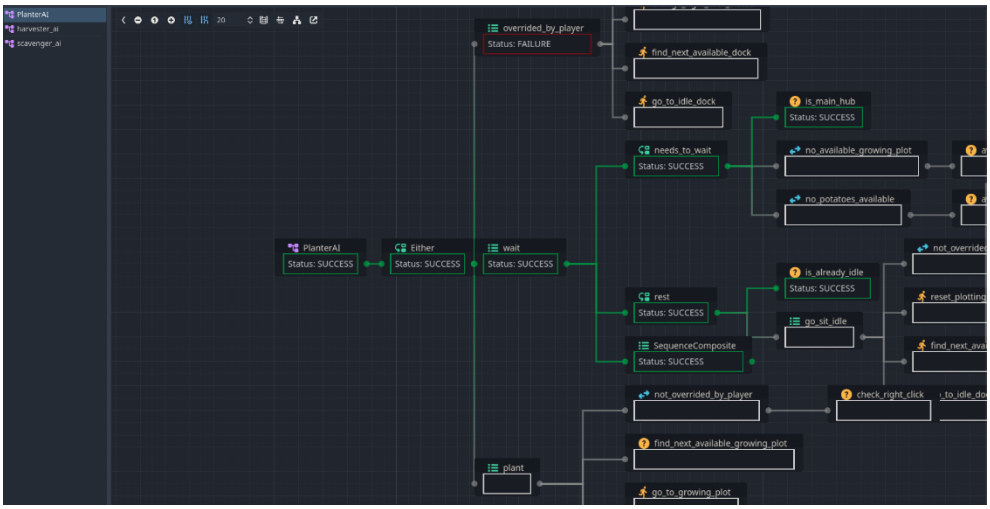
Hierarchical Architecture Example



Appendix Figure 4: Example of Hierarchical Architecture Diagram Demonstrating Game Structure and Component Communication

Appendix E

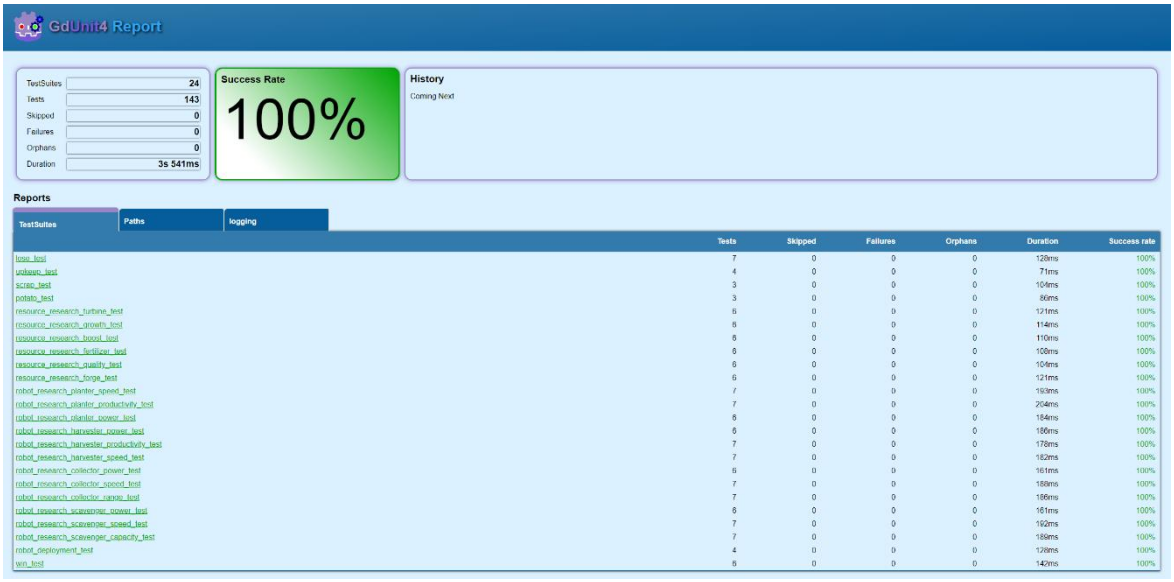
Beehave Debugger View



Appendix Figure 5: Debugger View of Beehave Behaviour Trees

Appendix F

Unit Test Coverage



Appendix Figure 6: GdUnit4 Unit Test Report

Appendix Figure 7: Unit Test 'results.xml' In Github Actions

Appendix G

Evaluation of MVP Requirements

ID	Requirement	Testing Method	Status	Comments
R1	30+ minutes of gameplay	Manual Test	Met	Given that each Sol (in-game day) lasts 2 minutes, the game has a potential maximum of up to 40 minutes based on the 20 Sols set.
R2	Deploy at least 3 types of robots	Unit Test	Met	A total of 4 robots can be deployed from the Robot Shop, with successful tests for Planters, Harvesters, Scavengers and Collectors.
R3	Assign robots to different jobs	Manual Test	Met	Player can left click on robots to select and right click to assign them. Manual tests confirm they cannot be assigned to the wrong type of task.
R4	Players and robots can collect potatoes and scrap	Unit Test & Manual Test	Met	Collector robots and Scavenger robots can collect potatoes and scrap respectively. Players can collect both. Unit tests focus on incrementing the resource count or robot capacity.
R5	Robots can plant and harvest potatoes	Manual Test	Met	Planters and Harvesters plant and harvest potatoes within expected time.
R6	Research upgrades for robots	Unit Test	Met	Both resources have three levels of three distinct upgrades, which enhance spawn rate, quantity, and value.
R7	Research upgrades for resources	Unit Test	Met	Each Robot has three levels of three distinct upgrades, which enhance speed, productivity, and power efficiency.
R8	The game can be won by building a rocket	Unit Test	Met	A launch pad is situated outside the hub, costing the player 25,000 potatoes and scrap to escape and win.
R9	The game can be lost by running out of potatoes or player cannot escape in time	Unit Test	Met	Upkeep is depleted every day, with the game being lost if the potato count < 0. The game is also lost if Sol 21 is reached.
R10	Compatible with Windows and Linux on PC	CI / CD	Met	The CI/CD workflow in GitHub actions exports the game to assess compatibility with both platforms.
R11	User Interface (UI) should be intuitive and user-friendly	Manual Test	Met	A recallable tutorial with visual guidance is provided, alongside information regarding resource counts, upkeep, upgrades, purchases, robot capacity etc.

Appendix Table 1: Discussion of how the MVP Requirements are tested and whether they were fulfilled.

Appendix H

Key Observations from User Study

Key Observations

Some players spent early stages of the game collecting scraps themselves instead of buying scavengers

Players expressed the build button timer was too long

Scrap and potato buttons in menus were not obvious enough, players were either confused or forget how to access the different types of purchases

Players found the predicted upkeep informative

Players naturally are drawn to dragging their mouse to select robots

Tutorial / Info section requires updating with easier to interpret instructions (Visual Aid)

Players that didn't optimize their robot composition often suffered in terms of production rates.

Appendix I

Post-play Survey

Interview Schedule

Pre-Session Questions:

1. **What age range would you say you belong to?**
    a. **18-24**
    b. **25-34**
    c. **35-44**
    d. **45-54**
    e. **55-64**
    f. **65+**
    g. **Prefer not to say.**

2. **What is your level of experience with gaming?**

If there is any experience in **2**, ask the following questions.

3. **What types of games do you typically enjoy playing?**

4. **Have you played resource management games before? If so, could you provide a few examples?**

Post-Session Questions:

User Interface

5. **How did the game controls feel to use?**
    a. **Very easy**
    b. **Easy**
    c. **Neither easy nor difficult (A fair amount of difficulty)**
    d. **Difficult**
    e. **Very Difficult**

6. **Was the game objective clear to you from the beginning?**

7. **Were there any specific game mechanics or features that you found confusing or challenging to use.**

The Game

8. **How did you find the difficulty of the game to be?**
    a. **Very easy**
    b. **Easy**
    c. **Neither easy nor difficult (A fair amount of difficulty)**
    d. **Difficult**
    e. **Very difficult**

9. **Were there any parts of the game that felt unfair or imbalanced?**

10. **Would you suggest adjusting the difficulty or balance of the game?**
    a. **If so, how?**

11. What are your thoughts on the game's graphic and visual style?

12. What was the impact of the sound effects and music towards your gaming experience?

13. Is there anything you would change about the game's audio-visual elements?

14. How did the user interface feel to navigate?
    a. Very easy
    b. Easy
    c. Neither easy nor difficult (A fair feeling)
    d. Difficult
    e. Very difficult

15. Did you find all necessary information readily available on the screen?

16. Were there any UI elements that were particularly helpful or frustrating?
    a. If so, what were they?

17. How would you rate your overall experience playing the game?
    a. Very poor
    b. Poor
    c. Acceptable
    d. Good
    e. Very good

18. What did you enjoy/dislike most about the game?

19. How does this game compare to others you've played in the same genre?

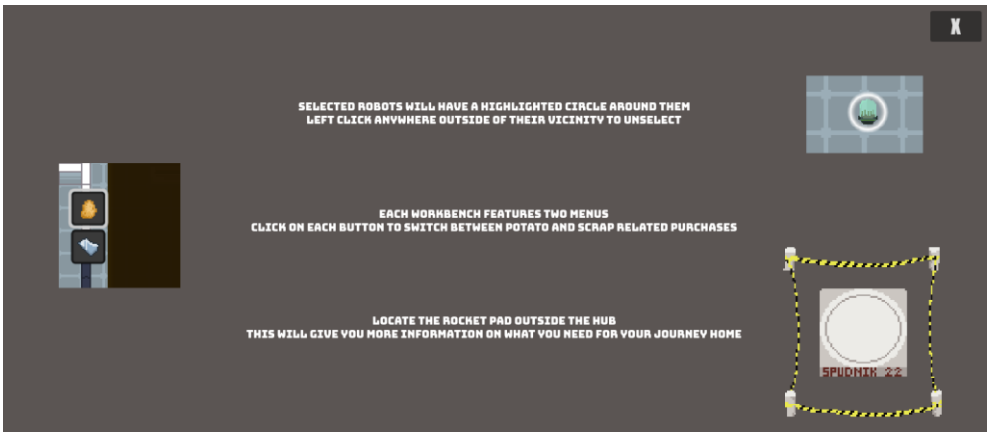
20. How did you find the interactions with the robots in the game?
    a. Very poor
    b. Poor
    c. Acceptable
    d. Good
    e. Very good

21. How would you rate the overall effectiveness of the agent-based ai (robots) towards your gameplay experience?
    a. Very poor
    b. Poor
    c. Acceptable
    d. Good
    e. Very good

22. Do you have any feedback, suggestions, or extra thoughts?

Appendix J

Post-feedback Improvements



Appendix Figure 8: Added visual guidance in the tutorial to clarify any uncertainties.

Appendix Figure 9: Left Old, Right New - Buttons are much more distinguished with a highlighted ring to represent which menu it is on.

Appendix Figure 12: Top Old, Bottom New - Upkeep Alert central and larger, and moved closer to the deadline time (0:00)

Appendix Figure 13: Destroy button on robot info panel was made larger for easier interactability.

Bibliography

[1] S. d. Jong, P. Spronck and N. Roos, “Requirements for resource management game AI,” 2005.

[2] Wube Software, “Factorio,” 2020. [Online]. Available: https://www.factorio.com. [Accessed 24 March 2024].

[3] Concerned Ape, “Stardew Valley,” 2016. [Online]. Available: https://www.stardewvalley.net. [Accessed 24 March 2024].

[4] Statista, “Most popular genres played regularly according to video gamers in the United States in 2022,” July 2022. [Online]. Available: https://www.statista.com/statistics/246766/favorite-video-game-genres-in-the-us/. [Accessed 24 March 2024].

[5] Academy of Interactive Arts & Sciences, “Third Interactive Achievement Awards,” 2000. [Online]. Available: https://www.interactive.org/awards/2000_3rd_awards.asp. [Accessed 2 March 2024].

[6] Ensemble Studios, “Age of Empires,” 1997. [Online]. Available: https://www.ageofempires.com. [Accessed 24 March 2024].

[7] “Getting Started with Age of Empires II: DE - PC,” [Online]. Available: https://www.ageofempires.com/learn-to-play/getting-started-aoe2/. [Accessed 24 March 2024].

[8] B. C. Shelley, Official Strategies and secrets to Microsoft's age of empires II, the age of kings, Sybex, 1999.

[9] Maxis and W. Wright, “SimCity,” 1989. [Online]. Available: www.ea.com/en-gb/games/simcity/simcity. [Accessed 24 March 2024].

[10] M. Pirovano and P. L. Lanzi, “Fuzzy Tactics: A scripting game that leverages fuzzy logic as an engaging game mechanic,” *Expert Systems with Applications,* vol. 41, no. 13, pp. 6029-6038, 2014.

[11] J. Laird and M. Van Lent, “Human-Level AI's Killer Application: Interactive Computer Games,” *AI Magazine,* vol. 22, no. 2, 2001.

[12] Spirit Of The Law, “How the AoE2 AI Thinks (ft. Promi),” 2019. [Online]. Available: https://www.youtube.com/watch?v=-S1CkfzEHSU. [Accessed 24 March 2024].

[13] J. Wiles and P. Sweetser, “Current AI in games : a review,” *Australian Journal of Intelligent Information Processing Systems,* vol. 8, no. 1, pp. 24-42, 2002.

[14] S. Franklin and A. Graesser, “Is it an Agent, or Just a Program?: A Taxonomy for Autonomous Agents,” 1996.

[15] M. Woolridge, An Introduction to MultiAgent Systems - Second Edition, 2009.

[16] J. Finge and I. Millington, Artificial Intelligence for Games, 2009.

[17] S. Rabin, “Designing a General Robust AI Engine,” *Game Programming Gems,* pp. 221-236, 2000.

[18] I. Watson and G. Robertson, “A Review of Real-Time Strategy Game AI,” *AI Magazine,* vol. 35, no. 4, pp. 75-104, 2014.

[19] Y. A. Sekhavat, “Behavior Trees for Computer Games,” *International Journal of Artificial Intelligence Tools,* vol. 26, no. 2, 2017.

[20] M. Iovino, E. Scukins, J. Styrud, P. Ögren and C. Smith, “A Survey of Behavior Trees in Robotics and AI,” *Robotics and Autonomous Systems,* vol. 154, 2022.

[21] K. Annaaisa, “Developers' Perspectives on Iteration in Game Development,” 2015.

[22] P. Abrahamsson, O. Salo, J. Ronkainen and J. Warsta, “Agile Software Development Methods: Review and Analysis,” *Proc. Espoo 2002,* pp. 3-107, 2002.

[23] “Agile Manifesto,” 2001. [Online]. Available: https://agilemanifesto.org/principles.html. [Accessed 24 March 2024].

[24] S. Saleh, S. M. Huq and M. A. Rahman, “Comparative Study within Scrum, Kanban, XP Focused on Their Practices,” in *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, 2019, pp. 1-6.

[25] B. Carvalho, C. Henrique and C. Mello, “Scrum agile product development method -literature review, analysis and classification,” *Product: Management & Development,* vol. 9, pp. 39-49, 2011.

[26] M. Alqudah and R. Razali, “A comparison of scrum and Kanban for identifying their selection factors,” 2017.

[27] Atlassian, “Trello,” [Online]. Available: http://www.trello.com. [Accessed 6 April 2024].

[28] Condor, Inc., “Diablo Game Concept,” [Online]. Available: https://www.graybeardgames.com/download/diablo_pitch.pdf. [Accessed 24 March 2024].

[29] Rockstar Games, “Race'n'Chase Game Design,” [Online]. Available: https://www.gamedevs.org/uploads/grand-theft-auto.pdf. [Accessed 24 March 2024].

[30] M. Dealessandri, “What is the best game engine: is Godot right for you?,” 15 April 2020. [Online]. Available: https://www.gamesindustry.biz/what-is-the-best-game-engine-is-godot-right-for-you. [Accessed 27 April 2024].

[31] itch.io, “Most used Engines,” [Online]. Available: https://itch.io/game-development/engines/most-projects. [Accessed 27 April 2024].

[32] R. J. Anthony, Systems Programming: Designing and Developing Distributed Applications, 2016.

[33] T. Williams, “The Singleton; a Dangerously Overused Pattern,” 6 May 2023. [Online]. Available: https://timjwilliams.medium.com/the-singleton-a-dangerously-overused-pattern-d4007758bca2. [Accessed 6 April 2024].

[34] bitbrain, “Beehave,” [Online]. Available: bitbra.in/beehave/. [Accessed 7 April 2024].

[35] Igara Studio S.A, “Aseprite,” [Online]. Available: https://www.aseprite.org. [Accessed 7 April 2024].

[36] Slant, “What are the best pixel art / sprite editors?,” [Online]. Available: https://www.slant.co/topics/1547/~best-pixel-art-sprite-editors. [Accessed 7 April 2024].

[37] bitwes, “Gut Docs,” [Online]. Available: https://gut.readthedocs.io/en/latest/. [Accessed 30 April 2024].

[38] M. Schulze, “GdUnit4,” [Online]. Available: https://mikeschulze.github.io/gdUnit4/. [Accessed 30 April 2024].

[39] I. Bogost, Persuasive Games, 2007.

[40] C. L. Bouton, “Nim, A Game with a Coplete Mathematical Theory,” *Annals of Mathematics,* vol. 3, no. 1, pp. 35-39, 1901.

[41] A. Davis, “AI Is All Around Us,” *The Institute,* vol. 40, no. 2, pp. 6-7, 2016.

[42] B. V. Bowden, Faster Than Thought: A Symposium On Digital Computing Machines, 1953.

[43] M. Rehkopf, “Scrum sprints,” [Online]. Available: https://www.atlassian.com/agile/scrum/sprints. [Accessed 6 April 2024].

[44] “IEEE Draft Guide: Adoption of the Project Management Institute (PMI) Standard: A Guide to the Project Management Body of Knowledge (PMBOK Guide)-2008 (4th edition),” *IEEE P1490/D1, May 2011,* 2011.

[45] C. Politowski, F. Petrillo and Y.-G. Guéhéneuc, “A Survey of Video Game Testing,” *2021 IEEE/ACM International Conference on Automation of Software Test (AST),* pp. 90-99, 2021.

[46] N. Pereira, P. Lima, E. Guerra and P. Meirelles, “Towards Automated Playtesting in Game Development,” 2021.