

6COSC023W – Final Project Report

Unprofitable Startup

Student: Alvaro Chavez Mixco

Supervisor: Dr Anastasia Angelopoulou

This report is submitted in partial fulfilment of the
requirements for the
BSc (Hons) Computer Games Development degree
@ the University of Westminster.

School of Computer Science & Engineering
University of Westminster

Wednesday, May 1st, 2024

Document Scope

The purpose of this document is to describe and reflect on the processes that took place for developing the Final Project, discuss any ethical issues associated with it and describe the methodology that was adopted to develop the project, its design, implementation, and testing.

All chapter word counts in this document are approximate and are not intended to be prescriptive.

Declaration

This report has been prepared based on my own work. Where other published and unpublished source materials have been used, these have been acknowledged in references.

Word Count: 19,959

Student Name: Alvaro Chavez Mixco

Date of Submission: Wednesday, May 1st, 2024

Abstract

Unprofitable Startup is a business simulation game that aims to integrate the usage of neural networks with the gameplay mechanics of managing a company.

The project was done with the purpose of simplifying the process of creating accurate and fair simulations in games by using machine learning. This is demonstrated by using a neural network capable of realistically forecasting an in-game company's profitability.

Background research showed that while machine learning in games is not a new concept, the use of machine learning, and more precisely neural networks, to affect the gameplay has been limited. This, combined with recent updates in game engines to support machine learning technologies better, presents the opportunity to develop this project.

In the game, the player manages a struggling company, an unprofitable startup, and must take it to profitability. This is achieved by allocating how the company spends on its monthly budget. Once the player has finished planning his monthly spending, the simulation of the company's performance will be executed via a neural network tasked with calculating the net income of the player's company.

The player's actions directly affect the neural network simulation by altering the neural network inputs or adjusting the net income after it has been forecasted. The project did not aim to just run the neural network on a game engine, but it also sought to integrate the neural network into the game design and game balance.

The project highlights how the neural network can go from being created and trained in a training framework, PyTorch, to being imported and used in a game engine, Unreal Engine 5, to enhance the gameplay experience. The use of the intermediate ONNX file format is vital for this process, allowing the neural network model to be represented in runtime applications like games.

The neural network was trained using the public financial statements of the last ten years of "Netflix," the world's most prominent video streaming company. From these financial statements, eight different variables were selected as inputs for the neural network, and the net income was determined as its single outcome.

These results were achieved through an agile methodology based on weekly sprints. After each weekly sprint, a shippable version of the project was produced and tested. Using this iterative process, any error found was quickly recognised and fixed; and provided an opportunity to reflect on the project's current state, reprioritising tasks or adjusting the scope as needed.

The project was tested using whitebox testing, through acceptance, regression, and unit testing matching the end of each agile sprint; and through blackbox testing, where a group of people playtested the game and provided feedback regarding its current state, user experience, and future developments.

While managing its primary aim of taking advantage of machine learning, the project lacks features in its current early state. As highlighted through user testing, the player needs more things to do in the game. Therefore, if the progress of this project were to continue, new features and game mechanics to keep the player engaged in the game would have to be developed.

Acknowledgements

I have always struggled with words, written or spoken. Still, I want to thank everyone who has helped me throughout my studies at the University of Westminster, directly or indirectly.

This thanks includes all the Computer Games Development course staff that has ever taught me and my project supervisor, Dr Anastasia Angelopoulou, for guiding me through the daunting task of developing this project.

I also want to thank my friends and family, who have supported me continuously through this process.

Table of contents

Document Scope.....	2
Declaration.....	3
Abstract.....	4
Acknowledgements.....	5
Table of contents.....	6
List of figures.....	10
List of tables.....	13
List of equations.....	14
1. Introduction.....	15
1.1. Problem statement.....	15
1.1.1. Context.....	16
1.2 Aims and objectives.....	18
2. Background.....	19
2.1 Literature survey.....	19
2.1.1. Machine learning for forecasting company profitability.....	19
2.1.2. Management simulations in games.....	20
2.1.2. Machine learning in games.....	23
2.2 Review of projects / applications.....	25
2.2.1. Machine learning algorithms.....	25
2.2.2. Management simulation game market.....	27
2.2.2. Similar games comparison.....	29
2.3. Review of tools and techniques.....	35
2.3.1. Game Engine.....	35
2.3.2. Training Framework.....	36
2.3.3. Financial Data Gathering API.....	37
3. Requirements.....	38
3.1 Stakeholders.....	38
3.2 Elicitation requirements.....	39
3.2.1. Business setting.....	39
3.2.2. Comparable games.....	39
3.2.3. Artificial intelligence.....	40
3.2.4. Simulation inputs.....	41
3.3 List of requirements.....	42
3.3.1 Functional requirements.....	42
3.3.2. Non-Functional requirements.....	44

3.3.3. Data requirements	45
3.3.4. Heuristic analysis	45
3.4. Requirement analysis and modelling.....	54
3.4.1. Context diagram.....	54
3.4.2 Use cases	55
4. Legal, social, and ethical issues.....	59
5. Methodology.....	60
5.1. Workplan	60
5.2. Prototypes	62
5.2.1. Neural network inference Unreal Engine prototype	62
5.2.2. Financial neural network prototype	64
5.2. Agile methodology.....	66
6. Design.....	68
6.1. UML	68
6.2. User interface	70
6.2.1. UI wireframe.....	71
6.3. Neural Network Design.....	75
6.4. Data flow.....	78
6.5. Neural Network – Gameplay Integration	80
6.5.1. Expenditures	80
6.5.2. Net income monthly change cap.....	82
6.6. Usability	83
6.6.1. User usability	83
6.6.2. Developer usability.....	84
7. Tools and implementation	86
7.1 Tools.....	86
7.1.1. Neural network training	86
7.1.2. Unreal Engine	89
7.2. Implementation	91
7.2.1. Creating net income neural network	91
7.2.2. Unprofitable Startup game	101
8. Testing.....	109
8.1. Test coverage	109
8.2. Test Methodology	112
8.2.1. Whitebox testing	113
8.2.1. Blackbox testing.....	121
9. Conclusions and reflections	129

9.1. Weaknesses	131
9.2. New knowledge and developed skills	131
9.2.1. Developed Skills	131
9.2.2. New Knowledge	132
9.3. Further works	133
9.3.1. Usability	133
9.3.1. Neural Network Improvement	134
9.3.2. Gameplay Features	134
9.3.3. Game Feel	135
10. References	137
Image references	142
11. Bibliography	144
Appendix 1 – Project Links	151
Appendix 2 – Elicitation requirements responses	152
Business Setting	152
Comparable Games	153
Artificial Intelligence	154
Simulation Inputs	155
Appendix 3 – Playtesting responses	157
Player Profile	157
Gameplay	158
Controls and UI	158
Company Simulation	159
Future Developments	160
Appendix 4 – Versions release notes	162
Release notes - Unprofitable Startup - V0.1	162
Release notes - Unprofitable Startup - V0.2	163
Release notes - Unprofitable Startup - V0.3	163
Release notes - Unprofitable Startup - V0.4	164
Release notes - Unprofitable Startup - V0.5	165
Release notes - Unprofitable Startup - V0.6	166
Release notes - Unprofitable Startup - V0.7	167
Release notes - Unprofitable Startup - V0.8	168
Appendix 5 – Versions test cases	169
Version 0.1	169
Version 0.2	170
Version 0.3	171

Version 0.4	172
Version 0.5	175
Version 0.6	179
Version 0.7	185
Version 0.8	192
Appendix 6 – Alpha Vantage Netflix financial statements samples	200
Cashflow.....	200
Earnings	200
Income Statement	201
Monthly Time Series.....	201
Balance Sheet	202
Appendix 7 – Completed work	203
Completed Issues.....	203
Requirements Met	203
Appendix 8 – Python required packages	204

List of figures

Figure 1: Unprofitable Startup Architecture.....	15
Figure 2: Unreal Engine Neural Network Integration [1]	17
Figure 3: Example of Variables Derived from Financial Statements [2].....	20
Figure 4: Sim City Land Value Cellular Automata [3].....	21
Figure 5: Resources-Agents [4]	22
Figure 6: OpenAI Five Dota 2 AI [5].....	23
Figure 7: Supreme Commander 2 Gameplay [6]	24
Figure 8: Machine Learning Model Comparison [2]	26
Figure 9: Steam Top Rated Management Games [7].....	27
Figure 10: Game Genre Median Earnings vs Number Release Since 2019 [8].....	28
Figure 11: Game Dev Story [9]	29
Figure 12: Game Dev Tycoon [10].....	30
Figure 13: OpenTTD [11]	31
Figure 14: Cities Skyline II [12]	32
Figure 15: Factorio [13].....	33
Figure 16: Stakeholders Onion Diagram.....	38
Figure 17: Questionnaire Business Setting.....	39
Figure 18: Questionnaire Comparable Games	40
Figure 19: Questionnaire Artificial Intelligence.....	41
Figure 20: Questionnaire Simulation Inputs	41
Figure 21: Context Diagram.....	54
Figure 22: Use Case - Start the Game.....	55
Figure 23: Use Cases - In-Game	56
Figure 24: Use Case - Resume the Game.....	58
Figure 25: Weekly Work Plan.....	60
Figure 26: Neural Network Inference Unreal Engine Prototype	62
Figure 27: Financial Neural Network Prototype Output.....	64
Figure 28: Prototype Predicted vs Actual Net Income	65
Figure 29: Issues Workflow.....	66
Figure 30: Jira Scrum Board	67
Figure 31: Jira Release Versions	67
Figure 32: UML Diagram.....	68
Figure 33: Game Dev Tycoon Gameplay Image [10].....	70
Figure 34: UI Colour Scheme.....	70
Figure 35: Netflix App UI [14].....	71

Figure 36: Wireframe Main Menu.....	72
Figure 37: Wireframe In-Game	72
Figure 38: In-Game	73
Figure 39: Wireframe In-Game (Expenditures).....	74
Figure 40: In-Game (Expenditures).....	74
Figure 41: Wireframe In-Game (Market History).....	75
Figure 42: Neural Network Inputs and Output.....	77
Figure 43: Data Flow Diagram	78
Figure 44: Marketing Investment Levels	80
Figure 45: Neural Network Gameplay Integration.....	82
Figure 46: User Usability Features.....	84
Figure 47: Developer Usability.....	85
Figure 48: Training Framework Usage.....	87
Figure 49: Alpha Vantage Endpoints Documentation	88
Figure 50: ONNX Runtime Integrations	89
Figure 51: Game Engine usage.	89
Figure 52: Interpolate Financial Data Function	92
Figure 53: Interpolated Data Stock Value	93
Figure 54: Net Income Neural Network.....	94
Figure 55: Code to create a PyTorch Neural Network	95
Figure 56: Code to train the Neural Network.....	96
Figure 57: Neural Network Predicted vs Actual Net Income	99
Figure 58: scaling_data_information.json	100
Figure 59: Code FSFinancialData.....	101
Figure 60: Code for header Neural Network Wrapper Class	102
Figure 61: Code to parse minimum and maximum Values	103
Figure 62: Code to Scale Financial Data	104
Figure 63: Code to run the neural network in Unreal Engine	105
Figure 64: Code Simulate Month	106
Figure 65: Game Manager Public Members	107
Figure 66: Base Submenu Widget Blueprint.....	108
Figure 67: Unreal Engine Unit Testing	109
Figure 68: Neural Network Test Level.....	110
Figure 69: Bug Report.....	112
Figure 70: Acceptance Testing	113
Figure 71: Playtesting Questionnaire Player Profile.....	122
Figure 72: Playtesting Questionnaire Gameplay.....	123

Figure 73: Playtesting Questionnaire Controls and UI	124
Figure 74: Playtesting Questionnaire Company Simulation.....	125
Figure 75: Playtesting Questionnaire Future Developments.....	126
Figure 76: Blackbox Testing Requested Features	127
Figure 77: Game HUD Visual Instructions	128
Figure 78: Example of Overlay Explaining UI [15]	133
Figure 79: Game Dev Story Project Feature [9].....	135
Figure 80: Backlog	136

List of tables

Table 1: Game Features Comparison.....	34
Table 2: Game Engine Comparison.....	35
Table 3: Deep Learning Training Framework Comparison	36
Table 4: Financial Data API Comparison	37
Table 5: Functional Requirements	43
Table 6: Non-Functional Requirements	44
Table 7: Data Requirements	45
Table 8: Heuristic Analysis.....	53
Table 9: Use case - Start the Game.....	55
Table 10: Use Case - End the Current Turn	56
Table 11: Use Case - Get a Loan	57
Table 12: Use Case - Start a New Project	57
Table 13: Use Case - Resume the Game	58
Table 14: Third Party Assets List	59
Table 15: Work Plan Deliverables.....	60
Table 16: Work Plan Milestones	61
Table 17: Neural Network Evaluation.....	98
Table 18: Best Neural Networks	99
Table 19: Version 0.8 Test Cases (Condensed).....	121
Table 20: Functional Requirements Fulfilled.....	130

List of equations

Equation 1: Mean Forecast Error (MFE)	96
Equation 2: Mean Absolute Error (MAE)	97
Equation 3: Mean Absolute Percentage Error (MAPE)	97
Equation 4: Mean Squared Error (MSE)	97
Equation 5: Root Mean Squared Error (RMSE)	97
Equation 6: Min-Max Normalization 0 to 1	102

1. Introduction

Unprofitable Startup is a business simulation game that uses a neural network to simulate a company's financial performance. In it, the player's goal is to manage the company budget to turn the unprofitable startup into a profitable, successful company.

1.1. Problem statement

Games occur in virtual worlds, where multiple systems and elements of the game world must be simulated. However, the process of creating various mathematical models to simulate these can be a daunting and time-consuming task that, if done incorrectly, can break the immersion in the game (Ashton, 2020).

This project aims to simplify the process of creating accurate and fair simulations in games by using machine learning.

This concept is showcased in this project by creating a neural network that can simulate a company in a virtual world and accurately and fairly (from the player's perspective) simulate the profitability of that company. The neural network will be fully integrated into a game and run at runtime to update the company's financial status.

The company simulation component will serve as the basis of "Unprofitable Startup", a business simulation game where the player will manage a well-established company that is continuously losing money despite its size and market share. The player's goal of making the company profitable will be achieved by making different budgetary decisions affecting the result of the machine learning simulation algorithm.

This will demonstrate the process of integrating a neural network created in a machine learning framework with the game mechanics created in a game engine.

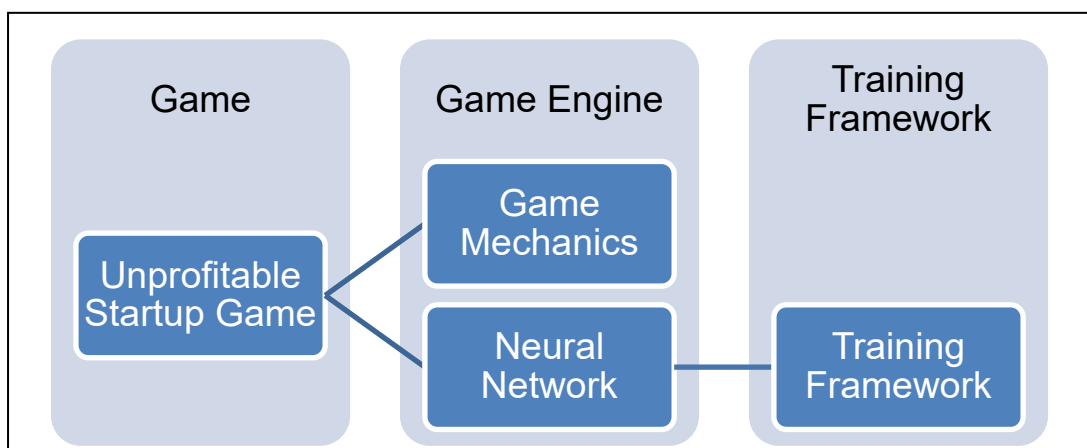


Figure 1: Unprofitable Startup Architecture

The Unprofitable Startup game will integrate a neural network and gameplay mechanics to simulate a company's performance. This means connecting the neural network produced by the training framework, with the gameplay code in the game engine.

1.1.1. Context

The use of machine learning in video games is a concept that has been introduced previously; however, this has rarely gone beyond an experimental or novelty state. In games, this technology has generally focused on two key areas (Lanham, 2018) (Engelbrecht, 2023):

- **Artificial Players:** An Artificial intelligence capable of playing a game by either simulating a player's behaviour (simulation learning); or perfectly optimising its gameplay (reinforcement learning).
- **Productivity Tools:** Machine Learning tools that can help with the process of making the game itself (assets generation) or testing already existing functionality.

Though often considered somewhat of a niche market (Wijman, 2024), there are multiple simulation games that focus on simulating different systems, and while the method for the simulation can differ from game to game, rarely has a neural network been used for this. Many games instead rely on a grid-based simulation (Millington & Funge, 2009), which works by representing graphically the states of a spreadsheet-like structure; or by an agent's system, where agents carry resources through different parts of the system (Librande, 2013).

Creating, maintaining, and tweaking these systems is a complex task prime for automation. For example, game economy simulations usually require specialised economy designers to develop and refine mathematical models to balance the game economy. These designers often can see the metrics of what is happening but are not able to establish correlations between them (Ashton, 2020), something that machine learning algorithms excel at.

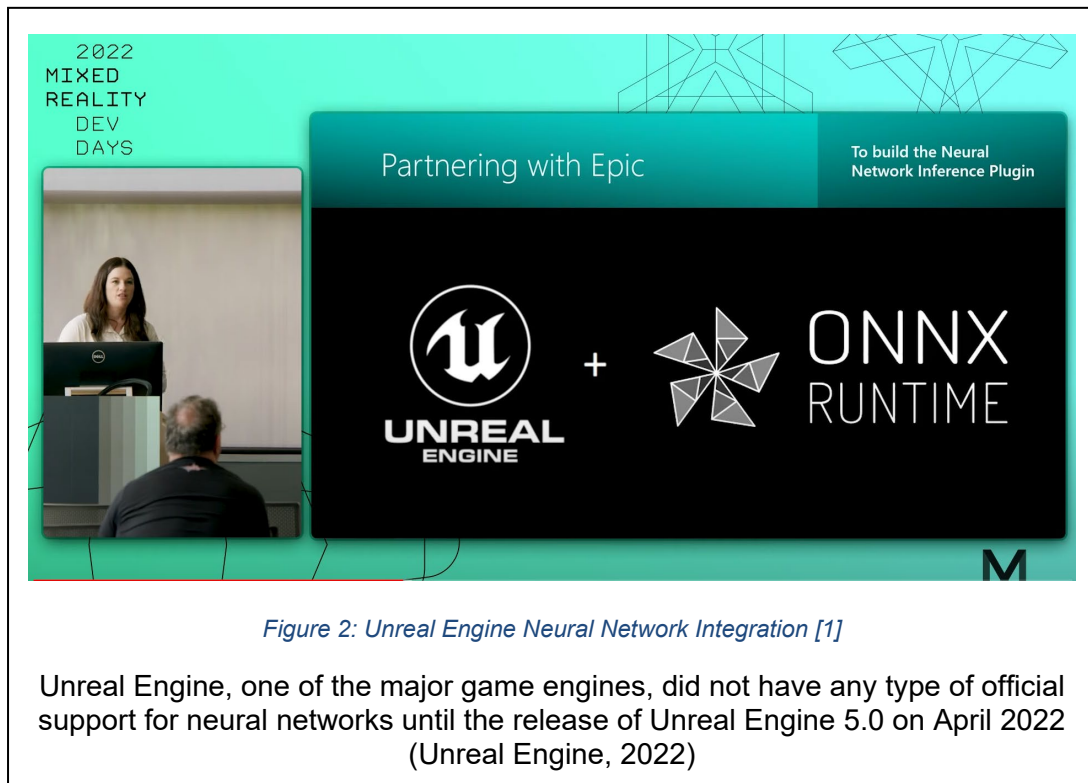
This shows a knowledge gap where the use of machine learning could simplify the process of simulating game systems, such as a game's virtual economy.

This gap was not directly created by a developer's/designer's choice of simply not using machine learning for the simulation in their game. However, a big reason for this gap is technology availability.

Until relatively recently, game engines did not offer any official support for using neural networks or another form of machine learning in their software. This meant that any developer who wanted to use a neural network in their software would be required to develop the code to create and train the neural network, as well as the inference engine needed to run the neural network at runtime (Robbins, 2013). This meant that adoption of these technologies was often not a viable option.

However, recent developments, with many of them still being in experimental states, mean that the major game engines (Unreal Engine, 2023) (Mattar, et al., 2020). Now, there is official support for neural network inference inside the game engine. While intermediary file formats now allow neural network models to be exported from a training framework into different runtime components (Breviu, 2022).

However, because of the novelty of many of these technologies, the knowledge base on how to use them and their possible potential is limited at this point. The project could potentially contribute to the process of filling this gap by taking the chance to work with these new technologies and documenting the results.



1.2 Aims and objectives

The aims describe the overall goals that wish to be achieved by implementing the Unprofitable Startup project. At the same time, the objectives provide measurable outcomes to evaluate if the proposed aims were met.

The project scope is to develop two main components: a neural network used to predict a company's net income and a simulation game to serve as the "front-end" element of the neural network. Integrating these two components will allow the player to see the effects of a machine learning simulation and affect it through its actions.

Aims:

1. To simulate in a plausible manner how different user choices can affect a company's economic prospects.

Objectives:

- a) Determine what data factors can affect a company's economic performance.
- b) Create or modify a Neural Network to forecast a company's financial state.
- c) Gather data on what people believe affects a company's profitability.

2. Allow user actions to affect the forecasting of the company profits in a simulation.

Objectives:

- a) Create tweakable variables to adjust a company's forecasted result.
- b) Create a UI for the user to interact with the simulation.
- c) Present simulated economic data to the user in an easy-to-understand manner.

3. To develop a project that takes advantage of Machine Learning technologies to provide realistic simulations.

Objectives:

- a) Export Neural Network models in ONNX format from training frameworks like PyTorch.
- b) Import Neural Network ONNX models into Unreal Engine,
- c) Run Neural Networks models from within the Unreal game engine at an acceptable performance.

4. To create a company simulation game (beta version) using a game engine

Objectives:

- a) Integrate the company's profit simulation in the game mechanics.
- b) Create a code data structure to represent the current state of a company's financial data in the game.
- c) Track and showcase the player's progress toward the overall goal of making the company profitable.

2. Background

To plan this project's development and fully gauge its viability and the outcome that could be produced, it was necessary to do background research on both the company simulation through machine learning “back-end” and the simulation game “front-end”.

2.1 Literature survey

Creating a simulation game using machine learning is a concept that has been introduced previously. However, limiting the literature survey to only this particular combination of game genre and technology would severely limit the number of resources available for the literature survey.

Instead, to fully understand how this project would fit with the existing content knowledge base, it was necessary to break down the project into individual components, like how neural networks are used in games and how machine learning can forecast a company's profitability.

2.1.1. Machine learning for forecasting company profitability

Forecasting a company's profitability has always been a priority of many stakeholders, such as investors and credit reporting agencies. Before the recent revolution of machine learning, stakeholders developed their own algorithms. These algorithms often relied on arbitrary parameters, such as the ability of the owner to manage the business, stability, the conditions of the company assets, reputation, etc. The use of machine learning instead of these proprietary algorithms not only allows a systematic and objective approach to these analyses but has also proven to improve the accuracy of the prediction by handling hidden relationships between the financial variables (Daisuke, et al., 2017).

The financial statements (Income Statement, Balance Sheet, Cash Flow Statement), presented quarterly by publicly traded companies, can be used to know the company's financial status and analyse their competitive advantage based on factors such as long-term debt, profit margins, and capital expenditure (Buffett & Clark, 2008).

The financial statements contain hundreds of parameters, requiring some form of feature reduction to reduce the complexity of the neural network. Some variables used are not even directly obtained from the financial statements; instead, they are derived through calculations of the variables that are available (Montesinos, 2022).

Variables	Calculation
Gross Margin Ratio	$\text{Revenue} - \text{Cost of Goods Sold (COGS)} / \text{Revenue}$
Profit Margin Ratio	$\text{Net Income} / \text{Revenue}$
Free Cash Flow Margin	$\text{Net Cash Flow} / \text{Revenue}$
Return on Assets	$\text{Net Income} / \text{Total Assets}$
Return on Equity	$\text{Net Income} / \text{Average Shareholders' Equity}$
Current Ratio	$\text{Current Assets} / \text{Current Liabilities}$
Quick Ratio	$\text{Current Assets} - \text{Inventory} - \text{Prepaid Expenses} / \text{Current Liabilities}$
Cash Ratio	$\text{Cash \& Cash equivalents} / \text{Current Liabilities}$
Debt Ratio	$\text{Total Debt} / \text{Total Assets}$
Debt to Equity Ratio	$\text{Total Liabilities} / \text{Total Shareholders' Equity}$
Cash Flow to Debt Ratio	$\text{Cash Flow from Operations} / \text{Total Debt}$
Operating Cash Flow Sales Ratio	$\text{Operating Cash Flow} / \text{Revenue}$
SGA to Revenue	$\text{Selling, General and Administration expenses} / \text{Revenue}$
R&D to Revenue	$\text{Research and Development} / \text{Revenue}$
Capex to Revenue	$\text{Capital Expenditures} / \text{Revenue}$

Figure 3: Example of Variables Derived from Financial Statements [2]

Previous projects have not only relied directly on the variables obtained from the financial statements, but also derive certain parameters based on the data available.

Supervised learning models are often used in forecasting via machine learning. These generally rely on previous years' financial statements to forecast whether a company will be profitable the following year or not. The output of this prediction can be a simple classification of whether the company will be profitable or not or a regression that predicts the profits. The accuracy of these models is evaluated by comparing the predicted values against the true values of the actual data when the neural network receives the same inputs (Gaikwad, et al., 2023).

2.1.2. Management simulations in games

Construction and simulation Games (CSGs) involve the player building and/or managing a system, like a city or company, within the game's constraints. A subset of this genre is pure business simulation games, where the player does not have to build anything in the virtual world but is only focused on managing the financial aspect. In essence, these games consist of a series of screens where different choices can be made to manage the finances (Adams, 2013).

The core mechanics of simulation games often rely on players managing resources, from their source, where they are produced, to their drain, where they are consumed. In many games, the primary resource being managed is money. This management of resources is usually indirect, with the player only being able to alter these processes in a limited way and generally having to rely on trial and error to see the effect of their decisions (Adams, 2013).

From a technical standpoint, how the simulation is done can vary widely from game to game, but the main approaches to simulation are:

- Cellular Automata: Mostly used for city simulation games, this breaks down the map into a grid. The performance of a structure or object in a tile is calculated relative to the values of the surrounding locations. This is then as an iterative process, where the performance is recalculated, each iteration based on the potentially new values of the environment (Millington & Funge, 2009).



Figure 4: Sim City Land Value Cellular Automata [3]

In Sim City, the cellular automata method is used to simulate the state of a building, based on the land value of the surrounding neighbourhood. If the area is run-down, the building will not prosper, decay, and eventually become abandoned.

- AI Agents: This approach relies on creating an AI that can make financial decisions by adjusting economic policies. This is generally done through rule-based reasoning, similar to a state machine, where the action to take is determined by the rules defined; or through a goal-oriented behaviour, where the AI uses a Markovian approach to evaluate the effects of each immediate choice (McCarlie & Hunter, 2021) (Zubek, 2015).
- Resources-Agents: The simulation relies on managing individual agents in the virtual world that transport the game resources to the different “units” or places where the resources can be converted or consumed. This can be quite a performance-taxing method due to requiring the simulation of multiple individual agents (Librande, 2013).

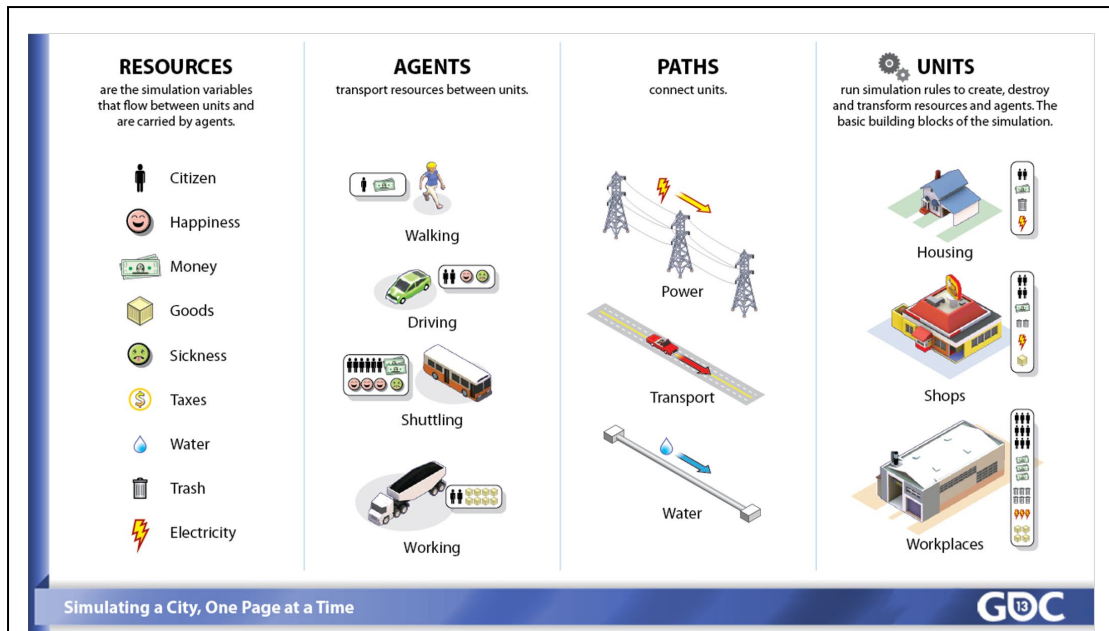


Figure 5: Resources-Agents [4]

The resources agents work by simulating the flow of resources from where they are produced to where they are consumed, including how they are transported via agents.

In terms of simulation, it is also worth noting that many simulation “games” are not done for recreational purposes but instead purely for a training simulation objective. These are focused on teaching certain principles in an engaging and effective way (Sitzmann, 2011).

2.1.2. Machine learning in games

The use of machine learning in games has, for the most part, been limited to two areas:

- **Artificial Players:** It involves creating AI capable of playing the game by either simulating the player's behaviour (imitation learning) or by optimising the gameplay performance through a punishment-reward system (reinforcement learning) (Engelbrecht, 2023).
- **Tools and Technologies:** Besides the rise of generative AI that can be involved with asset or code generation, machine learning has also been used for automating testing (Lanham, 2018), or even upgrading existing assets, like upscaling the image quality of older games for remasters/remakes (Thompson, 2021).

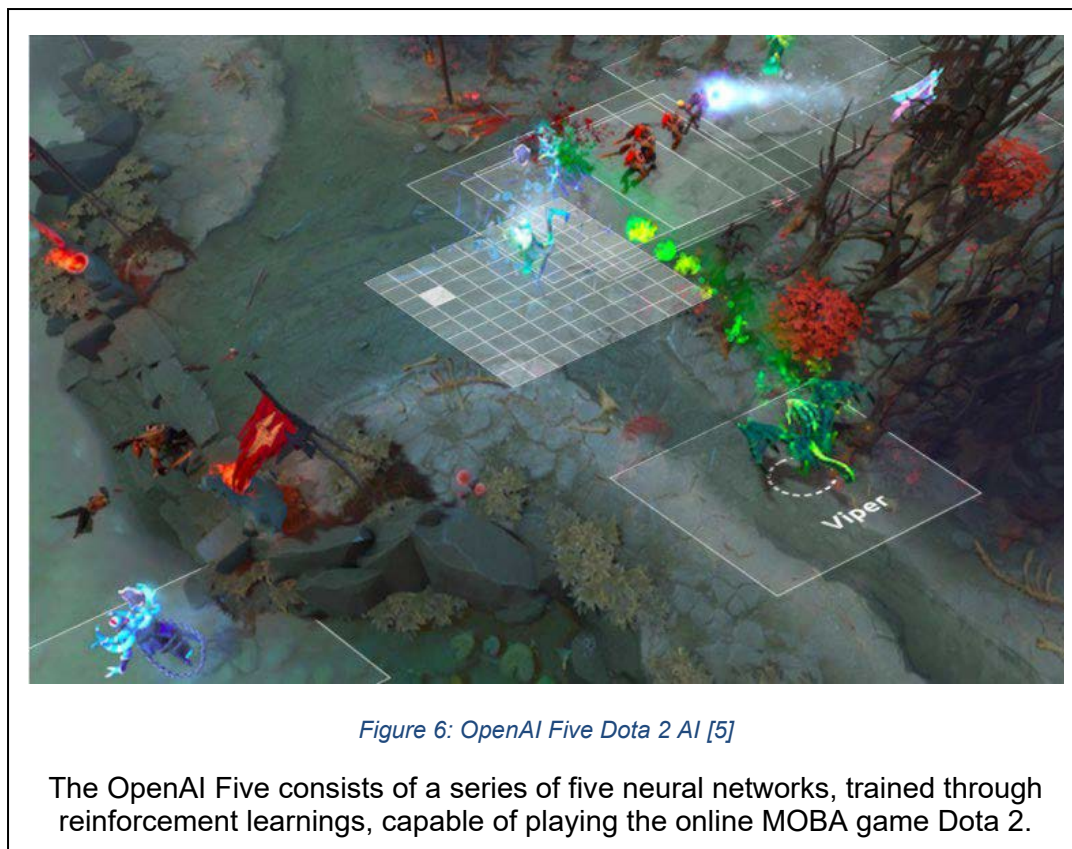


Figure 6: OpenAI Five Dota 2 AI [5]

The OpenAI Five consists of a series of five neural networks, trained through reinforcement learnings, capable of playing the online MOBA game Dota 2.

Though these are the current main areas of usage of this technology, reinforcement learning has already been used for other gameplay purposes, though in a more limited manner.

For example, Supreme Commander 2, a real-time strategy game released in 2010, used neural networks to control the threat response of the AI units in the game. The neural network had as inputs both the unit's and its opposing enemy status, and produced the utility values of the actions the AI unit could take as an output. This included which unit should be prioritised when attacking or even running away if the threat was too significant. This method proved effective in saving time since the relationships of each individual status variable did not have to be manually established. However, it did remove the ability of designers to finetune the project game balance (Robbins, 2013).

The limited usage of neural networks in games is also worsened by the lack of transparency since they are generally considered a blackbox algorithm. This means that it may be hard for testers and players to understand how or why the neural network made a decision and, makes it difficult to predict the neural network's output. This lack of transparency in the decision-making process complicates the testing and debugging of neural networks in games (Bourg & Seemann, 2004).

These limitations of neural networks can be somewhat mitigated by compartmentalising their functionality. Rather than having a single huge neural network decide everything as the entire AI system, it is better to have the neural network/s make decisions on a single issue. This means that the neural network is specialised in making a single type of decision for the overall AI system. This can not only improve the performance of the neural network but also make it easier to debug and more predictable (Bourg & Seemann, 2004).



Figure 7: Supreme Commander 2 Gameplay [6]

The Supreme Commander 2 game uses a neural network to calculate the best action a unit can take when facing a threat.

2.2 Review of projects / applications

2.2.1. Machine learning algorithms

As the project requires forecasting a company's net income, this requires a regression algorithm, which usually implies the use of supervised learning. Some of the algorithms that could be used for this are:

- Neural Networks (NN) are a deep learning algorithm that maps input and output through a series of hidden layers made up of neurons, interconnected nodes. The relationship between the input and output is established as a series of weights and biases through summation and other forms of non-linear transformation. These are generally a supervised learning algorithm, where the weights of the neural network are adjusted through an iterative process by using a dataset that already has the target output mapped with its inputs (Lado-Sestayo & Vivel-Búa, 2020).
- Random Forests work via a decision tree algorithm. This algorithm runs multiple decision trees and outputs the average value produced from all the decision trees. This tree partition often helps minimise the loss function when training the neural network (Gaikwad, et al., 2023).
- Support Vectors can be used for both regression (SVR) and classification (SVM) tasks. These operate by finding the multidimensional hyperplane that best fits the data. This process is usually done after kernelization, where the data has been broken down into easier-to-process dimensions (Gaikwad, et al., 2023).
- Adaptive Boosting (AdaBoost) is a non-linear algorithm that works by first training a weak learner model, and then having subsequent models that will adjust the errors of the previous model until they converge into a strong learner model. This boosting-based approach reduces the likelihood of overfitting (Shuying Li, 2020).

Though all these algorithms could be used for the regression, sharing similar inputs and evaluation metrics, the Neural Networks tend to provide the best performance. Nevertheless, this often requires a trade-off in increased training time for neural networks and being more at risk of overfitting (which would cause the model to have significantly worse performance with data that was not seen during its training) (Montesinos, 2022). Despite this, it could be argued that each algorithm's performance would depend on the inputs and architecture applied to each project's model.

	DNN	DNN wSD	Random Forest	AdaBoost	SVM
Recall True Positive	91	90	90	90	93
Precision True Positive	85	84	85	85	79
Recall True Negative	65	61	66	66	45
Precision True Negative	78	73	75	74	74
F1 True Positive	88	86	88	87	85
F1 True Negative	71	66	70	70	56
Accuracy	83	81	83	82	78
10 Fold Cross Validation	81.9	78.8	82.8	82.9	82.4

Figure 8: Machine Learning Model Comparison [2]

Research comparing different machine learning models have shown Deep Neural Networks (DNN) to have the best performance when classifying if a company is profitable or not.

However, the main advantage neural networks have is their portability. They are the only machine learning regression model that has an intermediate format, ONNX, that can be used to integrate it with a game engine (ONNX Runtime, n.d.).

Though I am sure some of these algorithms could be implemented on the game engine via code, the engines are not designed for data sciences, making this a time-consuming and less-than-ideal prospect.

2.2.2. Management simulation game market

Management games are often hard to classify into a specific game genre or even as a sub-genre. This is not because there are no games about management or simulation, but rather because a lot of games use these mechanics as a subset for another game (McAulay, 2019).

The clearest example of this can be seen when filtering games in [Steam](#), the leading storefront for online PC games, with the “Management” tag. Applying this filter can lead to a wide range of games where management may be an important component but not the game's focus. These can include city builders, 4X strategy games, among others.

Because of this, pure business simulation games (where managing is the core mechanic of the game) are often classified with both “Management” and “Simulation” tags, and even then, they can also include other types of hybrid games. Despite not being an official genre classification for this type of game, they are usually grouped as either strategy or simulation games (Kelly, 2021).

These issues classifying management simulation games mean that many analyses do not cover this genre directly, including it as a subset of another genre, or can potentially misclassify it. Because of this, the superset of strategy or simulation genre is considered when analysing certain characteristics of the simulation management genre.

Based on the strategy and roleplaying games, the target audience for these types of games would be males between 25-34 years old. Over half of these players are employed in sales, finance, or marketing (AARKI, 2020).

Meanwhile, in the growing mobile market, the simulation games genre is the biggest growing genre in terms of in-app advertising revenue. Though this increase has largely replaced the revenue generated by in-app-purchases (Unity, 2024).

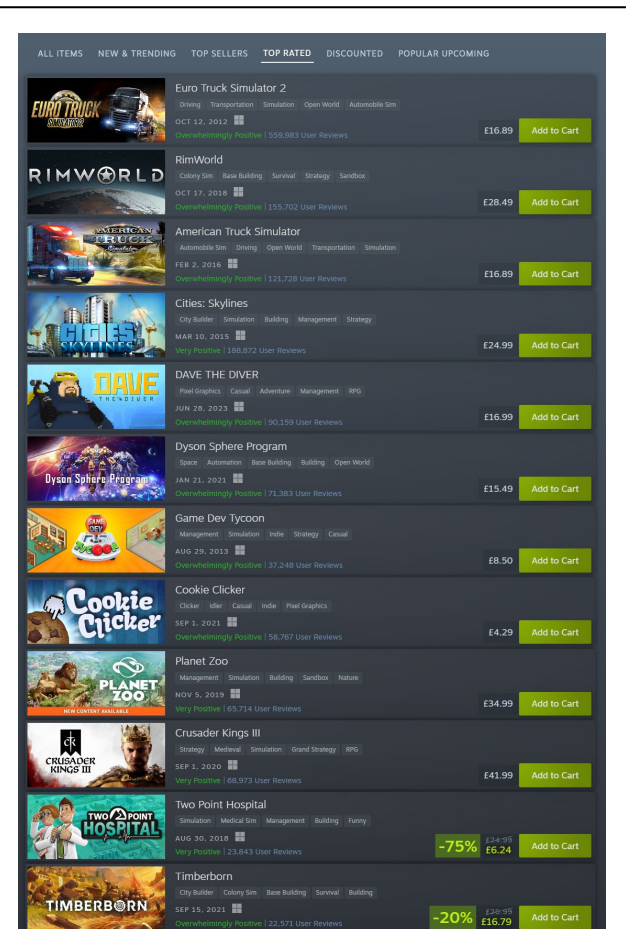
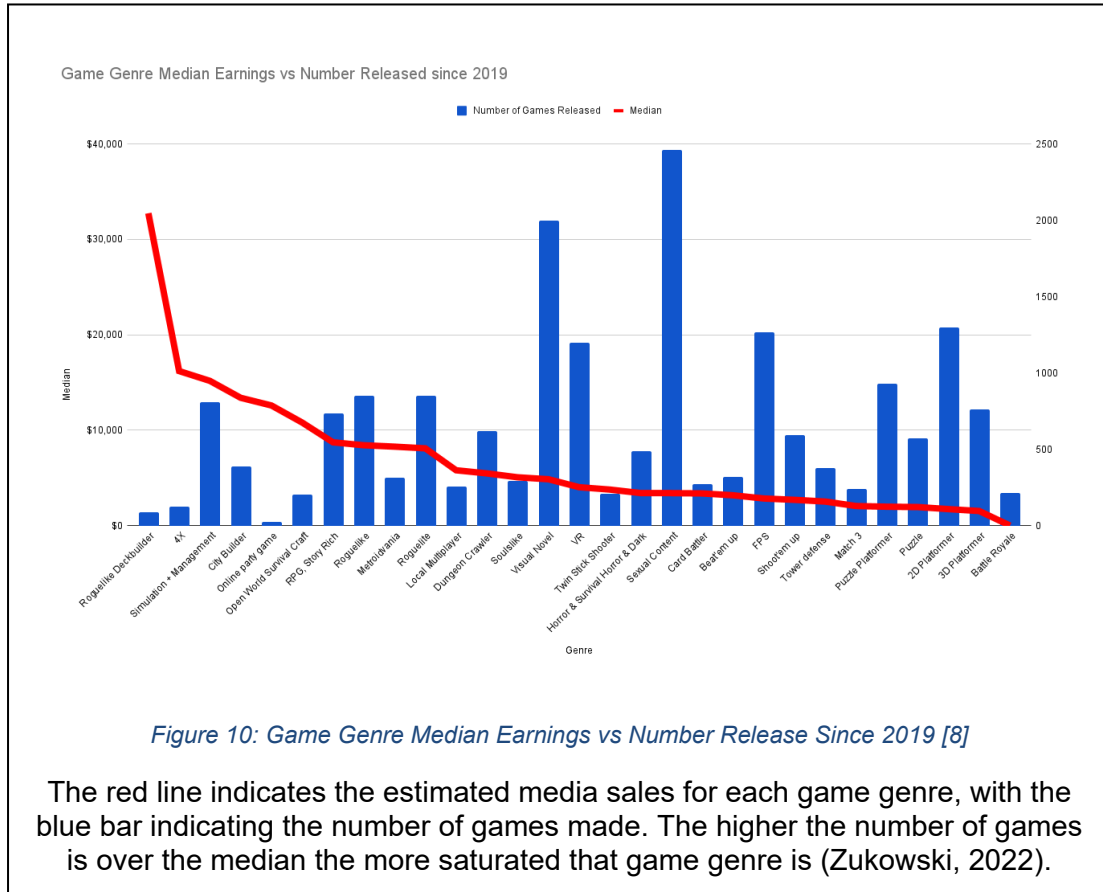


Figure 9: Steam Top Rated Management Games [7]

Comparing blindly all games labelled as “Management” would not be accurate, due to the wide range of actual game genre that involve management.

When analysing the number of games tagged as simulation + management (which, again, can include other types of hybrid games), with the median number of sales, it can be seen that the sales for this type of game rank among the highest per game made. Also, in comparison to other genres, the game market is not oversaturated, nor are players focused on only a couple of major games, with them being willing to try similar games in the genre (Zukowski, 2022).



2.2.2. Similar games comparison

To better understand the current market of management simulation games, one must see the features they offer compared to those planned for the Unprofitable Startup project.

2.2.2.2. Game Dev Story



Figure 11: Game Dev Story [9]

A management game made with a 2D-pixel art style. The player aims to make the best-selling game to earn profits. The game relies on choosing the right staff for each stage of the game development project and even allows the player to create their own video game console (Kairosoft Co., Ltd, n.d.).

When developing a game, the player can choose the genre of the game to make and the areas to focus on (graphics, sound, etc.). Once the game is released, it gets a review score from a panel of critics before being released for sale. The player must also manage his fanbase by advertising their games or attending conventions.

Since this started as a mobile game, the UI layout and art style were emphasised for that platform and touch controls.

2.2.2.2. Game Dev Tycoon



Figure 12: Game Dev Tycoon [10]

Classified as a business simulation game, the player is tasked with replaying the history of the gaming industry. The player manages a company starting from the 80s all the way up to the present time, experiencing console launches, conventions, and other events (Greenheart Games, 2012).

The game is focused on developing both game projects and game engine projects, where the player can decide which area to focus the workforce on.

This has often been compared to Game Dev Story, sharing most of the same mechanics. However, this one has a stronger emphasis on showcasing the history of the game industry.

2.2.2.3. OpenTTD



Figure 13: OpenTTD [11]

OpenTTD is an open-source business simulation game where the player is tasked with earning money by transporting passengers or resources. This game started as a remake of Transport Tycoon Deluxe but has expanded beyond it (OpenTTD, n.d.).

The game has a randomly generated map, where the player needs to construct and layout the different infrastructure for the transport (train tracks, bus stations, etc.), purchase the vehicles, and schedule the transportation routes. Some of the forms of transportation can include buses, trains, aeroplanes, lorries, and boats. In this context, the player is not solely managing the company itself but also must manage the whole transportation network.

Because the game is a remake of a 1994 MS-DOS game, the art and user interface are reminiscent of the one of that time, which may not be the most appealing or easy to use for new players. However, being open-source has also led to a great degree of customisation, with players being able to create mods and their own AI.

2.2.2.4. *Cities: Skylines II*

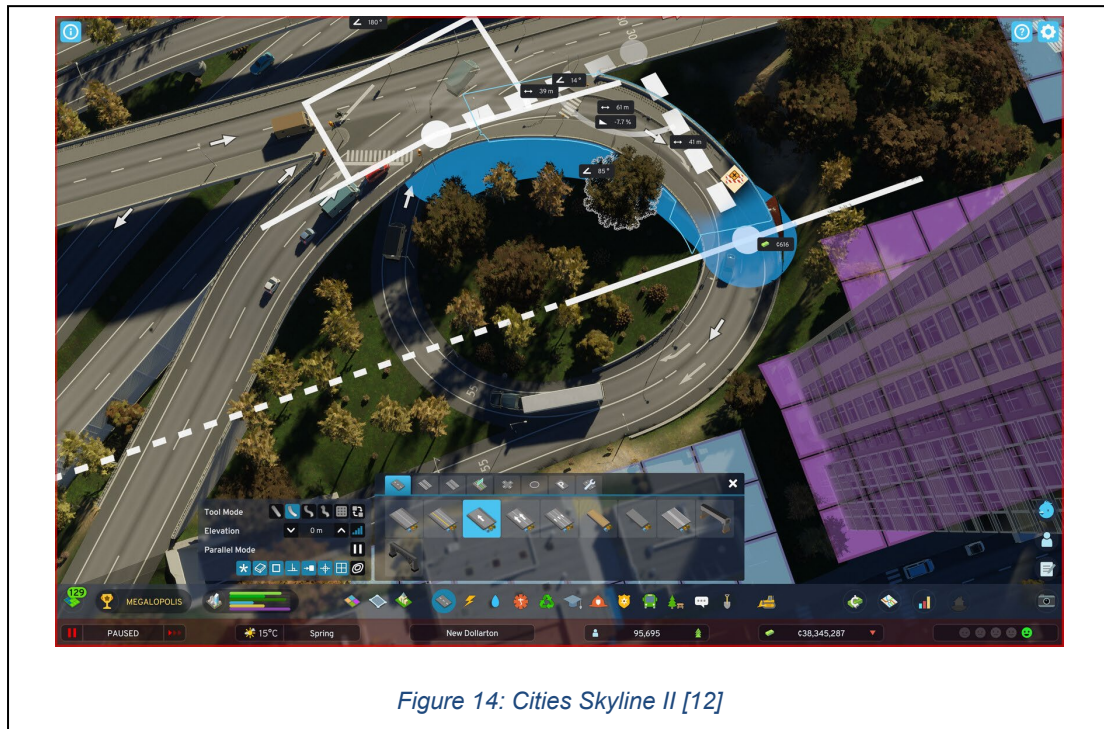


Figure 14: *Cities Skyline II* [12]

It is a city-building game where the player must build a city, managing systems like traffic, taxes, pollution, crime, etc. These rely on AI and intricate economics for a deep simulation of the game systems (Paradox Interactive, n.d.).

As a city builder, it has a component related to the placement of the different city improvements and managing the plots of land. This often requires balancing the various needs of the city and reacting to different challenges and opportunities.

The player can also manage the city districts' different policies, such as tax rates and ordinances, to further control their development. This is while being mindful about collecting funding to maintain and expand the city.

2.2.2.5. Factorio



Figure 15: Factorio [13]

A construction management game, the player must create a rocket to escape the alien planet where he has crash-landed by creating an automated factory to produce the required resources to survive and escape (Wube Software, n.d.).

The gameplay relies on managing the factory's production and optimising its production line. These optimisations often involve the further automation of the different processes and the research of new technologies.

Factorio also has a combat mechanic, where the player is tasked with defending his factory from the attacking aliens.

The game has a multiplayer component where the players can play cooperatively or against each other. Construction blueprints (layout of factory system) can also be shared between users.

2.2.2.6. Comparison

Feature	Unprofitable Startup	Game Dev Story	Open TTD	Cities: Skylines II
Platform	PC	PC/iOS/Android/Nintendo Switch/PlayStation 4/Xbox One/Xbox Series X	PC/Mac/Linux/Android	PC
Type of simulation game	Business Simulation	Business Simulation	Construction Simulation	Construction Simulation
Realtime/Turn-based	Turn-based	Realtime	Realtime (can be paused)	Realtime (can be paused)
Graphics	3D	2D pixel	2D retro	3D
Manage the budget	Yes	Yes (Indirectly through projects)	No	Yes
See market history over time.	Planned	No (Can only see past products)	Yes	Yes
Manage staff	Planned	Yes	Yes (vehicles)	N/A (workers demand/supply is simulated)
Research bonuses	Yes	Yes	No	Yes
Simulation algorithm	Neural Networks	AI Agents	Resources-Agents	Resources-Agents

Table 1: Game Features Comparison

As mentioned in 2.2.2. Management simulation game market, a lot of the management simulation games are hybrid, having some form of construction mechanic, and are not purely business simulation games, like Unprofitable Startup. Perhaps the closest games, in terms of gameplay, would be Game Dev Story and Game Dev Tycoon.

In this comparison, while Unprofitable Startup may be lacking in terms of features, for this early version at least (with a lot of the comparable features being planned for 9.3. Further works) the use of a neural network and the high-quality 3D graphics could prove to be a distinguishing factor.

The distinguishing factor, combined with the fact that Unprofitable Startup already has, or plans to have, many of the features present in some of the most successful simulation management games, indicates the project's potential success.

2.3. Review of tools and techniques

This project will involve a wide range of different tools, from project management like Jira to development IDEs like Visual Studio and version control software such as Git. However, the main tools to be evaluated for this project were:

- Game Engine: The software used to develop the gameplay mechanics.
- Training Framework: The tool used to create, train, and export a neural network capable of predicting a company's net income.
- Financial Data Gathering API: The API used to gather the historical financial statements of a company, which is the data to be used to train the neural network.

These three different tools are the basis for developing the essential requirements of the project and fulfilling its aims.

2.3.1. Game Engine

To simplify the game development process, a game engine that provides core features like rendering, scripting, audio playback, and asset management was required.

As integrating a Neural Network with run-time gameplay elements is one of the project's main aims, which is not a standard game engine component, this limited the choice of which game engine to use to those with official Neural Network/Machine Learning support. Of the major game engines, only Unity, through its Machine Learning Agent package, and Unreal Engine, through its Neural Network Engine (NNE) plugin, offer this feature.

Because of this, a comparison between the only two viable choices of Unreal Engine and Unity was needed:

Feature	Unreal Engine	Unity
Programming Language	C++	C#
Visual Scripting	Yes	Yes
3D/2D Games Support	Yes	Yes
Source Code Available	Yes (Free)	Yes (Paid Enterprise plan)
Neural Network Integration	Yes (Neural Network Engine plugin)	Yes (Machine Learning Agent Package*)

Table 2: Game Engine Comparison

**Note: The Unity Machine Learning Package is a superset of the whole machine learning system. The neural network inference library to run the neural networks is based on the Barracuda (to be superseded by Sentis) packages, installed as dependencies of the Machine Learning Package (Unity, 2023)*

Both game engines can import a trained neural network model that has been exported from a training framework as an ONNX file (ONNX Runtime, n.d.).

2.3.2. Training Framework

While the game engines support running an inference on a neural network at run-time, and, in varying degrees, the training of the model inside the engine; due to the considerable amount of data being handled, which is not related directly to gameplay, doing the training from the game engine would be impractical.

Because of this, a dedicated deep learning framework, where the data could be imported and filtered and the neural network model trained and evaluated, was required. The only requirement of this framework is for it to be able to export the trained neural network model as an ONNX file so that it can be brought into the game engine.

The most popular deep learning training frameworks have the following features (Sayantini, 2023) (Mariana Berga, 2024):

Feature	PyTorch	Tensorflow	Keras
Programming Interface	Python / C++	Python / C++ / Java	Python
API Level	Low	High and Low	High (Run on top of TensorFlow, Theano or CNTK)
Build-in Data Visualisation	Limited (Visdom)	Yes (TensorBoard)	Yes (TensorBoard, if used on top of TensorFlow)
Debugging	Python Standard Debugger	TensorFlow Debugger	None (Low-level API debugger can be used)
Models Export to ONNX	Yes	Yes	Yes (via Tensorflow library)

Table 3: Deep Learning Training Framework Comparison

2.3.3. Financial Data Gathering API

As is the case for machine learning, a substantial amount of data is needed to train the Neural Network. Since the neural network for the game is intended to infer a company's net income, financial data such as cash flows and income statements would be needed.

This financial data is available to the public for all publicly traded companies. An API to gather the financial data will be required to easily get enough data entries from across multiple years of a company's performance to be able to train the neural network.

From the prototyping stage of this project, two main viable APIs were found based on the financial data required to train the Neural Network. These were (Aroussi, 2024) (Alpha Vantage, n.d.):

Feature	Alpha Vantage	yFinance
Endpoints via	HTTP Requests	Python
Data Obtention Method	API Endpoint	Web scraping Yahoo Finance Website
Financial Data Records	~15 years	~1.5 years
Open Source	No	Yes
Endpoints	Stock Price, Income Statements, Cash Flow, Earnings	Stock Price, Income Statement

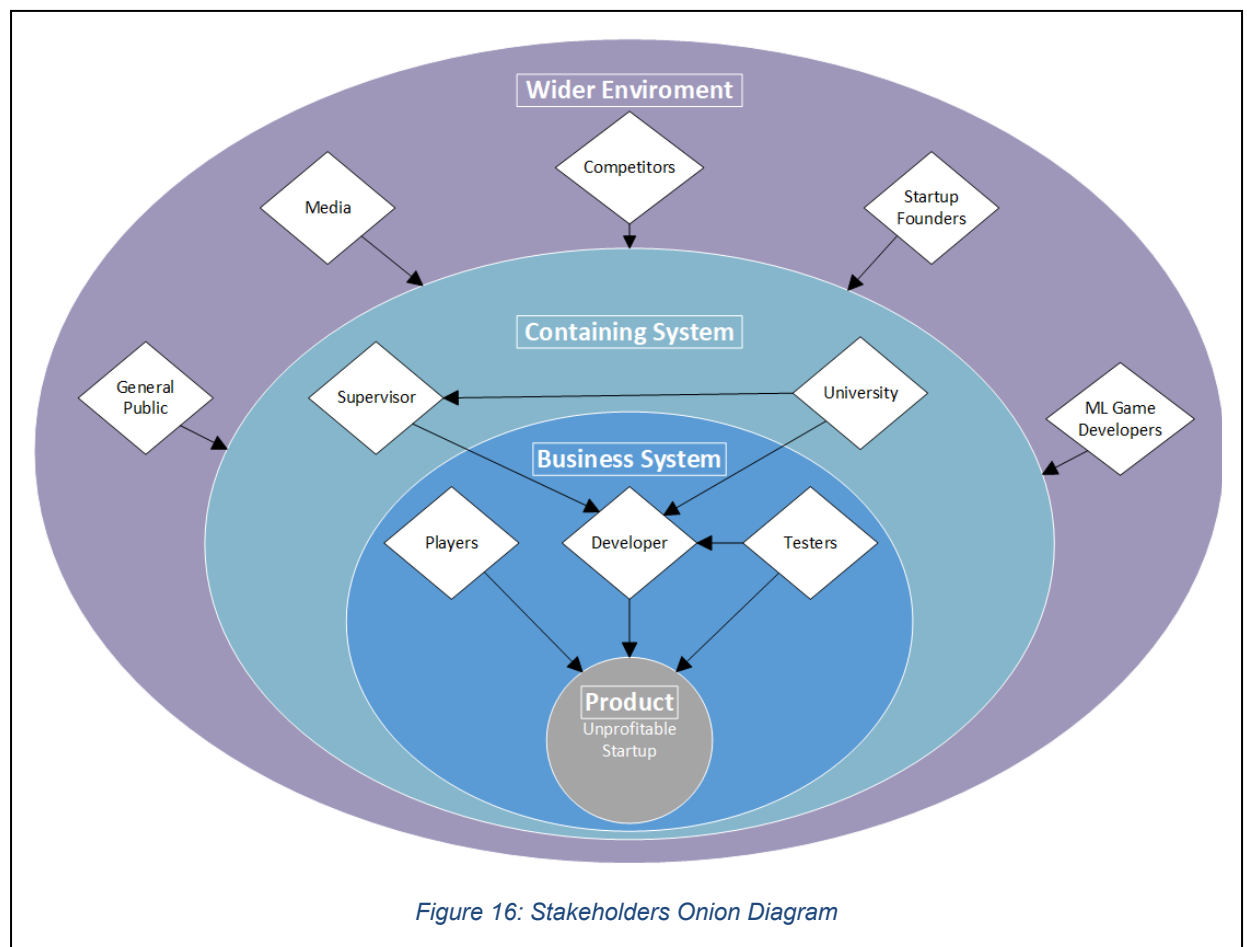
Table 4: Financial Data API Comparison

3. Requirements

3.1 Stakeholders

The onion diagram showcases the stakeholders in the layers for:

- Product: The Unprofitable Startup project itself.
- Business System: Actors directly interacting with the product, like the developer implementing the project; the testers that play the game and communicate feedback to the developer; and the players that purely enjoy playing the game.
- Containing System: The actors that control or host the product, and that are functional beneficiaries of it (who the project is being made for). The university sets major requirements and deadlines for the project, specifying the responsibilities of both the developer (student) and the project supervisor. The supervisor guides the developer throughout the project, approves the project area, and helps define its scope after the developer has proposed these.
- Wider Environment: This includes the actors in the environment where the project is released, including financial beneficiaries, negative stakeholders (competition) and other actors that may observe the product but do not interact with it directly. Depending on if this project is publicly released, this can include the public, the media, and competitors in the game market. Because of the topic of the project, it could potentially also include startup founders or people interested in a company's financial management, as well as Machine Learning enthusiasts who may want to know about ML projects integrated into video games.



3.2 Elicitation requirements

The requirements have been gathered through a Google Form.

The form collected 58 overall responses, as per the Participation Information Sheet, not all questions in the form had to be answered, so some questions may have less than the overall number of responses.

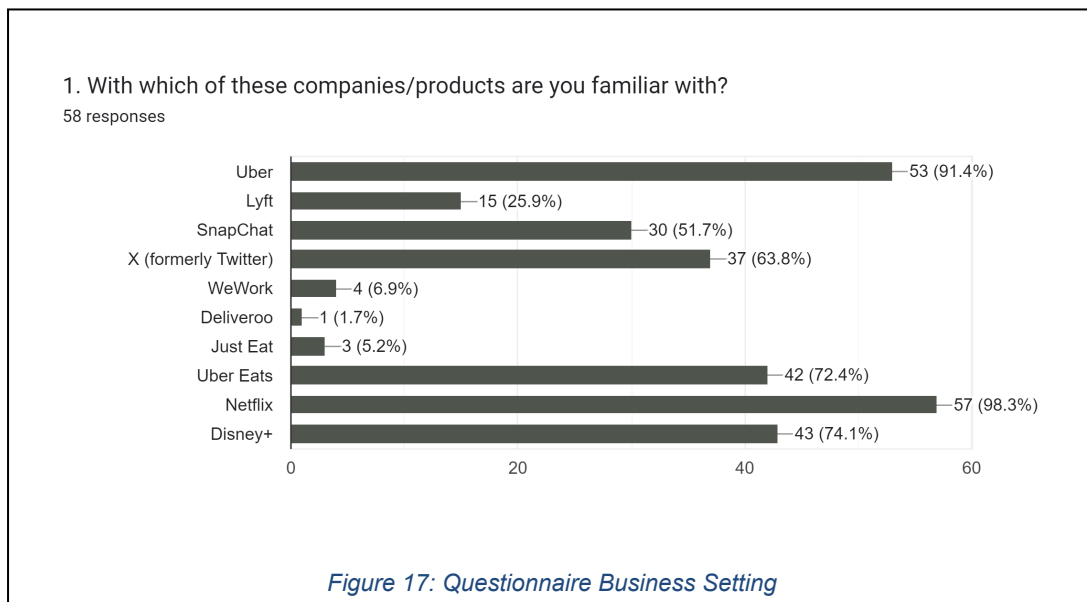
For the purposes of analysis, the form was divided into the following sections:

3.2.1. Business setting

As it is unrealistic to develop a simulation that can work for any type of company, especially due to the data training requirements of the neural networks, these questions help guide which type of company will be managed by the player. This can affect factors such as the art style of the game, or the type of choices the players will make in the game.

The results show that the video streaming companies, such as Netflix and Disney+, are the most known companies. Besides being known, per the questionnaire responses, a lot of people consider video streaming to be a profitable business venture, despite this generally not being the case (as the project name “Unprofitable Startup” implies) (Jeff Loucks, 2024).

This familiarity with the business will also favour the project from a game design/usability aspect, since it will be easier for users to understand how a company, they are already familiar with operates.



3.2.2. Comparable games

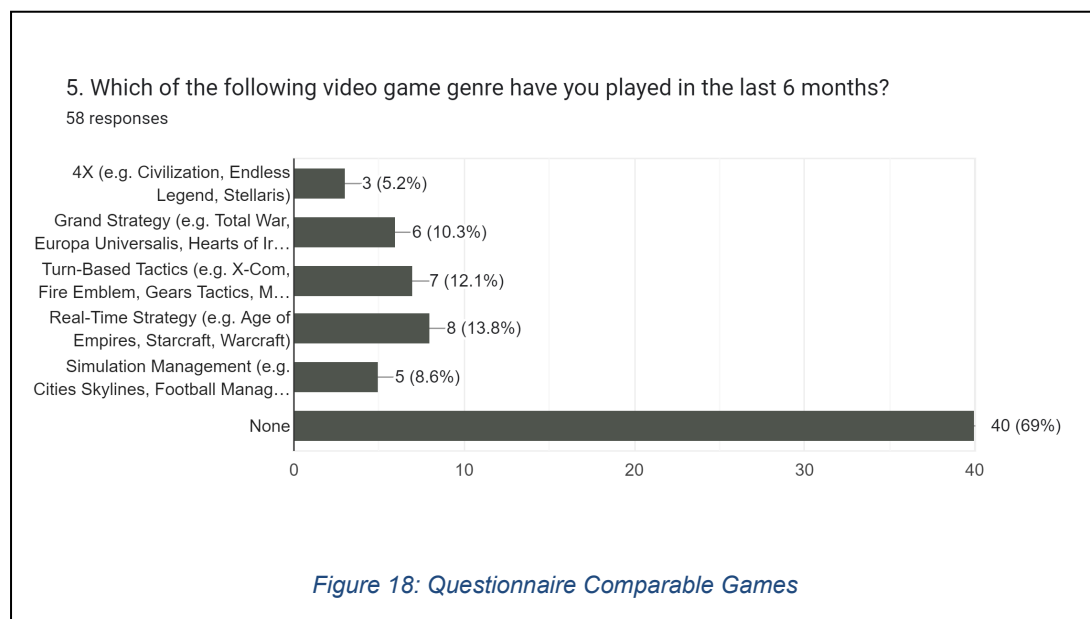
The comparable games included references to multiple game genres that can potentially have an economy simulator component. From this, we can determine the ideal game genre for the project.

Based on these responses, the Real-Time Strategy genre is the most popular and most well-known. However, since the real-time element would add unneeded

complexity to the project and would not highlight the simulation focus of the project, this would not be an ideal genre. Because of this, a turn-based game genre approach will be chosen to better highlight the simulation aspect of this project.

It is also to be acknowledged that while the game genres inquired about are not the biggest, noted by the considerable amount of people that responded not having knowledge of them, the aggregate data of them is significant. Commonly classified together under the “strategy” game umbrella, these are still profitable in the global in the global market (Wijman, 2024).

Analysing comparable games strategy games (Yee, 2016), and the strategy game genre as a whole (AARKI, 2020) the target audience of the game can be found. This indicates a slight majority of the male demographic in the 27-34 age range. Further analysis of comparable game sales and marketing could further narrow down this user demographic details.

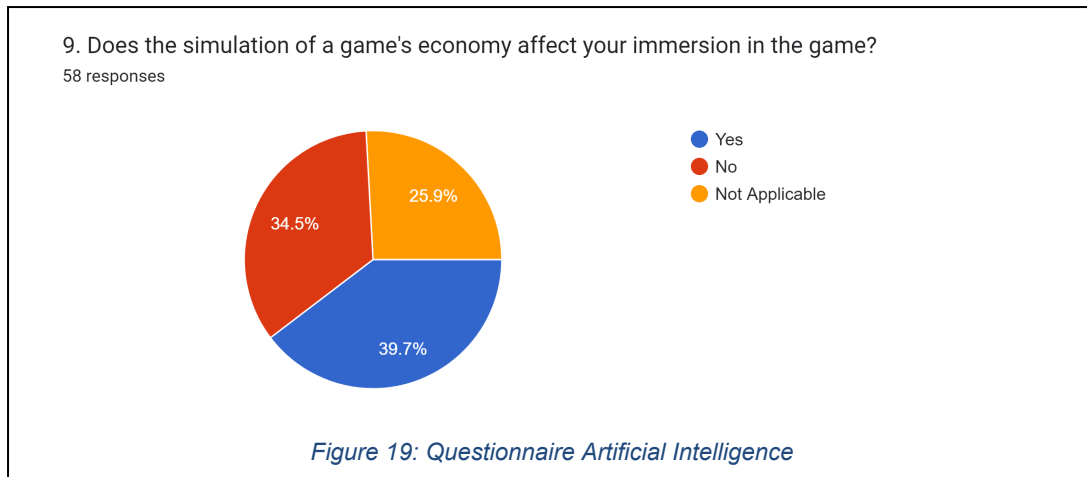


3.2.3. Artificial intelligence

These questions aimed to measure people’s experience with Artificial Intelligence in games. These responses allow for the measurement of weak points in current AI behaviour and possible knowledge gaps in their implementation.

Though most people had experiences ranging from average to positive, there is a non-insignificant amount of people who have had a negative opinion of AI. This can cause issues when most people in the survey consider that a bad game economy simulation affects their immersion in the game.

Somewhat surprisingly, people appear to have had a positive experience with the AI performance when operating independently of the player in a simulation-like environment.



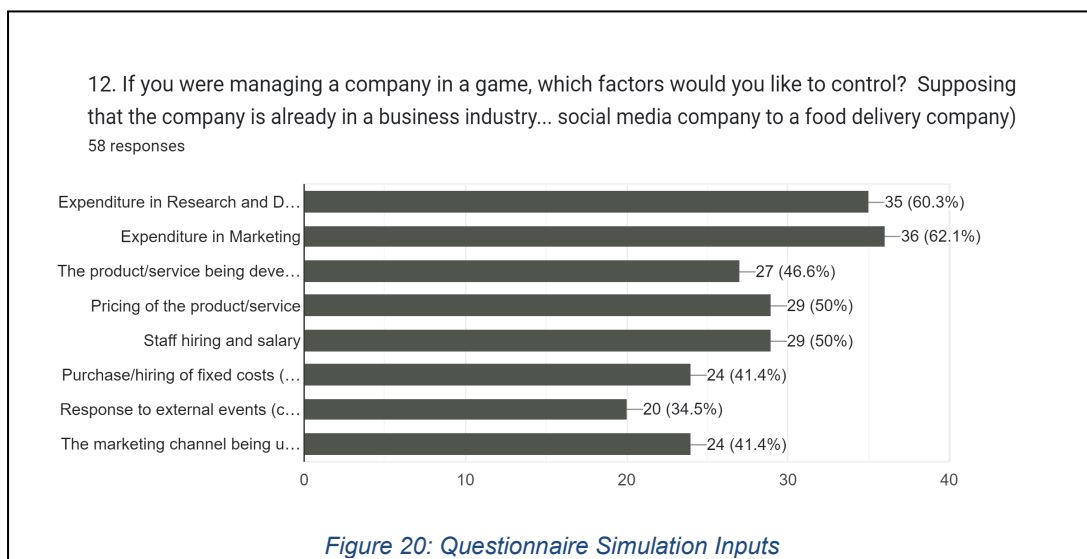
3.2.4. Simulation inputs

As previously mentioned, the objective of the economy simulation is to be believable to the player's rather than being a hundred-percent accurate. For this purpose, these questions aim to gauge people's opinions of what makes a company profitable.

These questions were used to determine:

- The factors that people perceive affect a company's profits, which will be used as inputs in the simulation algorithm. The options presented to respondents matched known factors that affect a company's income (Gaikwad, et al., 2023).
- The ideal time frame a company must make a profit, which will be used to control the timing of decisions in the game and the simulation.
- The elements that a player would like to control in a company for it to be able to generate profits.

Because some these factors are not pure objective data that can be used to train a neural network, the mix with gamification elements may be used to control how these affects the simulation. Like having them act as a multiplier or aggregate to the profit value calculated by the simulation neural network (Daniel de Almeida Rocha, 2019).



Full survey responses in Appendix 2 – Elicitation requirements responses

3.3 List of requirements

The requirements are separated into functional requirements, features that need to be implemented, non-functional requirements, conditions that the project must meet, and for this project, data requirements, which define how the data for the neural network is obtained and handled.

All the requirements are given a priority corresponding to their importance for development, which can be:

- Essential: Must be implemented for the project
- Desirable: Would be good to implement for the project
- Luxury: Only to be implemented in the project if there is enough time

3.3.1 Functional requirements

No.	Requirement	Description	Priority
1	Net Income simulation based on player's actions	The player's actions throughout the game and the current state of the company he is running in the game are considered to simulate the company's net income.	Essential
2	Manage budget	The player can manage the company's cash flow, choosing what areas to invest money in. At the start of the game, the player will start with a fixed amount of money in their budget.	Essential
3	Turn-based actions	The player can manually advance the time in the game (per turn) to progress the simulation further.	Essential
4	Animated background	The game's background shows the animation of a working company, even if the game actions are limited to a 2D UI layer.	Desirable
5	Main Menu	The game has a main menu screen to start the game, display controls, and show the credits.	Desirable
6	Pause Menu	The game can be paused, opening a separate screen, and from there, the player can resume or exit the game.	Desirable
7	Movies and Series Projects	The player can choose to start one-off projects; with the video streaming theme, these would be movies or	Desirable

		series productions that can affect the revenue produced after a money and time investment.	
8	Sounds and music	The game has sound effects for player actions and background music.	Desirable
9	Financial data tooltips and help section	There are easily accessible tooltips and a special guide section to explain to the player the meaning of different financial terms and how they may affect the company.	Desirable
10	Performance representative animated background	The animated background showing the player's company changes depending on the company's overall performance, showing a more modern-looking background when the company is doing well or a more rundown background when the company is doing poorly.	Luxury
11	Debts issuance and repayment	The player can get loans in the game and handle their repayment (monthly payment amount, etc.). This statistic is considered during the company simulation.	Luxury
12	Auto-run mode	The game time is continuously run without the player having to end a turn manually.	Luxury
13	Random events	Occasional random events can affect the company's immediate performance and force players to make specific choices in response to them.	Luxury
14	Options Menu	Options Menu accessible from both the Main Menu and In-Game that allows tweaking different non-gameplay settings, like sound volume.	Luxury

Table 5: Functional Requirements

3.3.2. Non-Functional requirements

No.	Requirement	Description	Priority
Usability			
1	Accessible financial data	The game data, including all the financial statistics, are shown in a way that can be easily visualised.	Essential
2	Intuitive to use	UI clearly shows what actions the user can take and how the game is played	Essential
Performance			
3	Optimal performance	The game runs at a consistent FPS, with no noticeable stuttering or lag.	Essential
Architecture			
4	Neural Network is used for simulations.	A neural network model is used at runtime by Unreal Engine to produce the profitability forecasting.	Essential
5	The Neural Network model is stored as an ONNX file	From a training framework (PyTorch, TensorFlow or other.), store the Neural Network model as an ONNX file to enable it to interface with Unreal Engine. (Breviu, 2022)	Essential

Table 6: Non-Functional Requirements

3.3.3. Data requirements

No.	Requirement	Description	Priority
Training Data			
1	Gathering Financial Data	Sufficient, accurate, and unbiased financial data is collected to train the Neural Network for the company income forecasting. When needed, this data will be scaled or transformed to be appropriately processed.	Essential

Table 7: Data Requirements

3.3.4. Heuristic analysis

These are the requirements related to some of the heuristics for the player's user experience (Koeffel Christina, 2010).

Though the game covers most of the user experience 29 heuristics in some manner, it mainly focuses on:

- Game Play/Game Story (9 Heuristics): 1, 3, 4, 6, 7, 8, 9, 10, 13.
- Visual Interface (5 Heuristics): 19, 20, 21, 22, 23.

No.	Heuristic	Application	Related Requirements
Game Play / Game Story			
1	The player should be presented with "clear goals" early enough or be able to create her own goals and "should be able to understand and identify them". There can be "multiple goals on each level", so that there are more strategies to win. Furthermore, the player should know how to reach the goal without getting stuck.	<p>The game goal, making the company profitable, is clearly shown in a special pop-up message at the start of the game. The in-game UI reinforces this by highlighting the company's income statistics throughout the playthrough.</p> <p>The game only has 1 "level"; therefore, no different goals per level exist.</p> <p>The in-game help section and a clear UI layout will guide the player toward reaching this goal.</p>	Non-functional :1, 2

2	The player should receive meaningful rewards. "The acquisition of skills" could also be a reward.	<p>The player will be rewarded intrinsically by the fact that they managed to turn a poorly performing company into a successful one.</p> <p>This is especially true since the company is relatable to real-life companies that most people are familiar with.</p> <p>In-game, the game's aesthetic will change as the player's performance, measured by the company's profits, improves.</p>	Functional : 10
3	The player should "feel that she is in control". That includes the "control over the character" as well as the "impact onto the game world". "The controls should allow management that is appropriate to the challenge. "Changes the player makes to the game world should be persistent and noticeable". Furthermore, the player should be able to "respond to threats and opportunities".	<p>The player will feel in control by ensuring the user's actions have meaningful effects on the company simulation.</p> <p>The player actions, like managing the budget investments, will serve as clear and noticeable inputs for the simulation neural network.</p> <p>Random events, where the player is forced to make a choice that may affect the company budget or other statistics, will allow the player to respond to different threats and opportunities.</p>	Functional : 1, 2, 7, 11, 13
4	"Challenge, strategy and pace should be in balance". "Challenges should be positive game experiences".	While the player will start the game with the company having a negative net income, he will also have ample available funds at his disposal to alleviate any pressure of the situation, giving him freedom of what actions to take.	Functional : 2
5	"The first-time experience is encouraging".	The player starts the games with enough cash flow to provide freedom of choosing what they want to do while providing room for errors.	Functional : 9

		The in-game UI will further encourage the player on how to perform best.	
6	The “meaningful game story should support the game play” and be “discovered as part of the game play”.	<p>The game setting of a company that, per the questionnaire responses, is well-known should prove relatable to the player.</p> <p>The game will provide an insight into how this company operates. The player will further explore the inner workings of this company as he continues his playthrough.</p>	<p>Functional : 1, 2, 7</p> <p>Non-Functional : 1</p> <p>Data: 1</p>
7	“The game does not stagnate” and the player feels the progress.	<p>The game background changes as the player progresses, indicating the player’s performance and how this has affected the company being managed.</p> <p>Special projects, which can be individually managed, and random events allow the player to make unique decisions outside of the usual budget management decisions.</p>	Functional : 7, 10, 13
8	The game should be consistent and “respond to the user’s action in a predictable manner”. This includes “consistency between the game elements and the overarching settings as well as the story”. The story should “suspend disbelief” and be perceived as a single vision, i.e. the story should be planned through to the end.	<p>The predictability of the simulation, due to the black box element of the neural networks, makes this one of the biggest challenges. The simulation neural network model is thoroughly tested with multiple data inputs.</p> <p>A cap on how much the net income can change, negative or positive, in a certain amount of time will help the game be fairer and prevent any outlier occurrence.</p> <p>Though the game has no narrative story per se, the simulation based on real-life data should present the player with a believable progression of the company’s performance.</p>	Non-Functional : 4

9	<p>“It should be clear what’s happening in the game, the player should understand failure conditions and be given space for making mistakes”</p>	<p>The UI clearly presents the financial data, and an additional help section guides the player on what the different data means.</p> <p>The UI will present running out of money as a clear failure condition, with a special notification shown when this is close to running out.</p> <p>The effects of a simulation iteration will be capped to a certain percentage, meaning that even if the player has a bad turn, it will not totally ruin the playthrough, providing room for mistakes.</p>	<p>Functional : 9</p> <p>Non-Functional : 1</p>
10	<p>“There should be variable difficulty levels” for a “greater challenge”. The game should be “easy to learn, but hard to master”</p>	<p>The scope of the project does not prioritise providing different difficulty levels. In the future, this could be added by affecting stats like the player’s starting budget and modifying the multipliers of the player’s decisions.</p>	
11	<p>The game and the outcome should be perceived as being fair.</p>	<p>The Neural Network will emphasise appearing believable, per user expectations, rather than being completely accurate.</p> <p>Game design provides different weighting to the neural network inputs based on their relevance to the user.</p>	<p>Non-Functional : 1</p> <p>Data: 1</p>
12	<p>The game itself should be replayable and the player should enjoy playing it. Nevertheless “challenging tasks should not be required to be completed more than once”. The challenge should create the desire to play more. This also includes the possibility to skip nonplayable and repeating content if not required by the game play.</p>	<p>The game will contain a certain degree of randomness to make each playthrough unique, therefore making the game more replayable by not having to do the same thing each playthrough.</p> <p>Furthermore, since there is no obvious “correct” path to achieve the game’s goal, the player will be encouraged to try to improve his performance in subsequent playthroughs.</p>	<p>Functional : 1, 2, 13</p>

		The in-game help section, the only non-playable part of the game, can be easily skipped without the player having to go through it again.	
13	“The artificial intelligence should be reasonable”, “visible to the player, consistent with the player’s expectations” and “yet unpredictable”	<p>The Neural Network, the artificial intelligence component in the game, aims to provide believable net income simulations for a company.</p> <p>Probability modifiers will be applied to the player’s input and the simulation output to make each playthrough unique.</p>	<p>Non-Functional : 1</p> <p>Data: 1</p>
14	The game should be “paced to apply pressure to but not frustrate the player.	<p>The game will start the player with a significant starting budget for him to make decisions. As the game progresses and this budget dwindles, the pressure on the player will increase.</p> <p>However, this would also likely result in the company losses being reduced each turn. This clear progress will avoid frustrating the player, since each turn, he will be able to see the effect of its actions (no frustration due to sudden randomness)</p> <p>Besides this, random events, which could further add up the pressure, will only happen after a certain number of turns once the player has already had a handle on the game.</p>	Functional : 2, 13
15	The “learning curve should be shortened”. The “user’s expectations should be met” and the player should have “enough information to get immediately started”. Tutorials and adjustable levels should be able to involve the player quickly and provided upon request throughout the entire game.	<p>The player’s primary goal and objectives will be presented through a starting game notification, which can be skipped.</p> <p>Tooltips in the relevant company statistics will always be available to easily show the player what his decisions may cause.</p> <p>A help section will always be available in the menu to further</p>	<p>Functional : 9</p> <p>Non-Functional : 1, 2</p>

		explain game concepts in more detail.	
16	“The game emotionally transports the player into a level of personal involvement (e.g. scare, threat, thrill, reward, punishment)”.	Per the questionnaire responses, the game setting is based on a company relatable to the user, with which they may feel a certain degree of personal connection. This connection can get the player more involved with the positive outcome of the company.	
17	The game play should not require the player to fulfil boring tasks”.	<p>The player will be tasked with making macro decisions when managing a budget rather than worrying about minor details.</p> <p>This ensures the player will only have to make a few choices with a significant impact rather than many small decisions that may cause negligible effects.</p> <p>Some legal requirements for managing a company, like declaring taxes, will be omitted from the game to focus on just the company’s core business decisions.</p> <p>The player’s choices regarding the budget allocations will carry over to the next turn. This means that instead of having each turn allocate the budget from scratch, the player will be able to just adjust the previous turn's budget. This allows for easy progress through multiple turns at once without having to do the tedious task of reallocating the budget to the same values each time.</p>	Functional : 2
18	“The game mechanics should feel natural and have correct weight and momentum”. Furthermore, they should be appropriate for the situation the user is facing.	The player’s actions, represented as inputs in the company’s simulation, are based on real-world data. This, along with the data used to train the neural network of the simulation, should make them feel natural.	Functional : 1, 2 Data: 1

		The effects, or weight, of each of these actions, will be relative to how much people perceive, per the questionnaire responses, these affect a company's performance.	
Visual Interface			
19	“The player should be able to identify game elements such as avatars, enemies, obstacles, power ups, threats or opportunities”. The objects “should stand out, even for players with bad eyesight or colour blindness and should not easily be misinterpreted”. Furthermore, the objects “should look like what they are for”.	The data and controls are clearly presented to the player as 2D elements over a 3D background. This highlights and clearly separates what game elements are interactable and/or relevant.	Non-Functional : 1, 2
20	The “acoustic and visual effects should arouse interest” and provide meaningful feedback at the right time. Hence the effects should give feedback to create a challenging and exciting interaction and involve the player by creating emotions. The feedback should be given immediately to the user's action	The background representing the company will change depending on how the company is doing. Audio will indicate when the players do an action, providing immediate feedback.	Functional : 4, 8, 10
21	The interface should be “consistent in control, colour, typography and dialog design” (e.g. large blocks of text should be avoided, no abbreviations) and “as non-intrusive as possible”.	The UI design, colour scheme, and font will be consistent throughout the game. Data will be presented at the edges of the screen and open as a pop-up window only when more detail is required.	Non-Functional : 1, 2
22	The player should not have to “count resources like bullets, life”, score, points, and ammunition. This “relevant information should	Main statistics, like the net income (the game's goal), are always shown on the main game screen.	Non-Functional : 1

	<p>be displayed, and the critical information should stand out". Irrelevant information should be left out. The user should be provided enough information to recognize her status and to make proper decisions. Excessive micromanagement by the user should be avoided.</p>	<p>The game only contains and shows the relevant data that has a direct effect on the simulation. Many of these data, which may not be necessarily shown in the main game screen, can be seen in further detail in the submenus.</p>	
23	<p>The menu should be "intuitive and the meanings obvious" and "perceived as a part of the game".</p>	<p>The 2D menu over the 3D game background will highlight the interactable elements in the game.</p> <p>The Help section will further guide the player on what some of the statistics of the game mean.</p>	<p>Functional : 9</p> <p>Non-Functional : 1, 2</p>
24	<p>"The player should know where she is on the mini-map if there is one and should not have to memorise the level design".</p>	<p>The game does not have a mini-map or a level design component.</p>	
25	<p>The player "should be able to save the games in different states" (applies to non-arcade-like games) and be able to "easily turn the game off and on".</p>	<p>Due to the project's scope, there will be no persistent data saves.</p> <p>However, the player will be able to access a pause menu to stop and continue.</p>	<p>Functional : 6</p>
26	<p>"The first player action is obvious and should result in immediate positive feedback".</p>	<p>A notification at the start of the game explains the main goal and gameplay mechanics to the player. Once this is closed, the player is immediately taken to the Manage Budget submenu.</p> <p>This will make the action of allocating the budget evident to the player. Though the player will be able to see the allocation effects on his budget amount, the results of these allocations will not be visible until the player turns end.</p>	<p>Functional : 9</p> <p>Non-Functional : 1. 2</p>
27	<p>Input methods should be easy to manage and have an appropriate level of</p>	<p>Mouse clicks will be the game-only input method.</p>	<p>Functional : 14</p>

	sensitivity and responsiveness. The input methods should allow customization concern the mappings.	Because of the scope of the project, no other input method or input mapping will be supported.	
28	The visual representation (i.e. the views) should allow the user to have a clear, unobstructed view of the area and of all visual information that is tied to the location.	The main game view will only show the most significant, summarised financial data. More detailed information about other financial statistics can be viewed in separate views, exclusively dedicated to that statistic (do not mix unrelated data).	Non-Functional : 1
29	The game should allow for an appropriate level of customization concerning different aspects.	The scope of the project does not make this heuristic a priority.	

Table 8: Heuristic Analysis

3.4. Requirement analysis and modelling

3.4.1. Context diagram

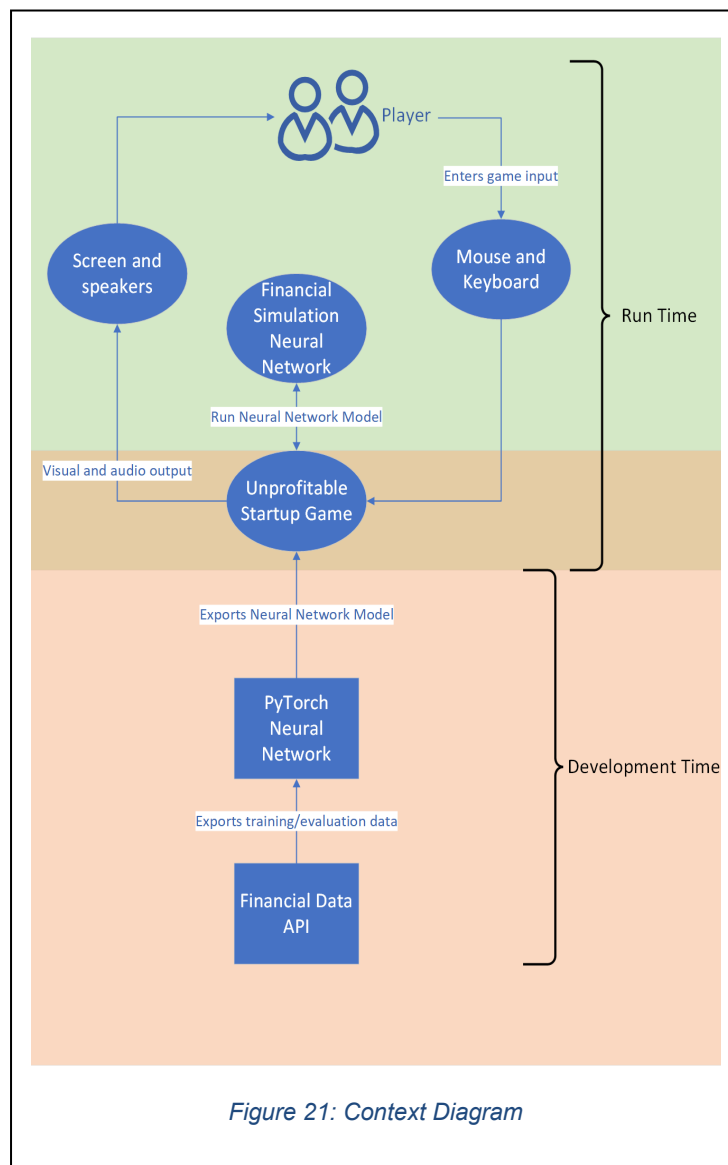


Figure 21: Context Diagram

The Unprofitable Startup game will not interact with any significant external entities at runtime. Instead, it will leverage the Unreal game engine abstractions to handle the input from the player and the game's visual and audio output.

The game loop of receiving and processing the player input and outputting image and sound will be done by the game locally. Also done at run time from within the game, the Neural Network model will be used for the game simulations to predict the player's company Net Income.

During the development stage, before the project is compiled for the end user, an external neural network framework, PyTorch, will be used to train the Neural Network so that it can be exported to Unreal Engine. The data to train the neural network will be obtained from an external financial API and exported into PyTorch.

3.4.2 Use cases

The use cases diagram shows cases of the possible actions the player, the main actor of the application can take throughout the game. These are considered design documents and include functionality not included in this version of the game but covered by 9.3. Further works.

3.4.2.1 Start the game

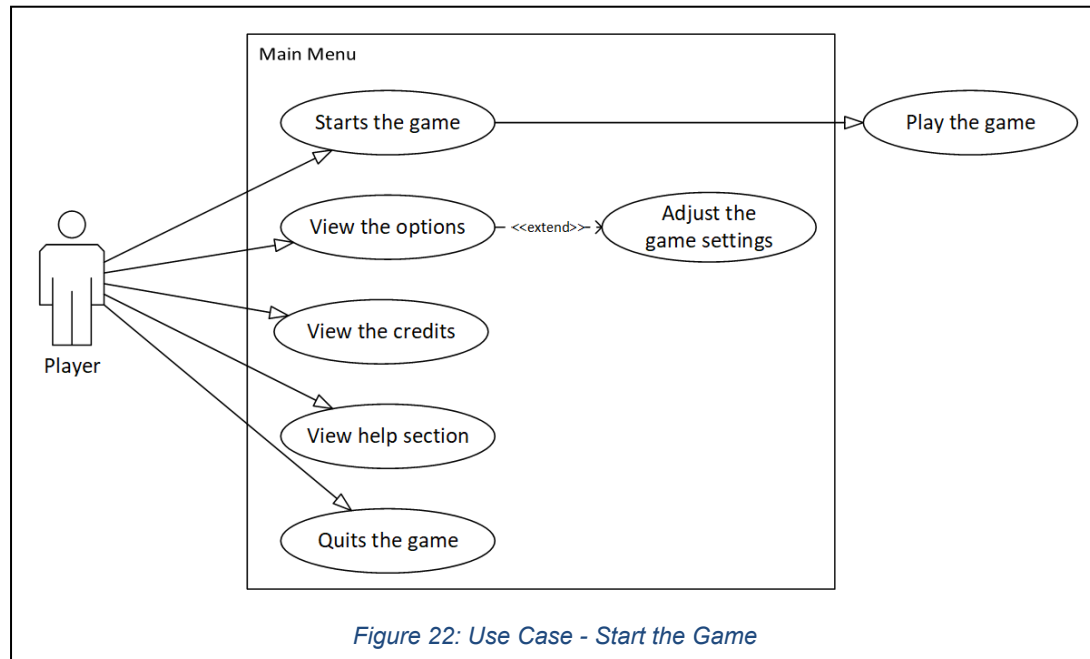


Figure 22: Use Case - Start the Game

Use Case	Start the game
Description	The player starts the game from the Main Menu.
Actor(s)	Player
Pre-Condition	The game is in the Main Menu.
Basic Flow	<ol style="list-style-type: none"> 1. The player launches the application. 2. The player selects to start the game. 3. The game scene is loaded, and the game begins
Alternative Flow	The option to exit the game is selected.
Post-Conditional	The game is started, or the application is closed.

Table 9: Use case - Start the Game

3.4.2.2 In-Game

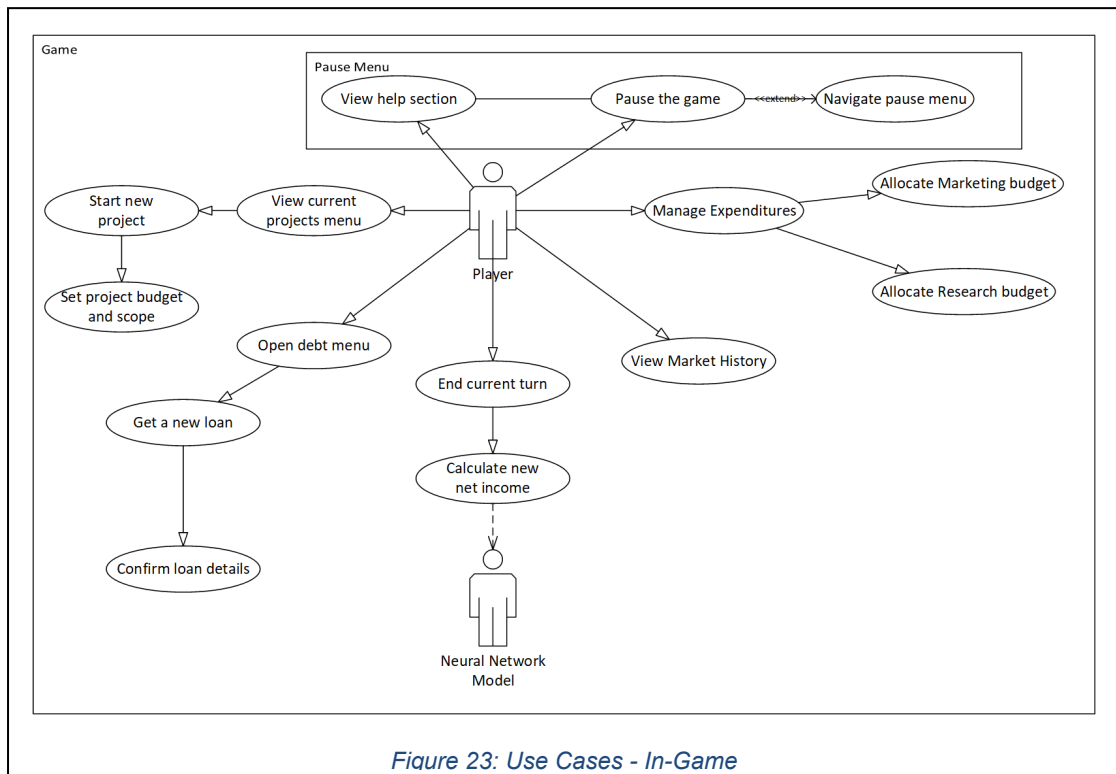


Figure 23: Use Cases - In-Game

Use Case	End the current turn
Description	The player ends the current turn, allowing the game simulation to update the net income value.
Actor(s)	Player, Neural Network Model
Pre-Condition	A game has been started.
Basic Flow	<ol style="list-style-type: none"> 1. The player ends the current turn. 2. The Neural Network Model calculates the new net income from the player's action during the turn and his current company statistics. 3. A new turn is started
Post-Conditional	The next turn in the game is started with updated net income calculations.

Table 10: Use Case - End the Current Turn

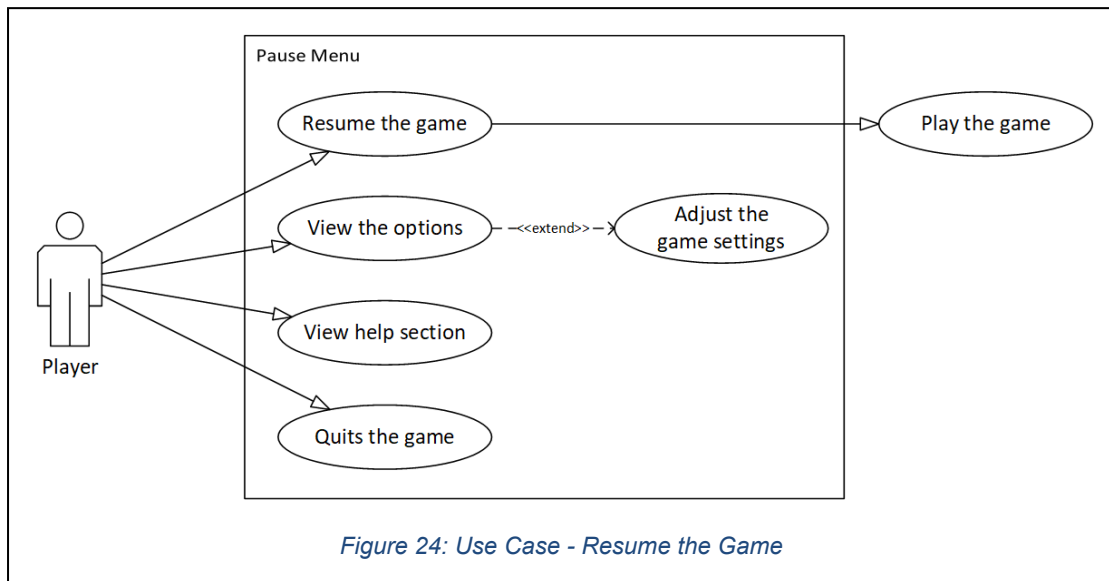
Use Case	Get a Loan
Description	The player gets a new debt by borrowing money.
Actor(s)	Player
Pre-Condition	A game is currently active.
Basic Flow	<ol style="list-style-type: none"> 1. Open the debt menu. 2. Select the get a new loan option. 3. Confirm the loan details (interest rate, loan amount, monthly payment, etc.).
Alternative Flow	The player does not find a satisfactory loan condition and leaves the menu without borrowing money.
Post-Conditional	The debt menu is closed, and the player gets a new debt paid due starting the next turn.

Table 11: Use Case - Get a Loan

Use Case	Start a new project
Description	The player starts a new project to develop and invest in.
Actor(s)	Player
Pre-Condition	A game is currently active, and the player has sufficient funds for a project.
Basic Flow	<ol style="list-style-type: none"> 1. View the current projects being executed. 2. Select the option to start a new project. 3. Set the project scope and budget
Alternative Flow	The player has insufficient funds and leaves the menu without starting a new project.
Post-Conditional	The current project's menu is closed, and the player's company has a new project to manage.

Table 12: Use Case - Start a New Project

3.4.2.3 Pause menu



Use Case	Resume the Game
Description	The player resumes playing the game after it has been paused.
Actor(s)	Player
Pre-Condition	The game is currently paused.
Basic Flow	The player clicks the Resume button from the pause menu to continue playing.
Alternative Flow	The player quits the game.
Post-Conditional	The next turn in the game is started with updated net income calculations.

Table 13: Use Case - Resume the Game

4. Legal, social, and ethical issues

One of the main concerns from the rising popularity of neural networks is the legal issues related to the data used to train the neural network (Walsh, 2023).

For this project, the financial statements of Netflix, the streaming company, were used to train the neural network. Though these statements are publicly available, as required by Netflix's status as a publicly traded company (Hayes, 2021), the legal framework for this area is not well defined.

However, it should be noted that since this is data from a public company and does not involve any personal data from people nor any form of authored content (drawings, photos, videos, etc.), this is unlikely to be an issue for either privacy laws or any future legal framework for machine learning data collection concerning copyright.

In terms of third-party assets used to develop the game, such as 3D models, art, and sound, all of them were properly obtained through legal means and adequately licensed for personal and commercial use. Though most of them do not require any attribution due to their license, if the project were to be further developed, a further analysis of what to include in a "license" or "credits" section of the game may be needed.

Name	Creator	Type	Link
Raleway	Matt McInerney; Pablo Impallari; Rodrigo Fuenzalida;	Font	https://fonts.google.com/specimen/Raleway
Big Office	1D.STUDIO	Meshes/Textures	https://www.unrealengine.com/marketplace/en-US/product/big-office
60 game cursors	cruizRF	Sprites	https://www.gamedevmarket.net/asset/60-game-cursors
Modern Lo-Fi Ambient - music pack	cyberleaf	Music	https://www.gamedevmarket.net/asset/modern-lo-fi-ambient
320 Interface and Item Sounds	GameDev Market	SFX	https://www.gamedevmarket.net/

Table 14: Third Party Assets List

It should also be noted that this project and its neural network design are used for recreational purposes and should not be considered an appropriate simulation for making financial decisions in real life. Though unlikely to cause issues, this distinction may be worth mentioning in the game to clearly indicate that the game should not be replicated in real life. This will help prevent any confusion with business simulation games that are used for training purposes (Semler, 2014).

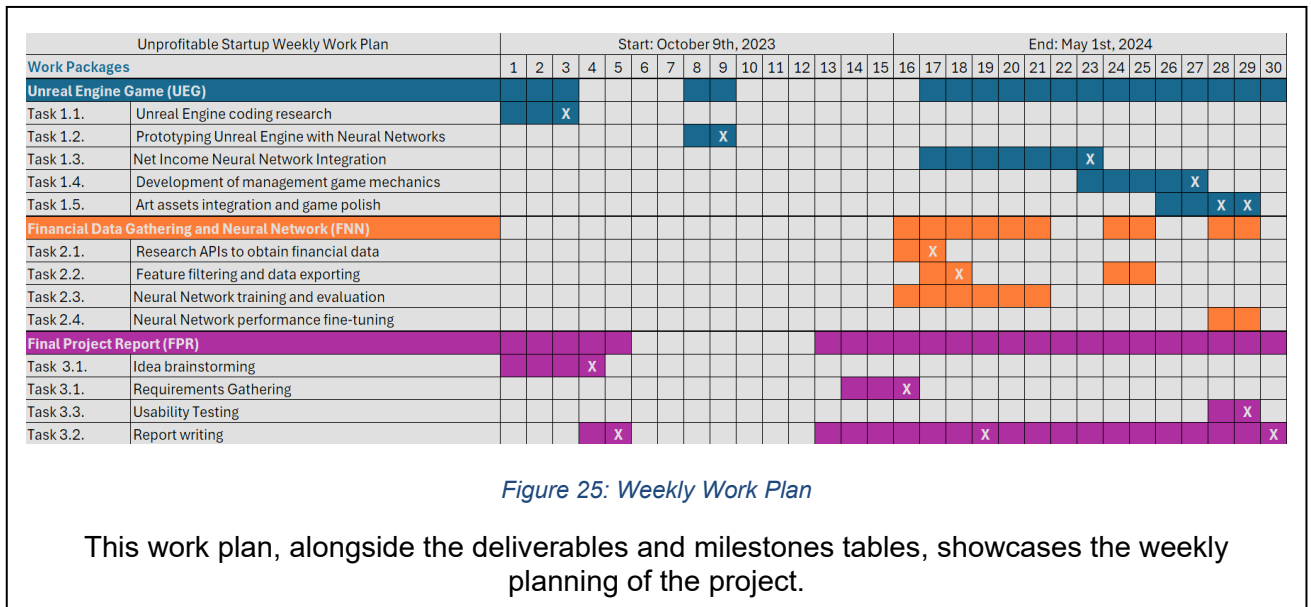
5. Methodology

The methodology for this project consists of a mixture of both waterfall and agile. The waterfall method was used for the initial prototypes of the project. These prototypes had no actual iteration, with their main objective being to test conclusively if the project's central concept of integrating a neural network with Unreal Engine at runtime was viable.

Subsequently, after the prototypes confirmed this idea was possible for the project, an agile scrum methodology was followed for the project's final development.

5.1. Workplan

Despite following agile, certain main milestones corresponding to the different deliverables in the project were planned in the following work plan. This is because the deliverables were fixed dates established by the stakeholders that needed to be accomplished. Because of this, the project scope had to be constantly adjusted for each agile sprint to meet these deadlines.



Deliverables	Content
D1.1 (M1.2)	Formative Project Idea: 3 November 2023
D1.2 (M1.3)	Project Proposal (PP): 9th November 2023
D2 (M2.5)	Project Specifications Design and Prototype (PSDP): 15 February 2024
D3.1 (M3.2)	Formative Progress Update: 28 February 2024
D3.1 (M3.6)	Final Project Report: 1 May 2024

Table 15: Work Plan Deliverables

Milestone	Week	Work Package	Criteria for milestones to occur
M1.1	3	UEG	Blog posts for using Unreal Engine with C++
M1.2	4	FPR	Project idea finalised
M1.3	5	FPR	Project proposal
M2.1	9	UEG	Prototype of neural network with Unreal Engine
M2.2	16	FPR	Requirement gathering survey
M2.3	17	FNN	Financial statements can be obtained via API
M2.4	18	FNN	Prototype of training neural network to predict the net income
M2.5	19	FPR	Project Specifications Design and Prototype
M3.1	23	UEG	Net Income neural network can be imported and run in Unreal Engine
M3.2	27	UEG	Features complete gameplay
M3.3	28	UEG	The game has art and sound
M3.4	29	UEG	Usability improvements have been implemented
M3.6	30	FPR	Usability testing forms collected
M3.7	30	FPR	Final Project Report

Table 16: Work Plan Milestones

The work plan details the project's main deliverables, as defined by the stakeholders, which in this case includes the project proposal, PSDP report, and the final project report.

Following the agile methodology, each weekly sprint underwent its own smaller scale, planning and testing phase. At the end of a sprint, the tasks done in the previous sprint were reviewed, the project's current state was reevaluated, and the work for the next week was planned.

It should also be noted that the work plan has noticeable “gaps”, where no significant progress was made in the project due to time commitments with other projects.

5.2. Prototypes

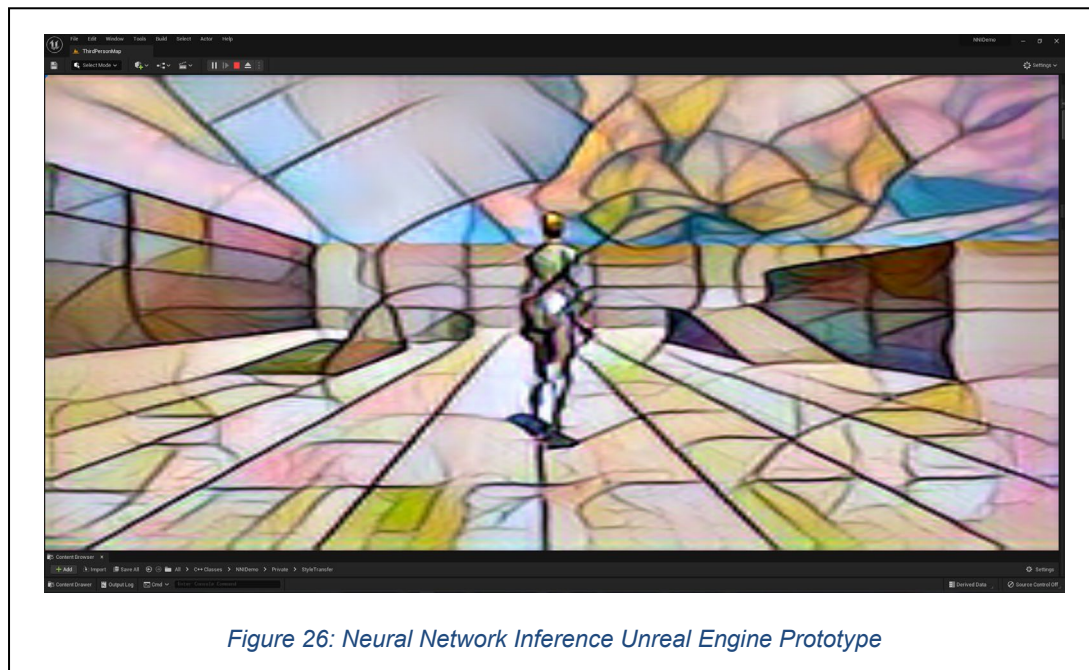
For the prototypes, the main technological requirements were first defined, per the project proposal, as creating a neural network in Python to predict a net income and loading and running a neural network at runtime in Unreal Engine.

These prototypes aimed to test if the project idea was viable to execute, the importance of which was significant since this project involved me working with both technologies currently in development, not officially released, as well as technologies that I was not highly familiar with.

There was no real iteration for these processes since these were not aimed at making a game but instead testing the viability of certain game technologies that would be the cornerstone of the project. Because of this, a waterfall approach was best suited since I only wanted to develop the idea in a straightforward manner, and I did not have to concern myself about any feature creep or game-balancing issue.

5.2.1. Neural network inference Unreal Engine prototype

YouTube Link: <https://youtu.be/niKxpE9DWhI>



This prototype is a recreation of the example shown in the original Microsoft Developer talk that introduced the use of Neural Network Inference with Unreal Engine (Breviu, 2022). This inference is done via ONNX models, which serve as an abstraction between the trained neural network models and the run-time game engine.

The prototypes use a Neural Network to alter the appearance of the image being shown on the screen, like a post-process effect, depending on the selected model. The models used are the same as those employed in the Microsoft Developer talk example.

The output of the prototype itself was not necessary. Instead, the development of the prototype had two main objectives:

- Test the viability of running a Neural Network at run-time in Unreal Engine. This was particularly concerning because much of the technology is still in the experimental stage and is currently being developed.
- Understand how Neural Networks in Unreal Engine work and understand their technical limitations. This is to observe how the data is inputted and outputted.

The prototype was done using the experimental Neural Network Inference (NNI) plug-in for Unreal Engine 5.0, which was replaced in Unreal Engine 5.2 and onward by the Neural Network Engine (NNE) plug-in (Unreal Engine, 2023).

5.2.2. Financial neural network prototype

YouTube Link: <https://youtu.be/vrPkeIIAYyU>

The objective of this prototype was to create a Neural Network that could forecast a company's Net Income, as this is one of the project's main components.

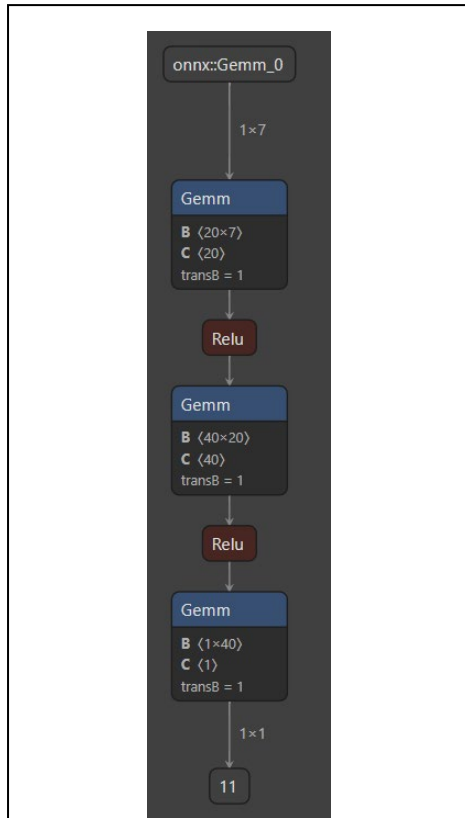


Figure 27: Financial Neural Network Prototype Output

The output of this prototype is a neural network model, in ONNX format, that can forecast the net income.

As per the questionnaire responses, a company in the video streaming business was chosen to model the data. Since Netflix is the most popular platform and the one with less diverse sources of income (it does not have other significant business ventures besides video streaming) (Durrani, 2024), its financial data was chosen to train the Neural Network.

The first step of this process was obtaining the required financial data to train the neural network model. Though different APIs to obtain the financial data were obtained, the prototype tested 2:

- [yFinance](#): An open-source Python API that obtains financial data directly from Yahoo! Finance by scraping the data in their website (Aroussi, 2024). The main issues with this API were that it had not implemented an endpoint to collect a company's earnings (the neural network output) and did not return enough historical data to train a neural network (returning just the latest 5 data entries).
- [Alpha Vantage](#): A free API to obtain financial data. This worked based on an HTTP request, returning a JSON output with the financial data (Alpha Vantage, n.d.). This provided different endpoints for different data (cash flow, income statements, stock price, etc.) from a significant number of years on a quarterly basis.

Because it provided access to the company's income and a longer record of the historical financial data, the Alpha Vantage API was chosen.

The financial data was saved as a JSON and filtered only to contain the data of the last ten years. This is to account for the rise of the video streaming market, since before, Netflix's leading business venture was renting DVDs via mail.

Through a Python script, the relevant fields of the data, obtained from multiple endpoints, were then selected, interpolated from quarterly data to monthly (to get more data to train the neural network) and finally saved as a single CSV file.

Which fields were selected to be used as inputs in the Neural Network was determined through a combination of the questionnaire responses (Simulation Inputs sections) and other research papers about using Machine Learning to predict a company's profit (Gaikwad, et al., 2023) (Vijh, et al., 2020). Based on this, the relevant data was:

- Fiscal Date Ending
- Operating Cash Flow
- Capital Expenditures
- Gross Profits
- Total Revenue
- Cost of Revenue
- Operating Expenses
- Net Income
- Stock Value

Though not covered in this prototype, many of these fields will be reinterpreted or weighed in the game based on different gameplay mechanics.

The data was then scaled and used to build and train a neural network using PyTorch. The “Net Income” field was set as the neural network’s output, with all the other fields serving as inputs. The performance of the neural network was then evaluated using different metrics (such as MAE, MSE, RMSE, and MAPE (Brownlee, 2020)), before finally being exported as an ONNX file to be able to use it in Unreal Engine (Breviu, 2022).

It should be noted that this prototype aimed to produce a working neural network that is reasonably accurate. However, more iterations regarding the neural network inputs and architecture could be done and measured further to improve the prediction accuracy for the final release.

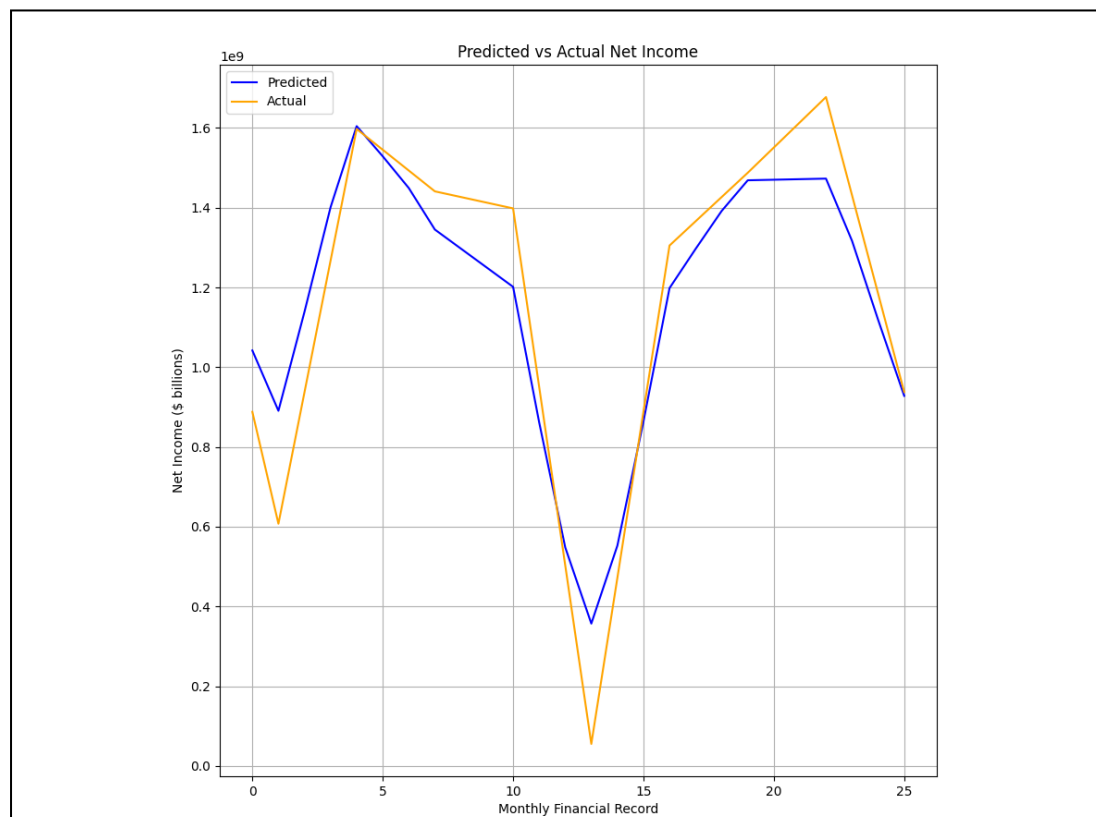


Figure 28: Prototype Predicted vs Actual Net Income

Though the accuracy of the neural network in this prototype must be further improved, in its model it seems to predict the trends in the net income.

5.2. Agile methodology

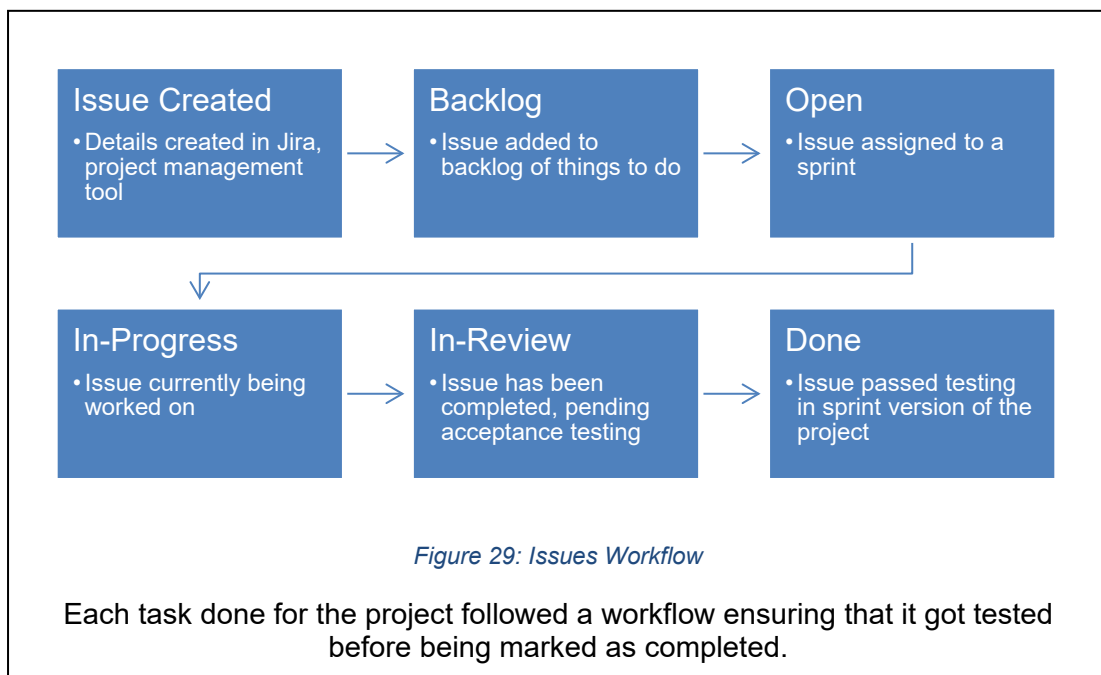
The agile methodology was chosen for this project to be able to adapt to any unexpected challenge. For this project, I decided to work on a couple of new technologies, so it was expected that some of my time estimations for developing certain features would be wrong. Besides this, the difficulty of capturing a “fun” gameplay mechanic makes it challenging to estimate a development time. So, the flexibility of agile allowed me to organise my priorities and project timeline as the development progressed.

One of the main focuses of agile is having short iteration processes. For this project, I aimed to have one week’s sprints, at which point I would have a new build for the game to test through. Then, before the start of the next sprint, I would evaluate what still needed to be done for the project and plan what to do for the next sprint.

This meant that at the end of the week, I could see significant progress in the project and be able to detect any major issues on the overall project. It also meant that I could tag each weekly sprint on GitHub, my version control software of choice. So, in case of any major bug or regression, I could spot in which version the issue started occurring and narrow down the changes that may have caused it or, in the worst-case scenario, revert to that working version.

It should be noted that as I was working by myself, instead of with a team, I used a significantly simplified version of Agile, obviously without any sort of group meeting, pair programming, daily stand-up, or other traditional Agile techniques. This also helped reduce the overhead of any of the processes so that I could focus more on development.

During development, I created “issues” (individual case reports for bugs/features) for all the tasks which needed to be completed. These issues then followed a workflow of assigning them to a sprint, developing them, placing them in review, and marking them as done after testing them in the sprint build.



To aid myself in the management of the project, I used the Jira development tool, a project management tool widely used through the software and game development industry (Hall, 2018). This tool allows me to create different issues, keep track of what needs to be done, and visualise the sprints.

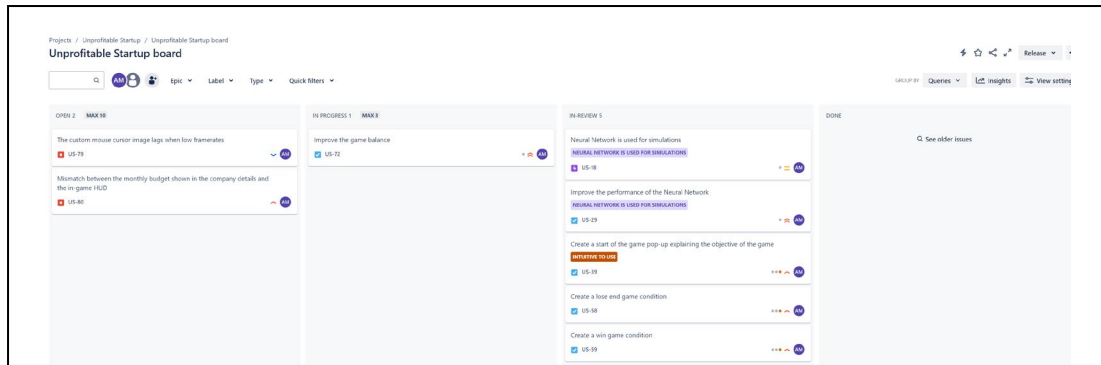


Figure 30: Jira Scrum Board

The Jira scrum board was used to keep track of development during each sprint, and measure that all the tasks followed the proper workflow.

Regarding prioritisation during development, the priority was creating and integrating the neural network into Unreal Engine, which was previously proved possible through the prototypes. This was then followed by developing the simulation gameplay mechanics. Finally, since the game has no real focus on level design, nor does it have a significant concern for the game's look, the last stage of the project involved adding different art and sound assets to the project.

Version	Status	Progress	Start date	Release date	Description
V0.8	RELEASED	[Progress bar]	April 23, 2024	April 30, 2024	Implemented the Help Menu and other user experience improvements.
V0.7	RELEASED	[Progress bar]	April 17, 2024	April 23, 2024	Add the end of game conditions and improve game balance.
V0.6	RELEASED	[Progress bar]	April 10, 2024	April 17, 2024	Add arts & sound
V0.5	RELEASED	[Progress bar]	April 6, 2024	April 10, 2024	Add Company Details screen and Pause Menu
V0.4	RELEASED	[Progress bar]	March 27, 2024	April 5, 2024	Create Expenditures Screen, and improved how the financial data is gathered and evaluated.
V0.3	RELEASED	[Progress bar]	March 20, 2024	March 27, 2024	Bug fixing, and setup the financial data truncation to thousands (k), millions (m), and billions (b)
V0.2	RELEASED	[Progress bar]	March 13, 2024	March 20, 2024	Base game to end the current turn, run the simulation, and update core UI
V0.1	RELEASED	[Progress bar]	March 6, 2024	March 13, 2024	Net Income Neural Network brought into Unreal Engine with respective test cases and test scene.

Figure 31: Jira Release Versions

At the end of each sprint, a new version of the game was built and tested. See [Appendix 4 – Versions release notes](#) for release notes on each version.

The UML diagram shows the code architecture of the most essential systems in the game, including their properties and methods and how they interact with each other.

The central game architecture will have the “UCCompany” class handling all the company's financial data, including running the financial simulation and storing company-specific data, like its current financial data and the monthly expenses.

The “UCFinancialNeuralNetwork” is the wrapper class around the neural network, used to set the neural network inputs and run the inference from it. This wrapper class also includes a reference to “UCFinancialDataScaler”, which is initialised using the scaling data exported from the training framework. This scaler is used to scale all the inputs to be processed by the neural network and unscaling its net income output.

The “UCMonthlyExpenseInvestment” is a non-specific investment type object, not linked to any specific “ECompanyMonthlyExpenseType”, used to store overall spending on that investment field and store the different “FSInvestmentLevel”. These investment levels require a certain amount of spending needed to unlock their bonuses.

The storage of the current financial data status and how it is passed between the different classes and to the neural network is handled by the “FSFinancialData” struct, which stores all the inputs of the neural network and its output.

“ACUnprofitableStartupMainGameMode” inherits from the Unreal Engine “AGameModeBase” class so that it is easily accessible in other scripts. This serves as a type of game manager, storing and controlling the reference to the player's company, triggering game-related events, and keeping track of the current game condition, marked by “EEndGameCondition”.

However, ACUnprofitableStartupMainGameMode has a further child class, “BP_MainGameMode”, a blueprint class (Unreal Engine visual scripting system). This was done purely to expose the class to the Unreal Engine editor, to simplify the process of hooking the UI, which is also done using Blueprints, and for easy access to variables related to the game balance, separately from the game code.

For readability, this UML omits references to the UI implementation done via Blueprints since this merely relies on getting information from the “UCCompany” or the “BP_MainGameMode” for its underlying functionality. This includes linking the UI to events defined in C++ for ACUnprofitableStartupMainGameMode, like “OnTurnEnded” or “OnEndGameConditionChanged”.

It should also be mentioned that to have these C++ classes integrate with Unreal Engine correctly, including with its visual scripting system, many inherit from core Unreal Engine classes, like “UObject”. This also provides access to the built-in engine garbage collection system, which is why no smart pointer variable is explicitly used.

6.2. User interface

Overall, the game UI will be a simplified version of similar simulation games, like Game Dev Tycoon, where the player visualises a “3D” background. Still, the relevant information will be presented in a 2D UI.

Even if the game is turn-based, it is important that the 3D background of the game itself has some animation or movement. This will help further distinguish it from the static 2D UI and give the game a sense of “life”. Otherwise, the game may feel like it is “stuck” without anything happening.

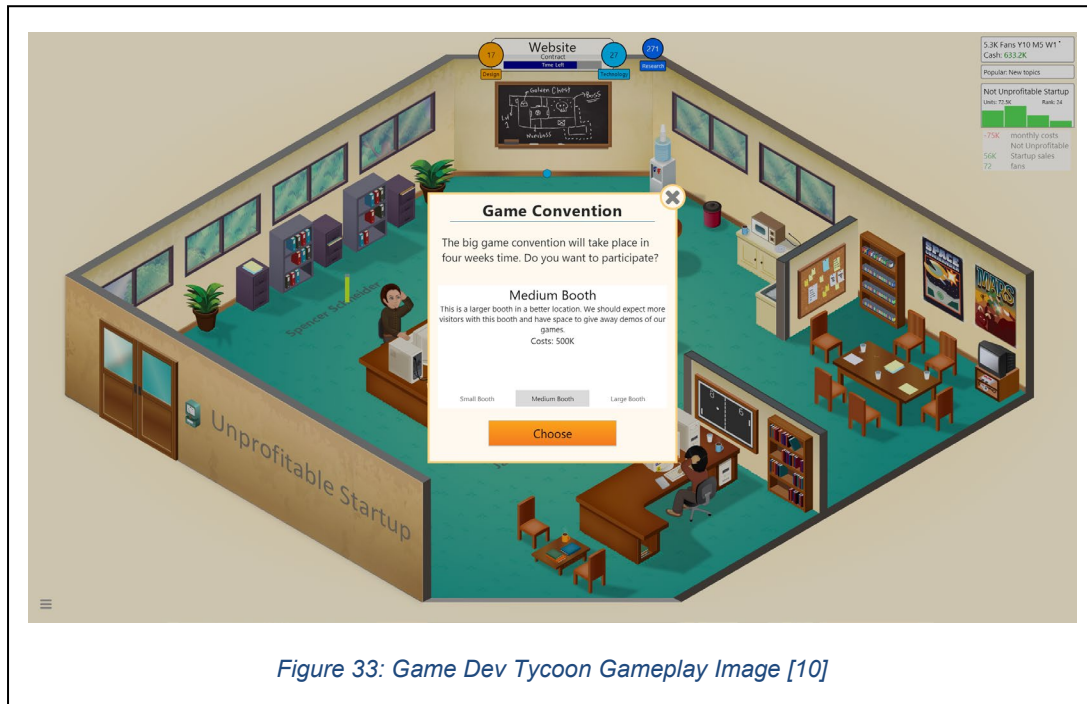
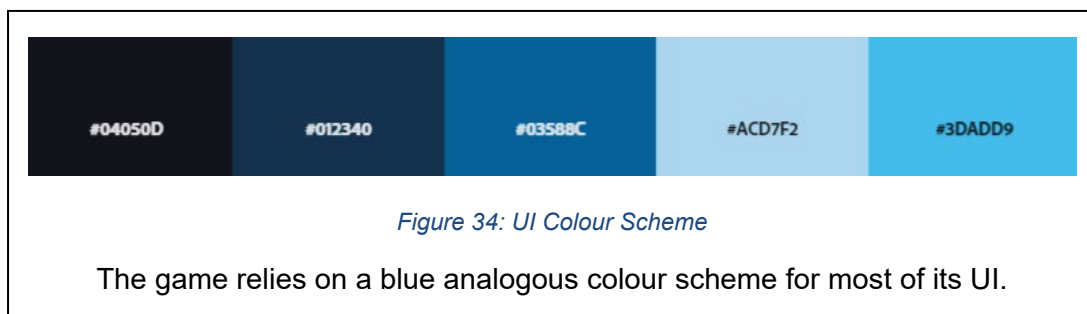


Figure 33: Game Dev Tycoon Gameplay Image [10]

In terms of the colour scheme for the game, most UI elements follow an analogous (colours next to each other in the colour wheel) colour scheme based on blue. Some notable elements, like the titles of menus, use a variation of an orange colour for highlighting since this is a complementary colour to blue.

To avoid issues with readability, all text presented always has a dark, almost black, background, with most of the colour of the text being a clear white.



The design of the buttons and UI in the game will also take inspiration from modern applications, such as Netflix, with a flat (no significant details) UI design. With these applications, the UI has no overly detailed UI element, with almost all of it having a simple colour as a design. This is to match the “unprofitable startup” (Durrani, 2024) condition that many of these companies have.

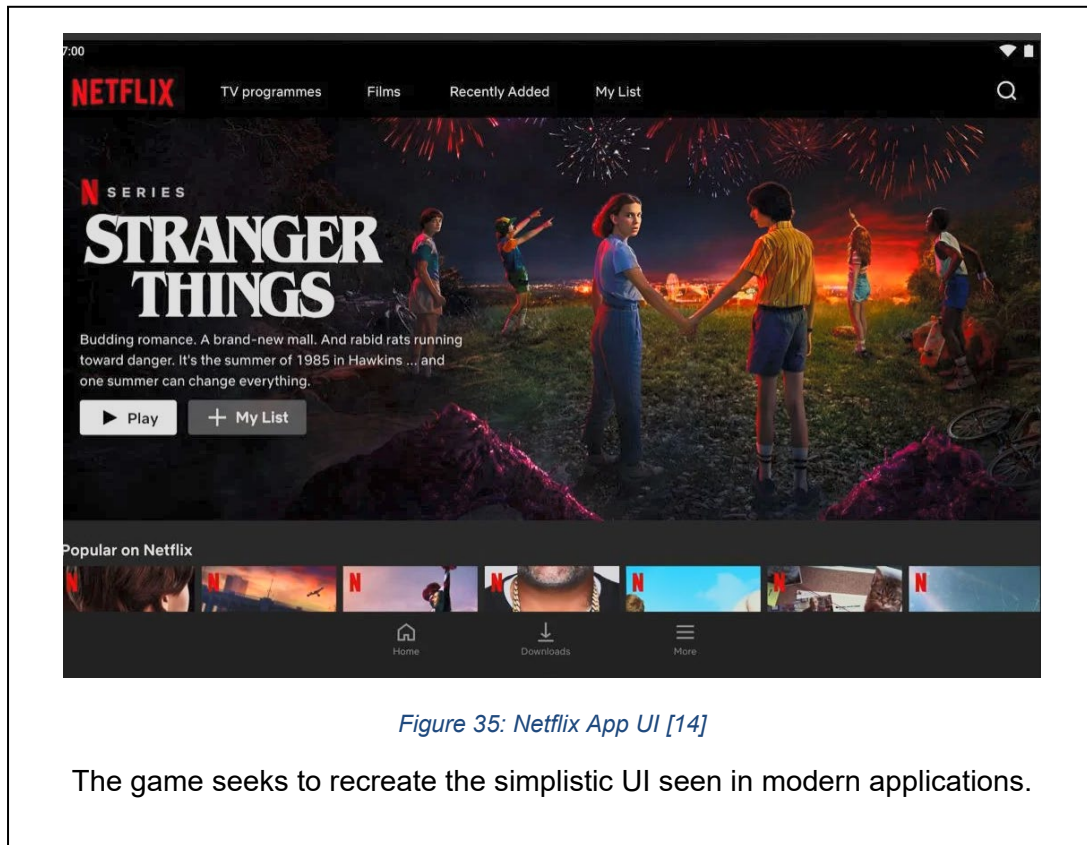


Figure 35: Netflix App UI [14]

The game seeks to recreate the simplistic UI seen in modern applications.

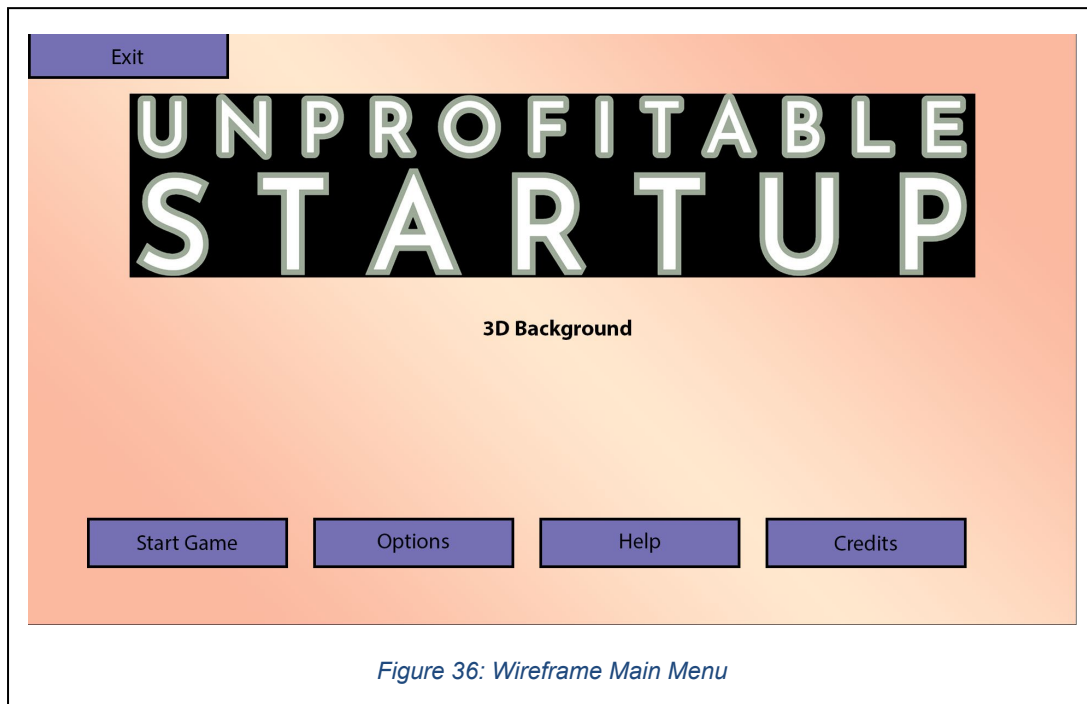
6.2.1. UI wireframe

Basic wireframe designs of the game menu and screens were developed to visualise the UI layout and, more importantly, how the information is presented, and the menus are navigated.

As wireframes, these designs used basic art assets, with the main focus being showcasing the UI elements' layout.

When applicable, if the screen is not pending future developments, the final version of the UI screens is shown after the wireframe to showcase how the wireframe helped decide the UI layout.

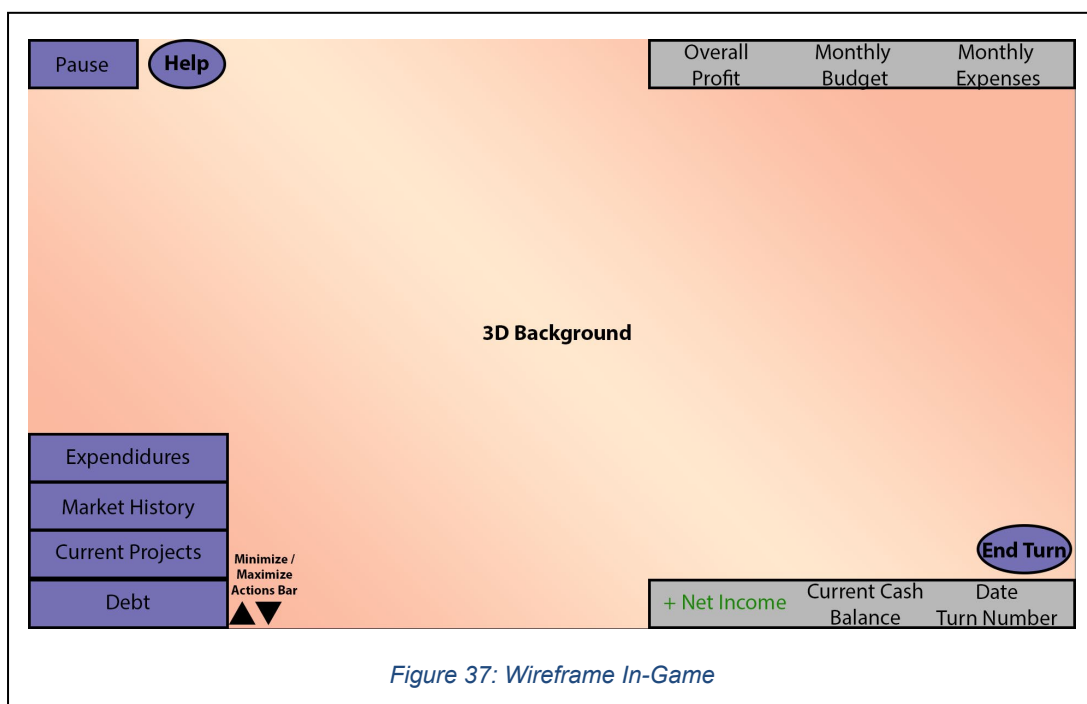
6.2.1. Main Menu



From the Main Menu, the player will be able to:

- Start the game.
- Open the Options sub-menu.
- Access the help/instructions section.
- See the game credits.
- Quit the application.

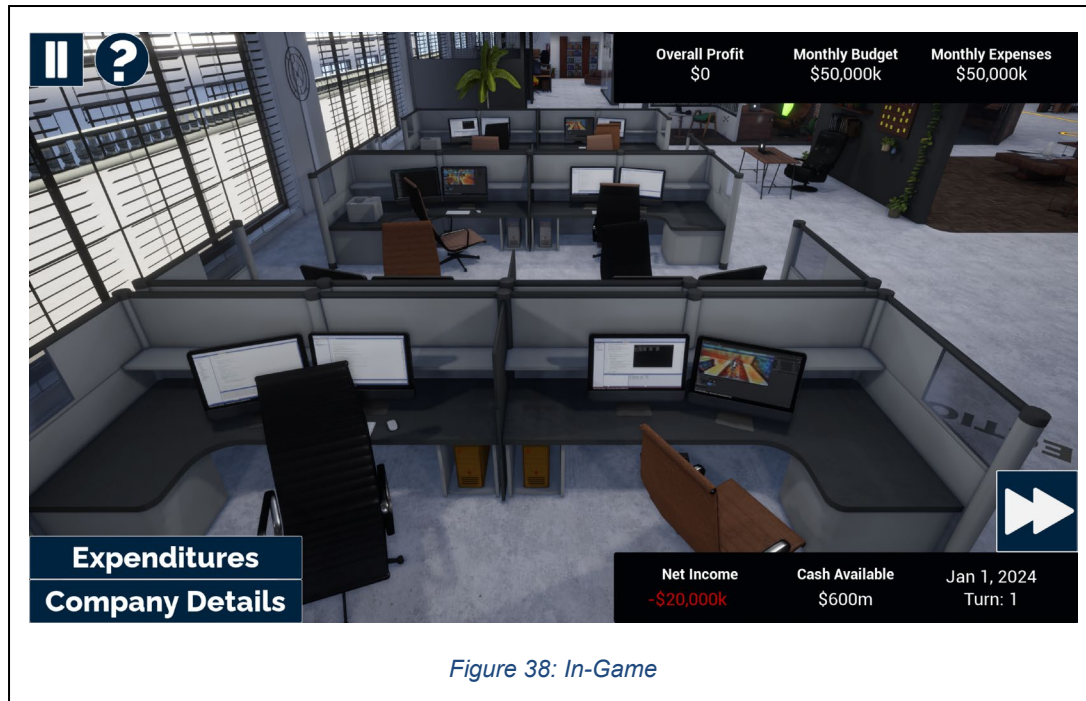
6.2.2. In-Game



The in-game HUD will have information-only data, which is non-interactable, on the right side of the screen. The actions the player can take will be visualised in an action bar menu on the bottom left of the screen. Pressing any of these actions' buttons will open a pop-up window with more details of the selected options.

The end turn button, since it is a significant action and with the objective of avoiding potential misclicks, will be placed by itself on the bottom right screen.

Non-gameplay-related actions, like the pause menu and the help section, will be placed on the top left side of the screen.



6.2.2. In-Game (Expenditures)

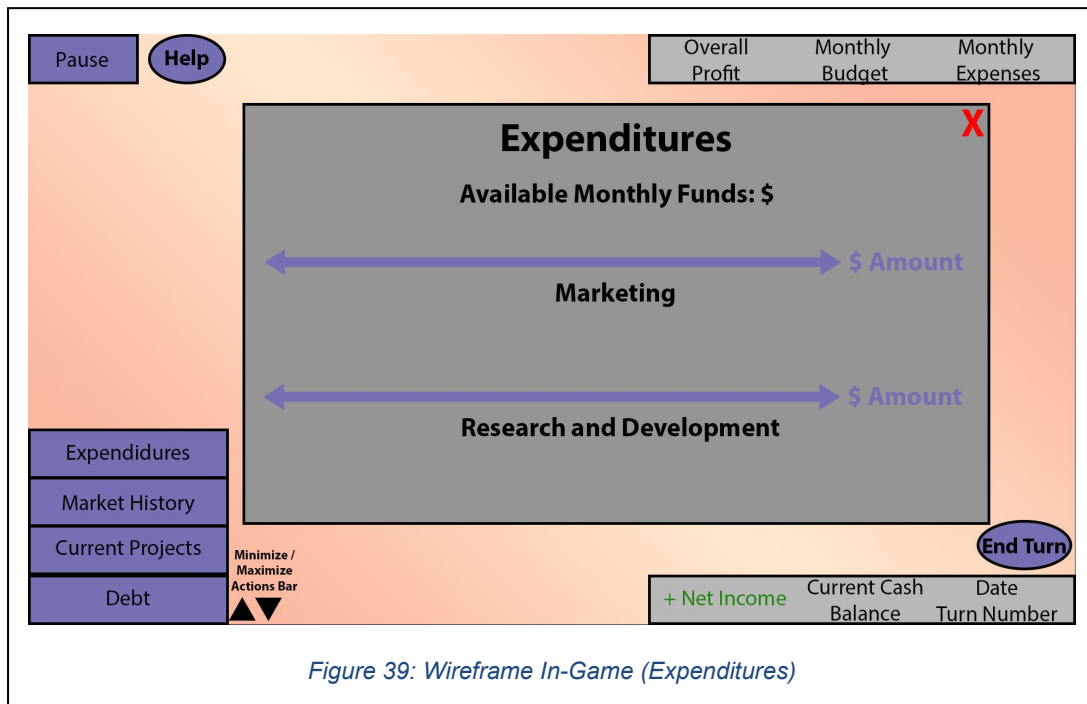


Figure 39: Wireframe In-Game (Expenditures)

The Expenditures window shows how the budget allocation mechanic would work by using a slider to control the amount of funds allocated. Other menus that must deal with allocating funds will have an interface like this.



Figure 40: In-Game (Expenditures)

6.2.3. In-Game (Market History)

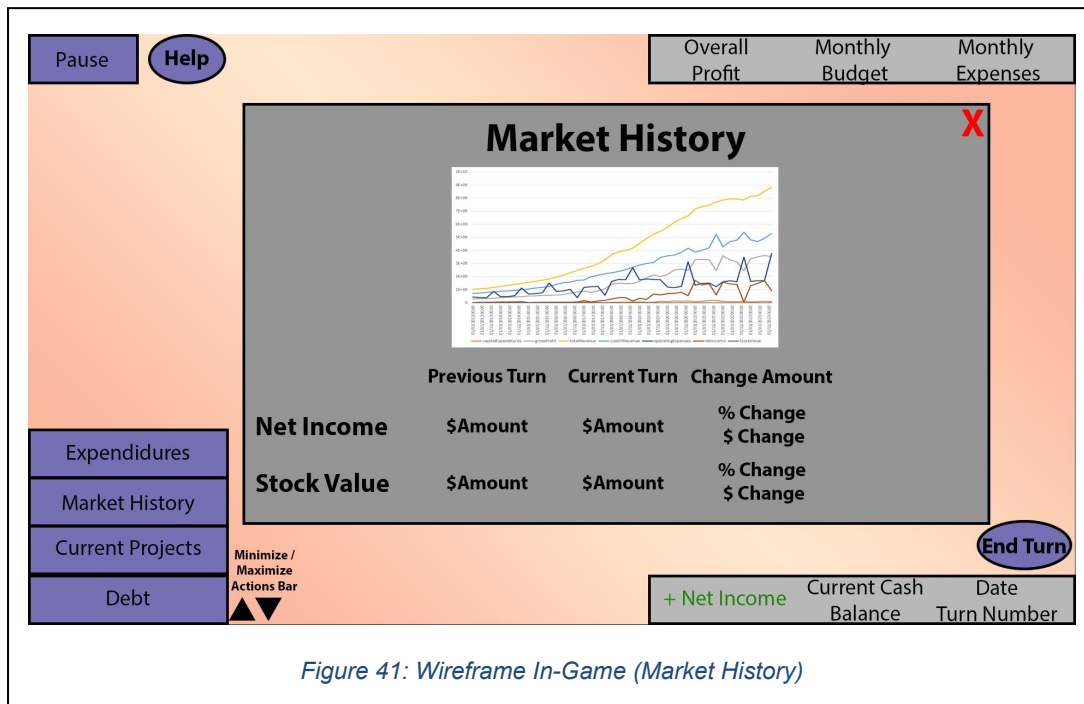


Figure 41: Wireframe In-Game (Market History)

The Market History window, which is not to take an action, is purely to visualise certain information more in-depth. This will focus on showcasing the change in financial statistics over time.

6.3. Neural Network Design

The main objective of the neural network is to, based on a company's financial status, be able to accurately predict the net income of a company. This caused two main concerns:

- Determining which data to use to train the neural network. This concern was one of the main reasons why the idea of simulating a company was chosen since companies have financial data publicly available that can be used for supervised learning. This contrasts with choosing to simulate an entirely fictional setting, where the training data would have to be generated in some manner.
- The feature selection of the neural network inputs, or which fields the neural network uses to predict the net income, with them having to be adjustable in the gameplay.

Per the [3.2 Elicitation requirements](#), the business setting best known to people is video streaming. Therefore, the financial data of a video streaming company should be used to train the neural network.

As some companies like Amazon or Disney have other business ventures and are not dedicated exclusively to streaming (Siddiq, 2024), using their data to train the neural network could skew the results by considering data from other business areas.

Because of this, it made sense to use the data of the company with the biggest market share (Durrani, 2024), which focuses almost exclusively on streaming, Netflix, to train the neural network. Though because until 2013, Netflix's business model also relied heavily on its mailing of DVDs business (Viki, 2023), no data from an earlier date of 2013 would be used.

Though as a public company, Netflix must publish certain financial information (Hayes, 2021), the information available presented challenges when deciding which data could be used to train the neural network. Since the best data may not be available in these financial statements.

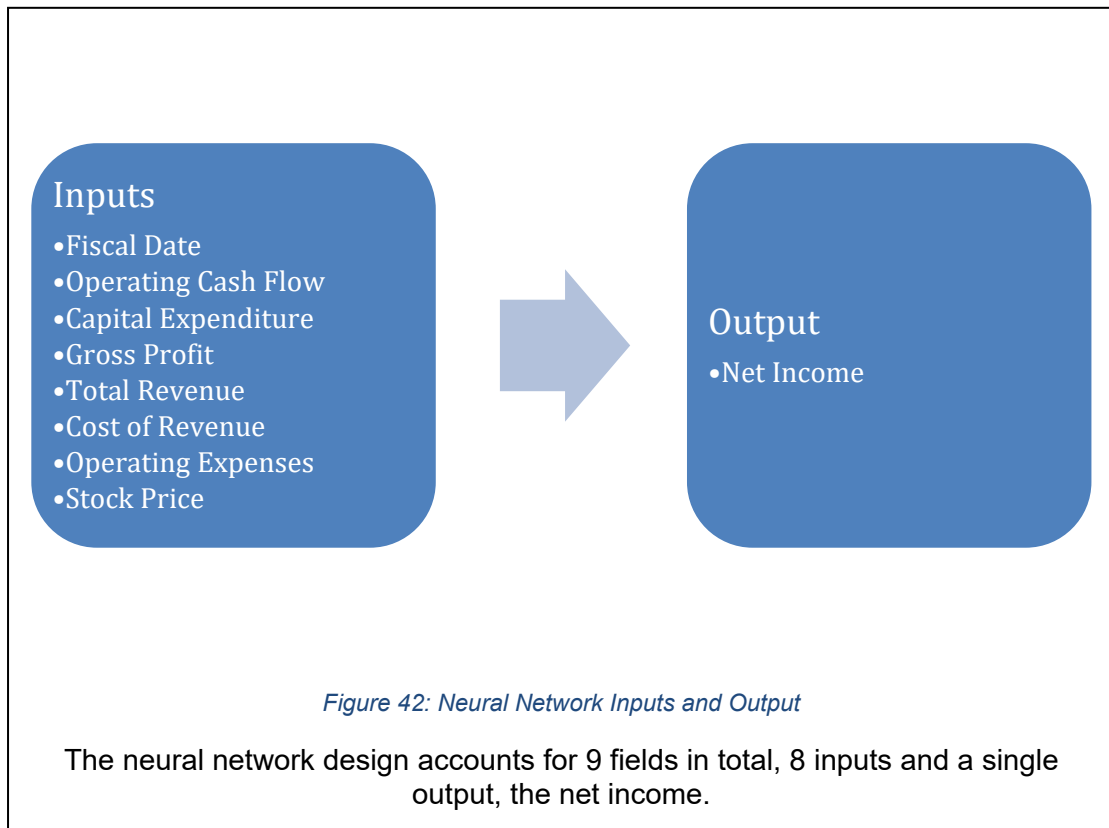
Based on the data available on the financial statements (see [Appendix 6 – Alpha Vantage Netflix financial statements samples](#)), different fields were chosen as inputs for the Neural Network. The fields chosen were based on the following:

- Previous literature survey to know the fields the net income is dependent from (Gaikwad, et al., 2023)
- Elicitation requirements survey to know the fields people both think affect the company and the areas they would like to have control over.

Following these parameters, the fields chosen from the available financial statements of Cash Flow, Income, and Stock price were (Hayes, 2021):

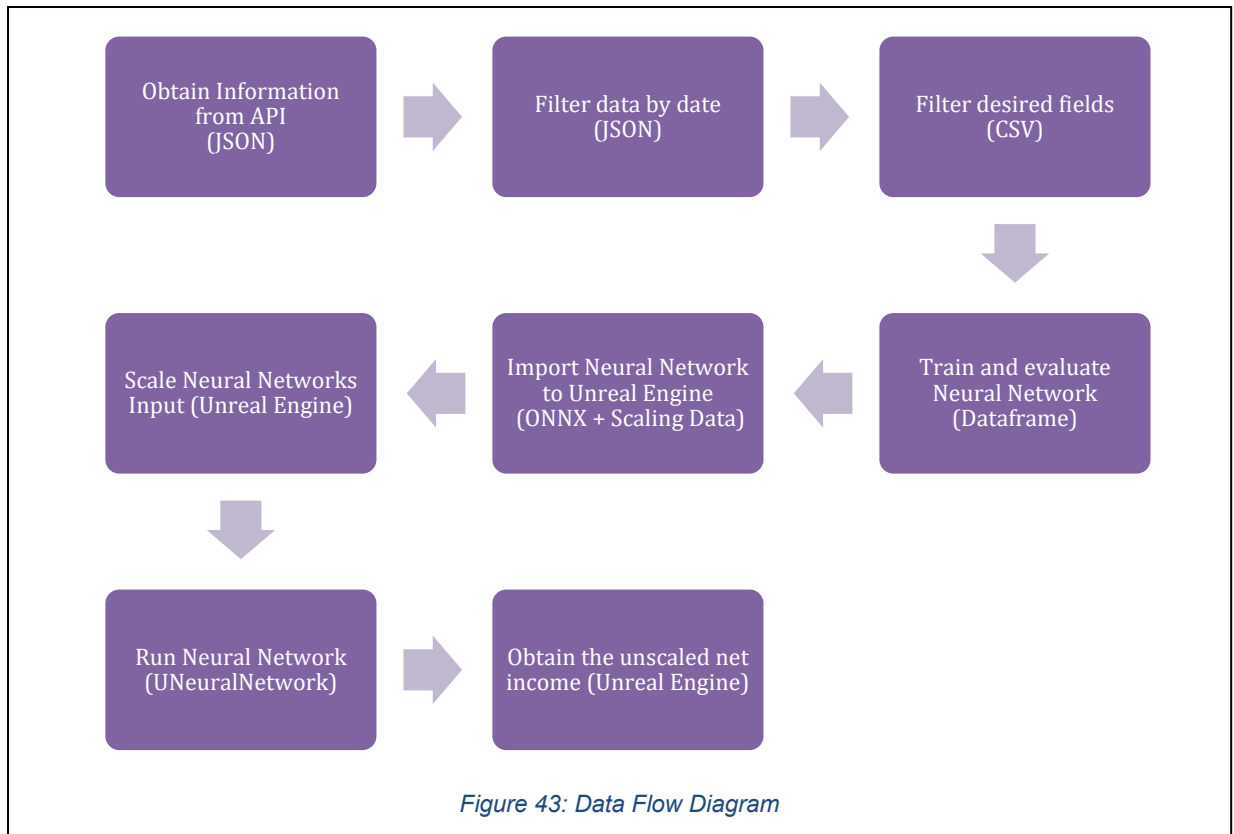
- Cash Flow Statement
 - Fiscal Date: The date of the data.
 - Operating Cash Flow: The cash the company generates during business operations.
 - Capital Expenditures: Non-recurring expenses commonly used to buy or upgrade large assets.
- Income Statement
 - Gross Profit: After deducting production and selling costs, the profits of a company.
 - Total Revenue: The amount of money generated by the business.
 - Cost of Revenue: The cost to manufacture and deliver a product or service.
 - Operating Expenses: Recurring expenses the company incurs through its operations.
 - Net Income: The company's overall profits after deducting all costs and expenses.
- Stock Price
 - Stock Price: The price of the company's stock in the stock market.

This meant that the neural network had the following inputs/outputs:



These fields provide enough information to predict the net income accurately and enough flexibility for gameplay adjustments. Such as increasing the stock price as marketing spending increases, linking the operating expenses to the amount the player spends monthly in the game, or making the capital expenditures match the initial, one-time cost of starting a new movie or series project.

6.4. Data flow



1. Obtain Information from API (JSON)

Obtain the financial data from the Alpha Vantage API from the following endpoints (Alpha Vantage, n.d.):

- [CASH FLOW](#)
- [INCOME STATEMENT](#)
- [TIME SERIES MONTHLY](#)

Each of these endpoints provides its own JSON file of the historical financial data.

2. Filter data by date (JSON)

Filter the data in the JSON files by date, ensuring they all cover the same time (ten years). Also, ensure that the data has the same number of time series entries; this includes filtering the TIME_SERIES_MONTHLY data quarterly to match the other data collected.

3. Filter desired fields (CSV)

Filter the desired fields of each JSON data file to only contain the desired columns. In this case, it would be:

- Cash Flow
 - Date
 - Operating Cash Flow
 - Capital Expenditures

- Income Statement
 - Gross Profit
 - Total Revenue
 - Cost of Revenue
 - Operating Expenses
 - Net Income
- Time Series Monthly
 - Opening Stock Value

These nine fields are saved on a quarterly (three months) basis. However, to get more data to train the neural network, the quarterly data is interpolated into monthly and weekly data. Though this interpolation may not be accurate to real-life monthly statistics, the quarterly data should still be the same.

This data is then exported to a single CSV file to be used to train the neural network.

4. Train and evaluate Neural Network (DataFrame)

Read the CSV financial data from a Python script, converting it into a data frame, and use it to build, train, and evaluate the simulation Neural Network. This will have the 'Net Income' field as the single output, with all the other fields as inputs. During the training stages the data is scaled using a MinMax scaler to better train the Neural Network.

5. Import Neural Network to Unreal Engine (ONNX + Scaling Data)

Export the built Neural Network as an ONNX file, an intermediary format between training frameworks and runtime applications (Breviu, 2022), and import it into Unreal Engine. Inside Unreal Engine, this will be shown as a Neural Network asset. This will also export the scaling data, the minimum and maximum value of each variable in the whole dataset, as a JSON.

6. Scale Neural Networks Input (Unreal Engine)

Using the scaling data JSON, do a MinMax normalisation of the game data (which is unnormalised to be seen by the player) to convert the data into a zero to one range.

7. Run Neural Network (UNeuralNetwork)

In Unreal Engine, run the Neural Network model by referencing its UNeuralNetwork class (a reference to the Neural Network asset) via code, passing the normalised data as parameters.

8. Obtain the unscaled net income (Unreal Engine)

For the player to properly visualise the data in a value range that makes sense, obtain the net income predicted by the neural network, which is in a zero to one range, and unscale it. This is once again done using the scaling data JSON. This unscaled value is the net income value shown to the player through the game UI.

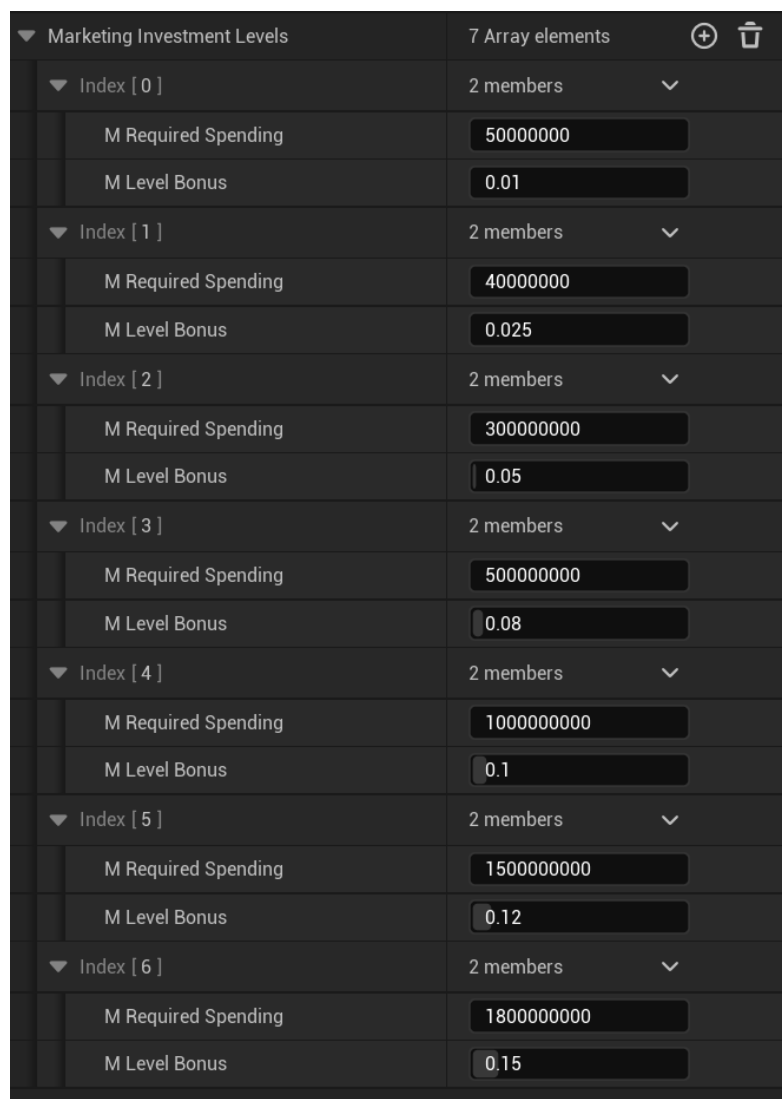
6.5. Neural Network – Gameplay Integration

Neural networks are a blackbox machine learning technique. This means they are not transparent about how they work due to the complexity of their inner workings.

This means that from a design perspective, there is no direct way to control the outcome of the neural network.

Instead, to balance the game, either the inputs of the neural network had to be tweaked based on the correlation they have with the net income output, or the net income output must be altered after this has been predicted by the neural network. For Unprofitable Startup, both of these approaches were employed.

6.5.1. Expenditures



Index	M Required Spending	M Level Bonus
Index [0]	50000000	0.01
Index [1]	40000000	0.025
Index [2]	300000000	0.05
Index [3]	500000000	0.08
Index [4]	1000000000	0.1
Index [5]	1500000000	0.12
Index [6]	1800000000	0.15

Figure 44: Marketing Investment Levels

The expenditures are handled as investment levels, were the more you have spent historically in a field, the bigger the bonus that field provides.

As the most requested features on the elicitation requirements survey and as a field used to calculate the profitability in other research papers (Gaikwad, et al., 2023), the main game parameters to alter the neural network were Marketing spending, and Research and Development spending.

However, these fields do not match directly to any of the inputs of the neural networks. So, they had to be linked to the inputs the neural network does possess or to the net income output itself.

In the case of Marketing, this was linked to the stock price. From the gameplay perspective that, the image of the company has a relation to the perceived value that investors give a company.

While Research and Development could have been reflected by reducing some of the company costs, it is unknown if the neural network correlates that lower costs to a higher net income. So, instead, investments in Research and Development provide a direct bonus to the net income after this is calculated by the neural network.

Together, spending in these two areas make up for the company's monthly expenses in the game, which do relate directly to the Operating Expenses input of the neural network.

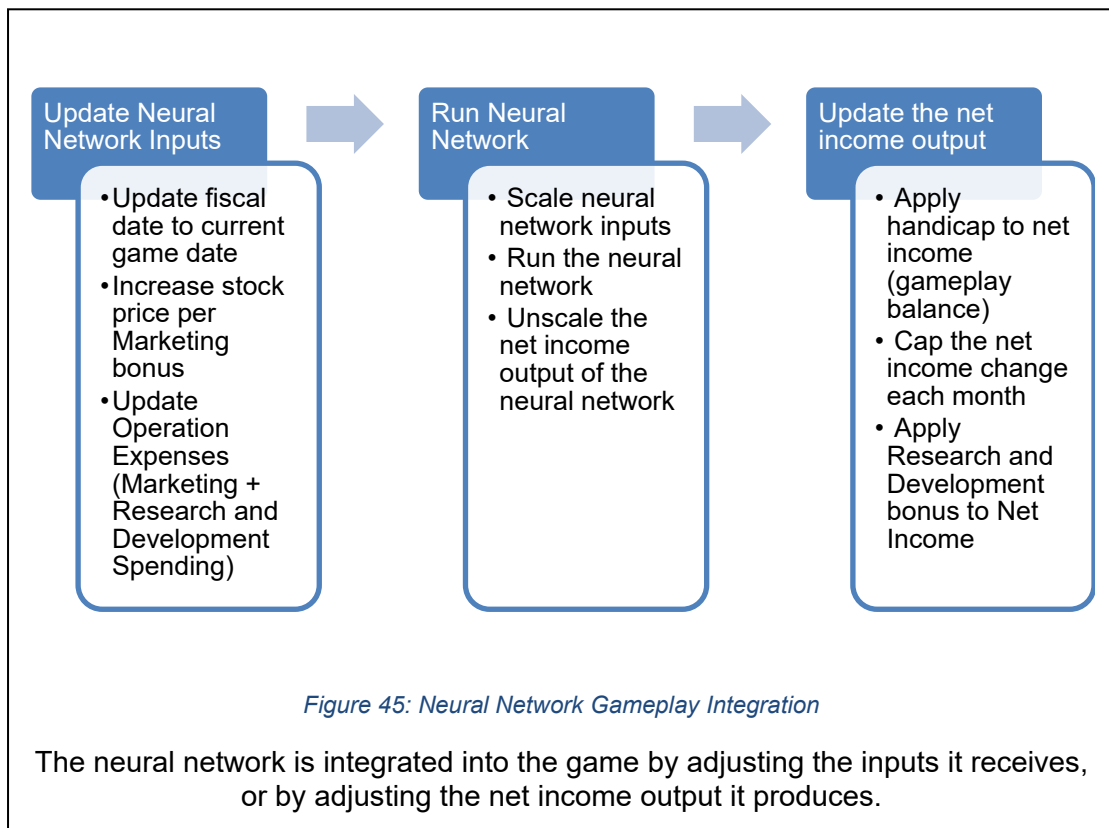
Further development of projects or random events could represent the capital expenditure field of the neural network or be used to affect such fields (e.g., a random event that increases the cost to produce something).

6.5.2. Net income monthly change cap

Even if the neural network shows good performance in predicting the net income, since it is a blackbox algorithm, there is no guarantee of the output it may produce.

So, to ensure that there is always some degree of fairness in the game, a cap on the amount of net income that can change each turn is needed. This cap ensures that, either for positive or negative changes, the net income never changes more than a predetermined amount each month.

In this way, there is no risk that the neural network will predict an outlier that causes a massive spike in the net income that could potentially make the player lose or win the game in a single turn. Instead, the neural network will mark the trend the profits are going into, providing the opportunity for the player to adjust its playing strategy.



6.6. Usability

6.6.1. User usability

Any type of strategy or simulation game can be daunting to new users, trying to introduce them to complex systems that can be confusing. Because of this, many project requirements were linked to making the game as easy to understand as possible.

Some of the features to be implemented to account for this issue are:

- Rounding of values: A particular issue of simulating a vast company's financial data is the massive value of variables for handling the financial data, with them being usually in the hundreds of millions. The Neural Network used for the simulation input and output is precise with float values and decimal points. However, from a user perspective, when dealing with millions, seeing cents, or even hundreds, of dollars, is not only useless, but it can also make the reading of the data confusing. This is why all the UI that shows the data should be rounded to the nearest thousands, million, or even billion, depending on the amount being shown. This rounding is represented by a suffix, like “k” for thousands or “m” for millions.
- Snapping slider values: When using a mouse with Unreal Engine UI sliders, there is no set “step size”, so if your slider represents a range from zero to one million, each individual number in this range can be set via the slider. Referring to the previous point, such a minute detail is useless for a user. To fix this the slider should “snap” to determined “step size” amounts, depending on the amount being shown in the slider. So, the user can increment the quantities of the sliders for each hundred thousand rather than for each individual value.
- Help Screen: Despite the relatively small scope of the gameplay component of the project, I consider it important to have both a screen at the start of the game explaining the basics of how to play the game, as well as a separate help menu explaining more in-depth instructions, as well as the meaning of some of the financial terminology used in the game.
- Tooltips: Unreal Engine functionality for tooltips should be used on multiple of the more complex UI elements in the game to explain things like what a financial term means or the functionality of the UI.



6.6.2. Developer usability

Even if developing this game as a programmer, the game code should be designed in such a way that non-programmers can adjust UI and design parameters, as it is the standard for any major project.

This means ensuring that all these UI or design variables are accessible for adjustments via the Unreal Engine editor and making different functions and code accessible via the visual scripting Blueprint system.

This ensures that when designing the game UI or tweaking the balance of the game, things that would typically not be handled by a programmer, this can be done purely via the engine GUI or visual scripting system, without having to modify any code nor must recompile the code when any changes are made.

This will, in turn, help enforce proper code abstractions in the code and simplify the iterative process of balancing the game or designing its UI.

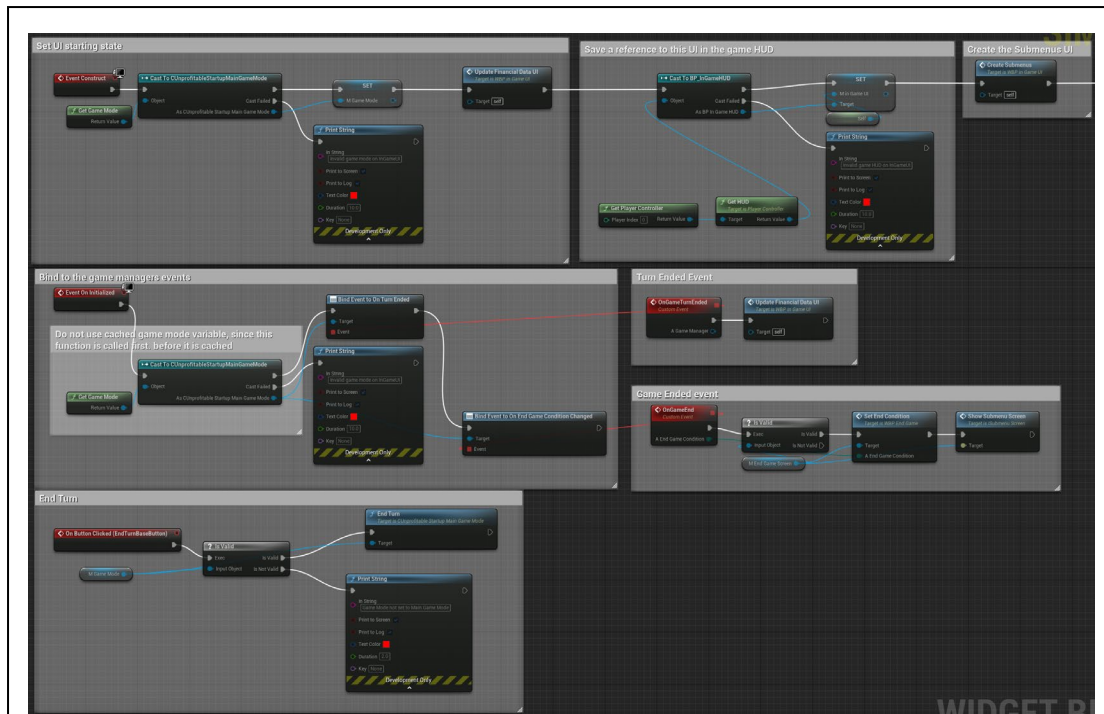


Figure 47: Developer Usability

Even if the implementation of the code is in C++, this interfaces with Unreal Engine visual scripting system and the game engine GUI to allow adjusting the game UI and game design balance. In the image above it can be seen that multiple C++ events and properties are hooked up with the game UI via the Blueprints system.

7. Tools and implementation

7.1 Tools

When choosing what tools to use for the project, there were two main criteria to consider:

- The chosen tool can be used effectively for the project and produce good results.
- The chosen tool can help me enhance my skills.

So, the tools chosen for this project presented a challenge to me to help me enhance my skills, but without compromising the quality of the project's end product.

From a personal point of view, I wished to take advantage of this project to develop my skills in:

- Machine Learning: Though it has existed in some form or another for years, the surge of generative AI in recent years has increased the importance of this skills (Jones, 2024).
- Python: Python is one of the most popular programming languages in the world (Daigle & Staff, 2023), and I considered my lack of experience in any project with it to be a significant gap in my skills.
- Unreal Engine with C++: Though I had previous experience developing a university project using C++ with the Unreal game engine, I did not consider myself experienced enough in this field due to the limited scope of my previous project.

I felt that developing these skills and learning these tools would help boost my knowledge and toolset for any future employment.

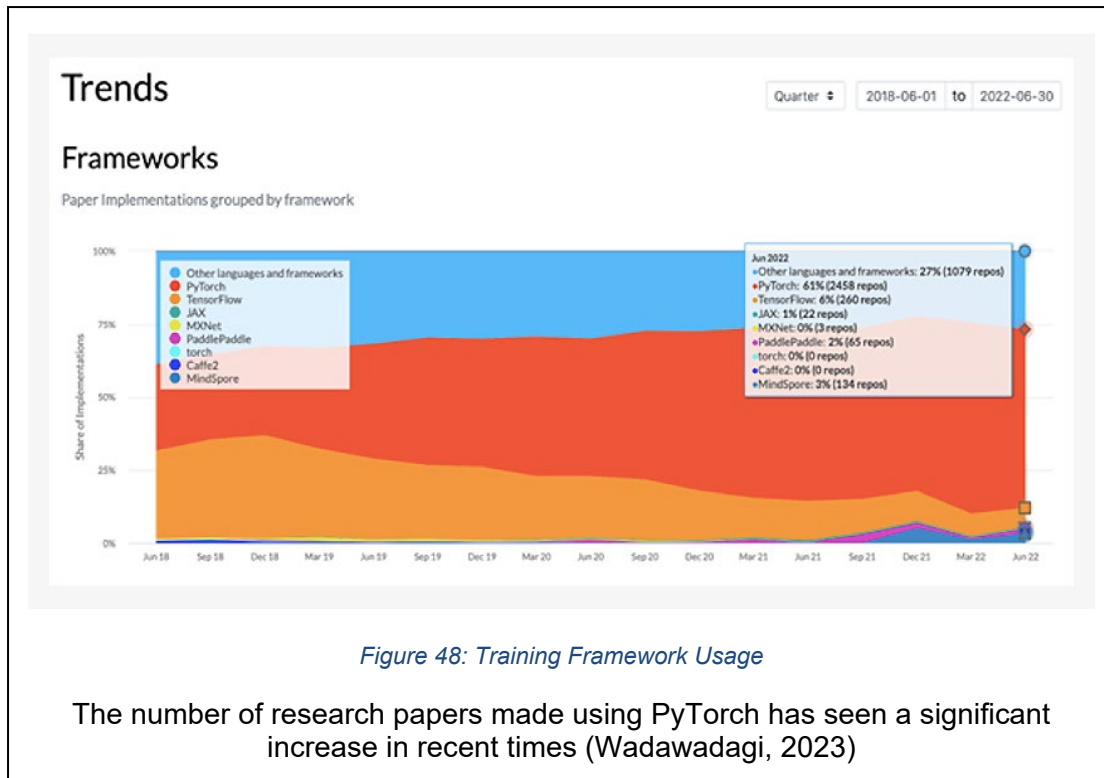
7.1.1. Neural network training

7.1.1.1. Training framework

The main requirement for the training framework was for it to be able to train the neural network model and export it as an ONNX file so that it can be imported into the game engine.

Keras, Tensorflow, or PyTorch could have easily fulfilled this requirement. However, I made the personal choice of using PyTorch for its more “Pythonic” approach since I wanted to develop my skills with Python, a programming language I had no previous experience with.

This choice was again reinforced by data that highlight that PyTorch is gaining a lot of popularity, and experience with it could, therefore, prove a valuable skill to have.



Since PyTorch is used in Python, a series of other Python libraries were used, the most notable being:

- Scikit-Learn: Used to scale the neural network data, split the training and evaluation datasets, and evaluate the neural network inference performance.
- Matplotlib: When evaluating the neural network, visualise the data predicted by the neural network with the true values from the data.
- Netron: Visualise the neural network architecture.

See [Appendix 8 – Python required packages](#) for detailed versions of all Python libraries used.

7.1.1.2. Financial data API

Although there may be datasets containing financial information, using an API to get the financial data would allow for a better setup of the neural network training process workflow. Since this would only contain the specific information requested through the API.

Despite the yFinance API being the most well-known of the researched APIs, the use of the Alpha Vantage API was for three main reasons:

- The historical data of yFinance was insufficient to train the neural network. yFinance only provided the last five quarterly statements of a company's balance sheet, which would give too few data entries to train the neural network. In contrast, Alpha Vantage provided access to roughly 15 years' worth of quarterly data.
- Lacking endpoint support in yFinance. From the prototyping stage of the project, it was seen that many of the more detailed endpoints concerning

earnings data for yFinance were not supported. In contrast, Alpha Vantage has a broader range of endpoints to get all the different data.

- Unreliability of web scraping. yFinance obtains its data via scraping the Yahoo Finance website. Though this is done using Yahoo's publicly available APIs, yFinance is not officially affiliated with it. So, a change in the Yahoo website, or even a local browser setting, could affect the gathering of financial data. In contrast, Alpha Vantage requests are made directly to their own API, reducing the risk of this process breaking down at some point.

The screenshot displays the Alpha Vantage Endpoints Documentation. On the left is a dark sidebar with a 'Table of Contents' listing various API categories like 'Core Stock APIs', 'Alpha Intelligence™', and 'Fundamental Data'. The main content area shows the documentation for the 'BALANCE_SHEET' endpoint, which is marked as 'Trending'. It includes a Python code snippet for making a request, a description of the endpoint's purpose, and 'API Parameters' such as 'Function', 'Symbol', and 'apikey'. Below this, there is an 'Example - annual & quarterly balance sheets for IBM' with a link to JSON output. A 'Language-specific guides' section offers options for Python, NodeJS, PHP, C#/NET, and Other. The 'CASH_FLOW' endpoint is also visible below, also marked as 'Trending'.

Figure 49: Alpha Vantage Endpoints Documentation

Alpha Vantage provides detailed documentation on its different endpoints with

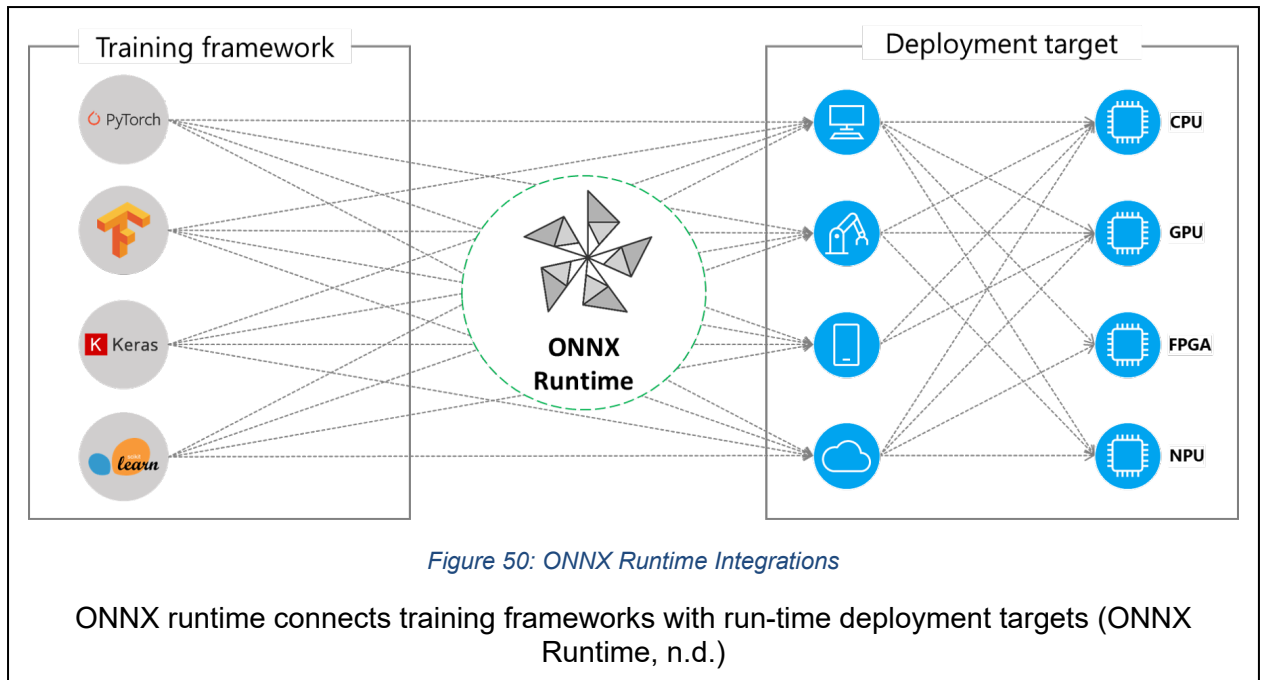
7.1.1.3. ONNX

When first researching this project's viability, one of the first tasks was finding how to run a neural network model on a run-time environment, such as a game engine.

Though some deep learning frameworks provide a C++ interface that could potentially be used to train and run the neural network, this appeared to overcomplicate the issue. Since at runtime, there was no need to train the neural network, at runtime the trained neural network just had to be run to make inferences.

From the research, the solution was found in the Open Neural Network Exchange (ONNX), a framework to export neural networks to a training framework independent model format, and that provides integration with different run-time environments (ONNX Runtime, n.d.), including the game engines of Unity and Unreal Engine (Breviu, 2022).

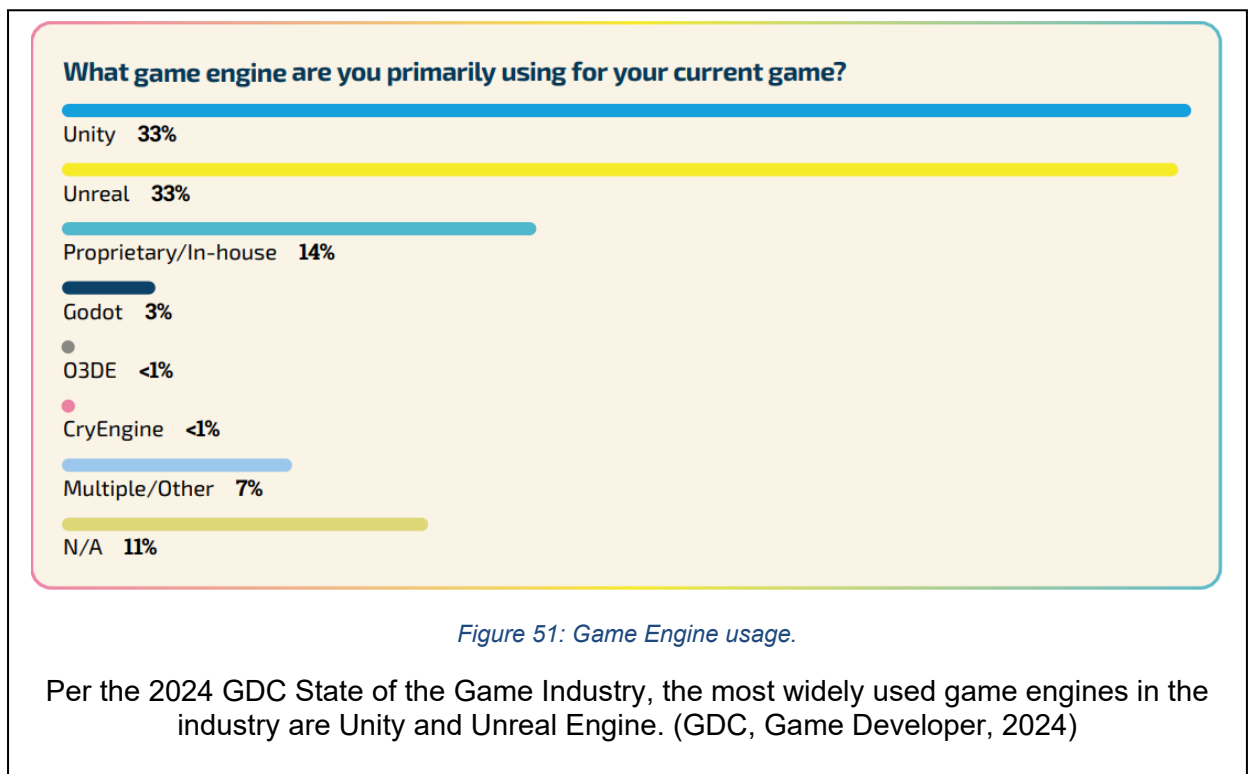
This allows the export of the neural network model from the training framework as an ONNX model, import into the game engine, and making inferences with it at runtime, serving as the connection between the training and runtime stages.



7.1.2. Unreal Engine

7.1.2.1. Game engine choice

The most popular game engines, Unreal Engine and Unity (GDC, Game Developer, 2024), are coincidentally the only ones with official support for Neural Network inference through their different plugins. So, both would be viable options for developing this project, but I took the personal choice of using Unreal Engine.



This is because I wanted to develop my personal skills in both Unreal Engine and, more specifically, its C++ coding implementation, which are highlighted as core employability skills (Into Games, 2024). Besides that, I felt that my previous experience and projects had been too focused on Unity and C#.

Changes in the Unity game engine pricing model (Batchelor, 2023) have also caused multiple developers to consider switching to abandon Unity for a different game engine (GDC, Game Developer, 2024).

7.1.2.2. Features employed

Though most of the project functionality will be implemented directly via C++, a lot of the UI will be handled via Unreal Engine `Blueprint` visual scripting system. This includes creating C++ functions and classes that can interact and be called from this visual scripting system.

This integration of both C++ and visual scripting allows design and art decisions to be made directly in the editor without having to recompile the code, improving workflow efficiency.

7.1.2.3. Game engine version

The project will be made using the version of Unreal Engine 5.0.3. Despite newer versions of the game engine being available, with version 5.0.3 being released in July 2022 (Schade, 2022), this was chosen to keep compatibility with the version installed in the University of Westminster computer labs. This was also done since it would be unfeasible to keep updating the project, trying always to use the latest version.

A constraint of this Unreal Engine version is that the Neural Network Inference plugin (Unreal Engine, 2022) released for it as an experimental feature, has already been superseded by the Neural Network Engine plugin, starting in Unreal Engine version 5.2 (Epic Games, 2023), also in an experimental state. Though there may be more prominent differences in their internal workings, feature-wise, both plugins support the core functionality of importing and running a Neural Network model through ONNX runtime.

7.2. Implementation

The implementation of this project can be broken down into two main components:

- The training of the neural network, via a Python script using PyTorch.
- The integration of the neural network into the simulation gameplay, which was done via Unreal Engine 5, primarily through C++.

7.2.1. Creating net income neural network

7.2.1.1. Obtaining training dataset

Since the start of the prototype stage, one of the major issues of this project was obtaining financial information to train a neural network. As a requirement to investors and commerce authorities, publicly traded companies must publish financial statements quarterly or annually (Hayes, 2021). So, the companies' data was available, but it required finding an appropriate method to get the historical data of the desired company.

From the prototyping stage, through the PSDP, it was found that the Alpha Vantage API could be used to obtain the different financial statements of a company throughout the years. These data could be obtained as a JSON and then read through a Python script.

Based on previous literature survey (Gaikwad, et al., 2023), and a combination of results from user surveys, different variables of the financial statements were extracted. The fields chosen to train the neural network not only had to be important for the training of the network, but in most cases, they also had to be able to contribute to the gameplay. As per the norm of a supervised learning method like neural networks, the actual net income of the company was also extracted from these financial statements.

Because of this, the job of my first Python script is:

1. Read the JSON data obtained via Alpha Vantage of the Cash Flow, Income Statements, and Stock value.
2. Filter only the desired features of this data since these statements contain a lot of additional information that is not needed.
3. Join the different data statements into a single data frame.
4. Since the statements have quarterly data from approximately the last ten years (data starting from 31-03-2013 until 31-12-2023), they only had 44 data entries. So, such as a small dataset would not be able to produce good results for training the neural network. Therefore, the quarterly data was interpolated into monthly data (130 data entries) or weekly data (562 data entries) to have more data to train the neural network.

```

#-----#
# Interpolation function
#-----#
# To have more data to train the neural network, interpolate the quarterly data
# into monthly or weekly data. So the values between quarters will be interpolated
# in the months/weeks between those quarters
def interpolate_financial_data(a_source_financial_data, a_date_column_name, a_target_frequency):
    # To be able to interpolate the data,
    # ensure all fields except the date are numerics
    # Separate the date from the other data being converted to numeric
    date_financial_data = a_source_financial_data[a_date_column_name]
    numeric_financial_data = a_source_financial_data.drop(columns=[a_date_column_name])

    # Convert data to numeric
    numeric_financial_data = numeric_financial_data.apply(pandas.to_numeric, errors='coerce')

    # Rejoin the date with the now numeric data
    interpolated_financial_data = pandas.concat([date_financial_data, numeric_financial_data], axis=1)
    interpolated_financial_data.columns = a_source_financial_data.columns # Preserve the column names

    # Set the date column as the index
    # so that it is used to determine the interpolation range
    interpolated_financial_data[a_date_column_name] = pandas.to_datetime(interpolated_financial_data[a_date_column_name])
    interpolated_financial_data.set_index(a_date_column_name, inplace=True)

    if a_target_frequency == 'monthly':
        # Interpolate the quarterly data to months
        interpolated_financial_data = interpolated_financial_data.resample('M').interpolate()
    elif a_target_frequency == 'weekly':
        # Just interpolating the data to weekly directly does not produce
        # good results (though it can be improved when source of interpolation is monthly instead
        # of quarterly)
        # So to get more accurate data first interpolate the data to daily, then to weekly
        daily_data = interpolated_financial_data.resample('D').interpolate()
        interpolated_financial_data = daily_data.resample('W').interpolate()
    else:
        raise ValueError("Target frequency should be either 'monthly' or 'weekly'")

    # Round data to 2 decimals for readability
    interpolated_financial_data = interpolated_financial_data.round(2)

    # Reset the index to make 'fiscalDateEnding' a column again
    interpolated_financial_data.reset_index(inplace=True)

    # Ensure the data is sorted from oldest to newest
    interpolated_financial_data[a_date_column_name] = pandas.to_datetime(interpolated_financial_data[a_date_column_name])
    interpolated_financial_data = interpolated_financial_data.sort_values(by=a_date_column_name, ascending=True)

    return interpolated_financial_data

```

Figure 52: Interpolate Financial Data Function

Interpolating the quarterly data into monthly or weekly data increased the available data to train the neural network.

5. After the data has been interpolated, compare it against the original data to guarantee the interpolation causes no significant distortion. This was done by checking that the interpolated weekly and monthly data have similar minimum and maximum values to the original data and by graphing a variable of the data through all three different time intervals.

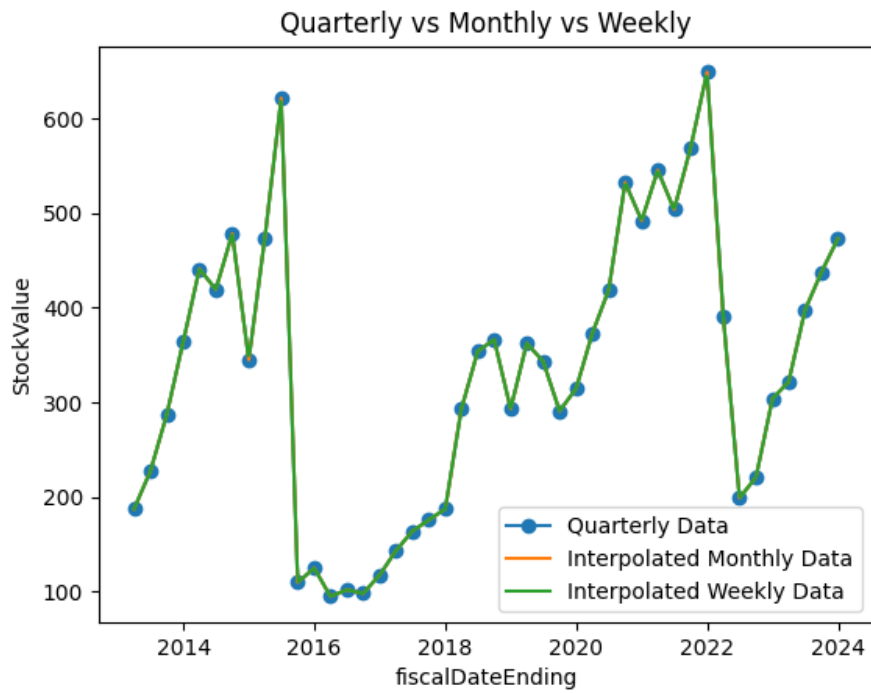


Figure 53: Interpolated Data Stock Value

As it can be seen, in the graph for the stock value, the interpolated data monthly pretty much matches the quarterly data precisely.

- Export the quarterly, monthly, and weekly data as separate CSV files. This is to allow testing of the neural network performance with datasets at different time intervals.

7.2.1.2. Creating the neural network

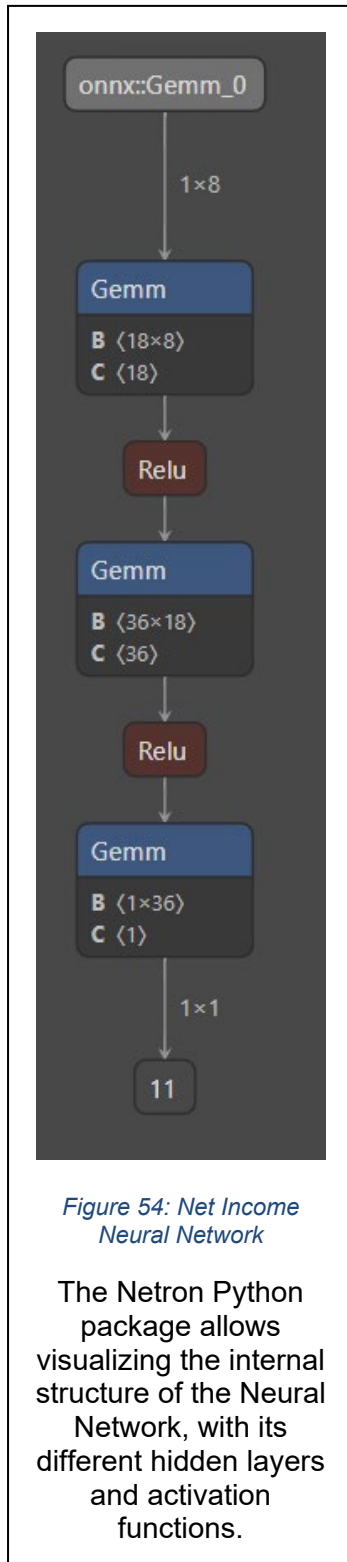


Figure 54: Net Income Neural Network

The Netron Python package allows visualizing the internal structure of the Neural Network, with its different hidden layers and activation functions.

To create the neural network, after reading the data from the CSV with the time interval of choice, the data first had to be pre-processed.

The financial data value range for each different variable can widely differ. This is most notable with the stock value, which is in the hundreds, and the expenses, which can be in the hundreds of millions. To avoid any of these values being overrepresented when training the neural network, they were normalised to a zero to one range using a min-max scaling.

After this, the data is separated into a training data set, used to train the neural network, and a testing data set, which is 20% of the overall data set. The testing data set is not introduced to the neural network during training and is purely used to evaluate the neural network performance after training.

The neural network to predict the net income is then created. This consists of 8 incomes and 1 output (the net income). This consists of 2 hidden layers, with rectified linear activation (ReLU) functions between the layers.

The layers are set as follows:

- First Hidden Layer: Receives the 8 original inputs from the financial data and produces 18 outputs.
- Second Hidden Layer: Receives the 18 outputs, and due to weights and backpropagation, produces 36 outputs.

The Rectified Linear Activation function was chosen since this seems to be the standard activation function for most neural networks, being simple to implement and relatively effective (Brownlee, 2021).

This type of architecture of neural network can be classified as a feedforward neural network (DeepAi, n.d.), with the information moving in a single direction and the training relying on backpropagation to adjust the neurons' weights.

```

209 #-----#
210 # Build the Neural Network
211 #-----#
212 # Class that inherits from nn.Module
213 class NetIncomeNN(nn.Module):
214
215     # Intialize layers in init(constructor) function
216     def __init__(self, input_size, output_size):
217         super(NetIncomeNN, self).__init__() # Call parent class nn.Module init
218         # Create hidden layers
219         hidden_layer1_output = 18
220         hidden_layer2_output = 36
221         self.hidden_layer1 = nn.Linear(input_size, hidden_layer1_output) # Set first linear layer
222         self.hidden_layer2 = nn.Linear(hidden_layer1_output, hidden_layer2_output) # Set second linear layer
223         self.hidden_layer3 = nn.Linear(hidden_layer2_output, output_size) # Set third linear layer
224
225         # Activation functions
226         self.activation_function_1to2 = nn.ReLU()
227         self.activation_function_2to3 = nn.ReLU()
228
229     # forward is the defined function to process input when the neural network is run
230     def forward(self, x):
231         x = self.hidden_layer1(x)
232         x = self.activation_function_1to2(x)
233         x = self.hidden_layer2(x)
234         x = self.activation_function_2to3(x)
235         x = self.hidden_layer3(x)
236         return x
237
238     # Get the number of inputs and outputs
239     input_size = scaled_input.shape[1] # 8 input features
240     output_size = 1 if len(scaled_output.shape) == 1 else scaled_output.shape[1] # 1 output feature
241
242     # Create the instance of the neural network
243     netIncome_neural_net = NetIncomeNN(input_size, output_size)

```

Figure 55: Code to create a PyTorch Neural Network

The neural network is created with 2 hidden layers, and the output layer. This is created using the PyTorch packages.

For training the neural network, a Mean Squared Error (MSE) function was set up as the loss function, and it was used to measure the actual output with the predicted output. This is because the data aims to solve a regression problem and has no notable data outlier (Yathish, 2022).

The neural network was then trained through multiple epochs, where for each epoch, the neural network predicted the net income and then, using the loss function and optimiser, it adjusted the weights of the neuron's connections.

The optimiser function used for the training was Adaptive Moment Estimation (Adam). This was chosen since it seems to have a good balance between the speed of the training and the ability to adapt to the epoch changes (Chugh, 2022). For training, this optimiser also provided control over the learning rate, which controls how much the model can change through each epoch pass.

From multiple tests, it was observed that the ideal number of epochs seemed to be around 200 – 300, at which point the neural network seemed to converge at the lowest loss value.

```

250 #-----#
251 # Train the Neural Network
252 #-----#
253 print(f"-----Training-----")
254
255 # Define loss function, used to measure the target with the predicted output,
256 # practically a function to determine how accurate the prediction will be
257 # Since data has no noticeable outliers, and is a regression problem, use Mean Squared Error (MSE)
258 loss_function = nn.MSELoss()
259
260 # The optimizer is the algorithm used to adjust the neural network weights
261 # Adam tends to be a generally good option for this (based on a gradient descent algorithm)
262 optimizer = optim.Adam(netIncome_neural_net.parameters(), lr=0.015)
263
264 num_epochs = 250 # How many times the entire dataset will be passed through the model
265 batch_size = 25 # The amount of samples that will be passed to the model at once
266 epoch_loss_print_frequency = 5 # How often to print the loss
267
268 # Use a dataloader, to account for the data not being divisible by
269 # the batch size (final batch may be smaller)
270 training_dataset = TensorDataset(training_input_tensor, training_output_tensor)
271 training_dataloader = DataLoader(training_dataset, batch_size=batch_size, shuffle=False, drop_last = False)
272
273 # Train data a certain number of times (epochs) at a certain
274 # number of samples (batch size) at a time
275 for epoch in range(num_epochs):
276     # Obtain batches from dataloader to account for possible "leftovers" from batch
277     for input_batch, output_batch in training_dataloader:
278         # Reshape the target batch to match the shape of the
279         # predicted tensor of torch.Size([10, 1])
280         output_batch = output_batch.view(-1, 1)
281
282         # Predict the result using the neural network model (forward pass)
283         predicted_output = netIncome_neural_net(input_batch)
284
285         # Measure how accurate the prediction was
286         loss = loss_function(predicted_output, output_batch)
287
288         # Reset the optimizer gradients
289         optimizer.zero_grad()
290
291         # backpropagation, compute the optimizer gradients per data model
292         loss.backward()
293
294         # Update optimizer weights for next pass
295         optimizer.step()
296
297         # Print loss (how accurate prediction is) every 5 epochs
298         if epoch % epoch_loss_print_frequency == 0 or epoch == num_epochs - 1:
299             print(f"Epoch {epoch}, has loss of: {loss}")

```

Figure 56: Code to train the Neural Network

The neural network is trained, using only the training data set, through multiple epochs. The optimizer function relies on backpropagation to adjust the weights of the model.

7.2.1.3 Evaluating the neural network.

To measure the neural network performance, the neural network predictions were evaluated against the testing dataset, which was excluded from the training. The same input variables were used to make a prediction of the net income, and this was compared against the actual net income these variables generated.

Some of the different metrics used to compare the results of these predictions were (Brownlee, 2020) (Bobbitt., 2021) (Vandeput, 2019):

- Mean Forecast Error (MFE): Also known as forecast bias, this measures how much the neural network over forecast (negative value) or under forecasts (positive value). The closer to 0 the result, the more accurate the prediction is.

$$MFE = \frac{\sum(Expected\ Value - Predicted\ Value)}{n}$$

Equation 1: Mean Forecast Error (MFE)

- Mean Absolute Error (MAE): Measures the accuracy of the prediction where the error rate forecasted is absolute (positive). The smaller the error, the more accurate the prediction.

$$MAE = \frac{\sum |(Expected Value - Predicted Value)|}{n}$$

Equation 2: Mean Absolute Error (MAE)

- Mean Absolute Percentage Error (MAPE): This metric is the same as MAE but converted to a percentage to better visualise the error margin against different value ranges.

$$MAPE = \frac{\sum \frac{|(Expected Value - Predicted Value)|}{Expected Value}}{n} \times 100$$

Equation 3: Mean Absolute Percentage Error (MAPE)

- Mean Squared Error (MSE): This is the average of the squared forecast error. The lower the MSE, the more accurate the predictions. Since this uses the squared value of the errors, it is more sensitive to outliers since it increases their effects on the evaluation.

$$MSE = \frac{\sum (Expected Value - Predicted Value)^2}{n}$$

Equation 4: Mean Squared Error (MSE)

- Root Mean Squared Error (RMSE): This is the root squared MSE. This has the advantage that the previously squared units of the MSE are converted back to the same units used for the predictions.

$$RMSE = \sqrt{MSE}$$

Equation 5: Root Mean Squared Error (RMSE)

Based on these metrics, different parameters for the neural network were evaluated. This was done by iterating multiple structures for neural networks following the ones that gave the same metrics. The changes in the neural network included the number of neurons in each hidden layer, the number of epochs to train, the time interval of the data to train it (weekly, monthly, or quarterly) and the start date of the data used to train it.

ID	Data Start Date	Data Frequency	Data Entries	Outputs Hidden Layer 1	Outputs Hidden Layer 2	MFE	MAE	MAPE	MSE	RMSE
1	31/03/2013	Monthly	130	20	40	-103356890	135843700	40%	3.18E+16	178445599
2	31/03/2013	Weekly	562	20	40	-75005322	179293304	33%	5.72E+16	239080992
3	31/03/2013	Quarterly	44	20	40	-218839208	236019232	122%	9.25E+16	304188372
4	07/01/2018	Weekly	313	20	40	-266349457	266349457	57%	9.68E+16	311051674
5	31/01/2018	Monthly	72	20	40	-101750103	154683358	61%	3.90E+16	197585007
6	07/01/2018	Weekly	313	20	40	-225313447	239146856	62%	9.30E+16	304982858
7	07/01/2018	Weekly	313	20	40	-233930126	239378173	57%	8.64E+16	293961710
8	07/01/2018	Weekly	313	10	20	-179635368	208618480	56%	7.62E+16	276133039
9	07/01/2018	Weekly	313	5	10	-201776795	202799697	45%	5.74E+16	239510013
10	07/01/2018	Weekly	313	15	10	-248922126	248922126	53%	8.21E+16	286459928
11	07/01/2018	Weekly	313	5	0	-188644196	199102162	53%	6.71E+16	259100058
12	31/01/2018	Monthly	72	5	0	-202461036	206760076	74%	5.81E+16	241018661
13	31/03/2013	Monthly	130	5	0	226573206	256193940	26%	8.17E+16	285915476
14	31/03/2013	Monthly	130	25	0	-210596718	210596718	66%	7.13E+16	266949996
15	31/03/2013	Monthly	130	15	0	47555709	159231626	32%	3.38E+16	183950983
16	31/03/2013	Weekly	562	15	0	260486890	333062874	32%	1.35E+17	367698385
17	31/03/2013	Weekly	562	10	15	305253295	374153086	34%	1.71E+17	413351392
18	31/03/2013	Monthly	130	10	15	-86811696	169060649	53%	4.79E+16	218877598
19	31/03/2013	Monthly	130	25	15	-61808246	145126084	43%	3.51E+16	187337544
20	31/03/2013	Monthly	130	25	35	-57034809	138303249	41%	3.01E+16	173555771
21	31/01/2018	Monthly	72	25	35	-178855001	181930480	63%	4.21E+16	205260446
22	31/03/2013	Weekly	562	25	50	292117573	368534656	34%	1.64E+17	404866757
23	31/03/2013	Weekly	562	15	30	3192389	191786632	29%	5.67E+16	238087540
24	31/03/2013	Monthly	130	15	30	89834720	190268758	40%	4.45E+16	211046538
25	31/03/2013	Monthly	130	40	20	-67486458	135205590	40%	3.19E+16	178563507
26	04/01/2015	Weekly	470	20	40	117003651	230408309	37%	6.49E+16	254831778
27	31/03/2013	Weekly	105	20	30	-65597418	177036425	33%	6.44E+16	253725169
28	31/01/2015	Monthly	108	20	30	6957051	141329426	55%	3.11E+16	176382403
29	31/03/2013	Monthly	130	20	30	-78343370	164398290	58%	4.89E+16	221219402
30	31/03/2013	Monthly	130	18	36	-61094685	114673726	38%	2.70E+16	164419928
31	31/03/2013	Monthly	130	16	32	-137079534	177210185	53%	4.94E+16	222285548
32	31/03/2013	Monthly	130	18	32	-50344575	166849392	52%	4.53E+16	212756829
33	07/01/2018	Weekly	313	18	36	-146860257	158027668	46%	4.47E+16	211361418
34	04/01/2015	Weekly	470	18	36	-181156586	181600560	36%	5.39E+16	232253291
35	31/03/2013	Weekly	562	19	38	-25681227	221344433	35%	7.63E+16	276178004

Table 17: Neural Network Evaluation

From these tests, it was observed, as expected, that training the data with the quarterly information provided an abysmal performance, likely due to the low number of data entries for training.

It can also be seen that, generally, the best results were obtained from a 2 hidden layers neural network with between 15-20 neurons in the first layer and 30-40 neurons in the second layer.

It should also be noted that the start date of the financial information was never moved beyond 2020, even if the weekly interpolated data could provide sufficient data entries to train the neural network. This was purposely done to avoid skewing the neural network with financial data skewed by the effects of the COVID-19 pandemic.

Metric	Neural Network ID
Best MFE	23
Best MAE	30
Best MAPE	13
Best MSE	30
Best RMSE	30

Table 18: Best Neural Networks

Based on these metrics evaluations, it was observed that the neural network number 30, with monthly data and 2 hidden layers of 18 and 36 neurons, respectively, provided the best results in 3 out of the 5 metrics being evaluated, with an overall good performance.

As a final evaluation method for the neural network, the predicted net income values were plotted against the actual true net income values of the evaluation data set.

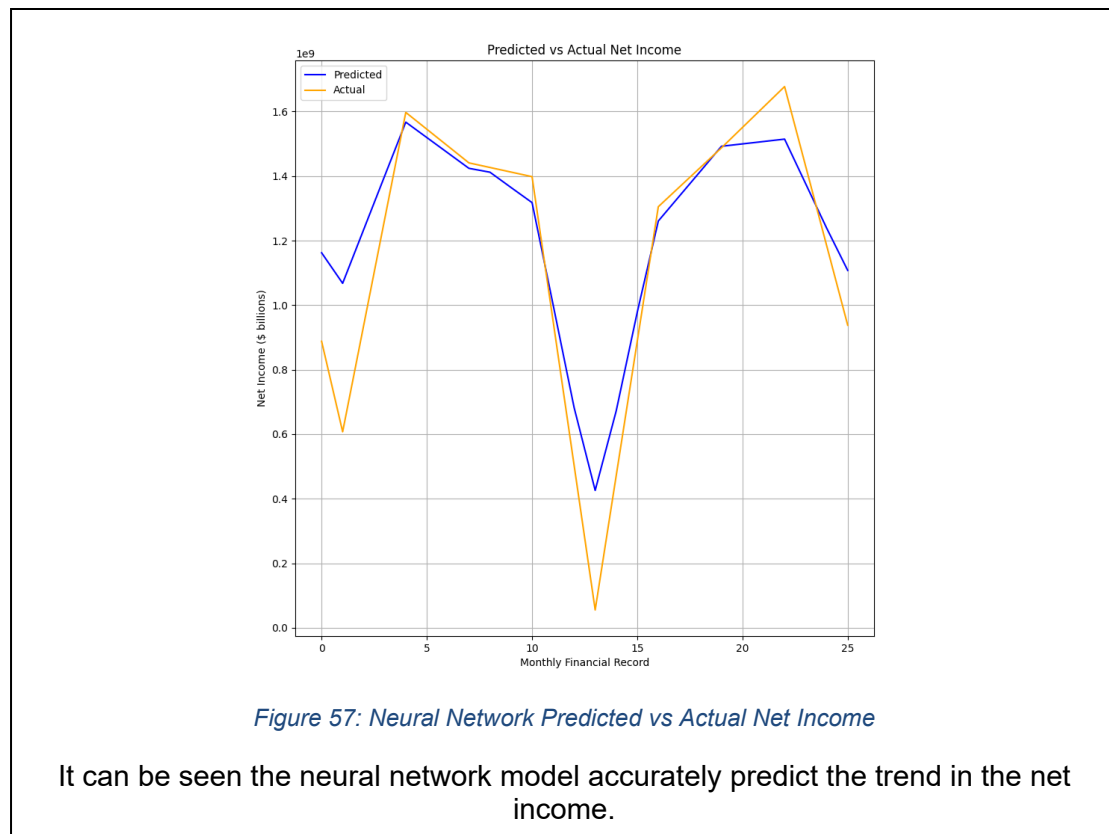


Figure 57: Neural Network Predicted vs Actual Net Income

It can be seen the neural network model accurately predict the trend in the net income.

7.2.1.4. Exporting the Neural Network

The final step is to export the neural network so that it can be used in the Unreal game engine. This consisted of two file exports:

- The ONNX neural network model. This represents the trained neural network, with its structure and weights.
- The scaling data. The neural network expects its input to be scaled to a zero-to-one range, and it also predicts its net income output in a zero-to-one range. As the name implies, this min-max scaling is done by scaling the value based on the variable's minimum and maximum values in the data. When training the neural network, this is easy to do, and not a performance concern. However, it cannot be expected that these values will be found at run time in the game engine since it would require all the datasets to be loaded and parsed in the game engine. So, export the same minimum and maximum values used to train the neural network so the data can be appropriately scaled/unscaled.

```
1  {
2  }
3  "fiscalDateEnding": {
4    "min": 1364688000.0,
5    "max": 1703980800.0
6  },
7  "operatingCashFlow": {
8    "min": -1461975000.0,
9    "max": 2178740000.0
10 },
11 "capitalExpenditures": {
12   "min": 8088000.0,
13   "max": 167327000.0
14 },
15 "grossProfit": {
16   "min": 287009000.0,
17   "max": 3610880000.0
18 },
19 "totalRevenue": {
20   "min": 1023961000.0,
21   "max": 8832825000.0
22 },
23 "costOfRevenue": {
24   "min": 736952000.0,
25   "max": 5404160000.0
26 },
27 "operatingExpenses": {
28   "min": 397979000.0,
29   "max": 3761231000.0
30 },
31 "StockValue": {
32   "min": 94.58,
33   "max": 649.48
34 },
35 "netIncome": {
36   "min": 2689000.0,
37   "max": 1706715000.0
38 }
```

Figure 58: scaling_data_information.json

The scaling data exported from the Python script allows scaling the input received by the neural network in the game, as well as unscaling its predicted net income.

7.2.2. Unprofitable Startup game

The gameplay component of the project was done via Unreal Engine 5.0, which integrated the Neural Network with a management simulation style gameplay.

The game's core functionality was done via “Unreal Engine C++”, an abstraction layer on top of C++ that provides functionality such as memory management. This C++ code not only allows integration with Unreal Engine functionality, but most importantly for this project, it also allows integration of the C++ code with the engine visual scripting “Blueprint” system.

7.2.1.1. Integrating the neural network

To use a neural network, the Neural Network Inference (NNI) plugin had to be added to the project first. (Frames, 2022). This allows importing neural networks that have been saved as an ONNX format into Unreal Engine, as well as using an inference engine to be able to set up the neural network and run it.

To simplify the handling of the neural network data, I created a struct, “FSFinancialData” to represent all the financial data of the company. This object can then be easily passed between the player’s company and the neural network whenever the company simulation is required.

```
10 USTRUCT(BlueprintType, meta = (ToolTip = "Representation of the financial data of a company"))
11
12 struct FSFinancialData
13 {
14     GENERATED_BODY()
15
16 public:
17     // Financial terms definitions from https://www.investopedia.com/
18
19     UPROPERTY(EditAnywhere, BlueprintReadWrite, meta = (ToolTip = "The date of the financial data"))
20     FDateTime m_date;
21     UPROPERTY(EditAnywhere, BlueprintReadWrite, meta = (ToolTip = "Measure of the amount of cash generated by a company's normal business operations. "))
22     float m_operatingCashFlow;
23     UPROPERTY(EditAnywhere, BlueprintReadWrite, meta = (ToolTip = "Funds used by a company to acquire, upgrade, and maintain physical assets such as property, technology, or equipment"))
24     float m_capitalExpenditure;
25     UPROPERTY(EditAnywhere, BlueprintReadWrite, meta = (ToolTip = "The profit a company makes after deducting the costs associated with producing and selling its products"))
26     float m_grossProfit;
27     UPROPERTY(EditAnywhere, BlueprintReadWrite, meta = (ToolTip = "The money generated from normal business operations"))
28     float m_totalRevenue;
29     UPROPERTY(EditAnywhere, BlueprintReadWrite, meta = (ToolTip = "The total cost of manufacturing and delivering a product or service to consumers"))
30     float m_costOfRevenue;
31     UPROPERTY(EditAnywhere, BlueprintReadWrite, meta = (ToolTip = "Expense that a business incurs through its normal business operations"))
32     float m_operatingExpenses;
33     UPROPERTY(EditAnywhere, BlueprintReadWrite, meta = (ToolTip = "Value of the shares of the company in the stock market"))
34     float m_stockValue;
35
36     UPROPERTY(EditAnywhere, BlueprintReadWrite, meta = (ToolTip = "Company's overall profitability after all expenses and costs have been deducted from total revenue"))
37     float m_netIncome;
38 };
```

Figure 59: Code FSFinancialData

The FSFinancialData struct is used to simplify the passing of the financial status of the company to the neural network.

To use the neural network, I created a wrapper class around it. This wrapper class allows running the neural network model, receiving the unscaled financial data as input. This wrapper class only takes care of setting up the neural network model, and running the neural network inference, internally scaling the neural network inputs and output.

```

12 UCLASS(BlueprintType,
13     meta = (ToolTip = "Neural Network wrapper to make inferences on the Net Income of a company"))
14 class UCFinancialNeuralNetwork : public UNeuralNetwork
15 {
16     GENERATED_BODY()
17
18 public:
19     UCFinancialNeuralNetwork();
20
21     /**
22     * Sets the Neural Network Model to be used
23     * @param aNeuralNetworkModel The Neural Network Model (obtained from ONNX file) to be set
24     * @param aNeuralNetworkModel The path of the scaling data for the neural network model (path relative to Content folder)
25     */
26     UFUNCTION(BlueprintCallable, Category = "Financial Neural Network")
27     void SetModel(UNeuralNetwork* aNeuralNetworkModel, const FString& aScalingDataFilePath);
28
29     /**
30     * Run the Neural Network model, and from the parameters infer the net income
31     * @param aFinancialInput The unscaled financial data that will be inputted to the Neural Network
32     * @return the unscaled net income inferred by the Neural Network
33     */
34     UFUNCTION(BlueprintCallable, Category = "Financial Neural Network")
35     float RunModel(FSFinancialData aFinancialInput);
36
37 private:
38
39     // UPROPERTY helps ensure that this pointer is not automatically garbage collected
40     // https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/ProgrammingWithCPP/UnrealArchitecture/Objects/Optimizations/
41     UPROPERTY(VisibleInstanceOnly)
42     UCFinancialDataScaler* m_dataScaler;
43
44     // Transient properties are always initialized to zero and are not serialized to disk
45     UPROPERTY(Transient)
46     UNeuralNetwork* m_simulationNN;
47
48 };

```

Figure 60: Code for header Neural Network Wrapper Class

The wrapper class for the neural network handled both the neural network, and the scaling of its parameters.

This financial data needs to be scaled to a zero to one range before being inputted into the neural network. The neural network then produces the net income as an output, which is also subsequently unscaled. This scaling is done using a Min-Max normalisation.

$$0 - 1 \text{ Scaled Value} = \frac{(\text{Unscaled Value} - \text{Minimum Value})}{(\text{Max Value} - \text{Minimum Value})}$$

Equation 6: Min-Max Normalization 0 to 1

The data scaling is handled by parsing each data variable's minimum and maximum values per the scaling data JSON (exported by the training framework) and doing the min-max scaling. As mentioned in the design, it is vital for usability that the values of the financial data make sense and are not in a relative zero to one scale. This is why the user always sees and handles the unscaled value, rather than the apparently non-sensical, for the player, zero to one range.

```

25  bool UCFinancialDataScaler::ParseMinMaxValues(const FString& aScalingDataFilePath)
26  {
27      if (aScalingDataFilePath.IsEmpty())
28      {
29          UE_LOG(LogTemp, Error, TEXT("No JSON file for scaling data has been specified"));
30          return false;
31      }
32
33      // Make the path relative to the project content directory
34      const FString projectFilePath = FPaths::ProjectContentDir() + aScalingDataFilePath;
35
36      // Load the JSON file as a string
37      FString stringJSON;
38      if (FFileHelper::LoadFileToString(stringJSON, *projectFilePath))
39      {
40          TSharedPtr<FJsonObject> JsonObject = MakeShareable(new FJsonObject());
41          TSharedPtr<TJsonReader<>> JSONReader = TJsonReaderFactory<>::Create(stringJSON);
42
43          // Deserialize the JSON into a JSON object
44          if (FJsonSerializer::Deserialize(JSONReader, JsonObject) && JsonObject.IsValid())
45          {
46              // Setup the minimum and maximum value of all the fields' MinMax Scalers
47              ParseMinMaxValueOfField("fiscalDateEnding",    JsonObject, *m_scalerFiscalDateEnding);
48              ParseMinMaxValueOfField("operatingCashFlow",  JsonObject, *m_scalerOperatingCashFlow);
49              ParseMinMaxValueOfField("capitalExpenditures", JsonObject, *m_scalerCapitalExpenditure);
50              ParseMinMaxValueOfField("grossProfit",        JsonObject, *m_scalerGrossProfit);
51              ParseMinMaxValueOfField("totalRevenue",       JsonObject, *m_scalerTotalRevenue);
52              ParseMinMaxValueOfField("costOfRevenue",      JsonObject, *m_scalerCostOfRevenue);
53              ParseMinMaxValueOfField("operatingExpenses",  JsonObject, *m_scalerOperatingExpenses);
54              ParseMinMaxValueOfField("netIncome",          JsonObject, *m_scalerNetIncome);
55              ParseMinMaxValueOfField("StockValue",         JsonObject, *m_scalerStockValue);
56
57              return true;
58          }
59          else
60          {
61              UE_LOG(LogTemp, Error, TEXT("Error deserializing JSON file %s"), *projectFilePath);
62              return false;
63          }
64      }
65      else
66      {
67          UE_LOG(LogTemp, Error, TEXT("Error reading JSON file %s"), *projectFilePath);
68          return false;
69      }
70
71      return false;
72  }

```

Figure 61: Code to parse minimum and maximum Values

The minimum and maximum value of each data field is parsed from a “Scaling Data” JSON file that is exported from the training framework.

To simplify the scaling of all the financial data in “FSFinancialData”, the neural network wrapper class batch all the scaling of these variables into a “UCFinancialDataScaler” object. This object handles setting up the minimum and maximum values for each field and grouping their scaling calls when required. So, all of the financial data can be scaled at once through a single function call.

```

74  ~FSFinancialData UCFinancialDataScaler::GetMinMaxScaledData(FSFinancialData aUnscaledData, float& aNanosecondscDate) const
75  {
76      // Convert the data to nanoseconds, since Neural Network cannot take pure dates
77      float unscaledNanoSeconds = static_cast<float>(aNanosecondscDate.ToUnixTimestamp());
78
79      FSFinancialData scaledData = aUnscaledData;
80
81      if (IsValid(m_scalerFiscalDateEnding) == false ||
82          IsValid(m_scalerOperatingCashFlow) == false ||
83          IsValid(m_scalerCapitalExpenditure) == false ||
84          IsValid(m_scalerGrossProfit) == false ||
85          IsValid(m_scalerTotalRevenue) == false ||
86          IsValid(m_scalerCostOfRevenue) == false ||
87          IsValid(m_scalerOperatingExpenses) == false ||
88          IsValid(m_scalerStockValue) == false ||
89          IsValid(m_scalerNetIncome) == false
90      )
91      {
92          UE_LOG(LogTemp, Error, TEXT("UCFinancialDataScaler has not been initialised, exiting function"));
93          aNanosecondscDate = unscaledNanoSeconds;
94          return scaledData;
95      }
96
97      // Convert the data to nanoseconds, the format expected by the Neural Network
98      // scale it by MinMax
99
100     // Scale all the data using the previously saved min and max values
101     aNanosecondscDate = m_scalerFiscalDateEnding->ScaleValue(unscaledNanoSeconds);
102     scaledData.m_operatingCashFlow = m_scalerOperatingCashFlow->ScaleValue(aUnscaledData.m_operatingCashFlow);
103     scaledData.m_capitalExpenditure = m_scalerCapitalExpenditure->ScaleValue(aUnscaledData.m_capitalExpenditure);
104     scaledData.m_grossProfit = m_scalerGrossProfit->ScaleValue(aUnscaledData.m_grossProfit);
105     scaledData.m_totalRevenue = m_scalerTotalRevenue->ScaleValue(aUnscaledData.m_totalRevenue);
106     scaledData.m_costOfRevenue = m_scalerCostOfRevenue->ScaleValue(aUnscaledData.m_costOfRevenue);
107     scaledData.m_operatingExpenses = m_scalerOperatingExpenses->ScaleValue(aUnscaledData.m_operatingExpenses);
108     scaledData.m_stockValue = m_scalerStockValue->ScaleValue(aUnscaledData.m_stockValue);
109     scaledData.m_netIncome = m_scalerNetIncome->ScaleValue(aUnscaledData.m_netIncome);
110
111     return scaledData;
112 }

```

Figure 62: Code to Scale Financial Data

After the minimum and maximum values have been set, once, all the financial data can be easily scaled through a single function call. This optimizes the process of updating the data for the neural network each month in the game.

The process of running the neural network itself is straightforward. After the neural network model has been loaded, to run the neural network the financial data inputs just must be scaled and added as float inputs to the neural network (the order of the inputs must match the same as the neural network design). After the neural network model is run, the output can subsequently be extracted and unscaled.

It should be noted that the wrapper for the neural network has nothing to do with any gameplay element and is purely used to run the neural network. Any adjustments to the game’s balance or design are made by adjusting the inputs the neural network receives or altering the net income after it has been calculated.


```

27  ~float UCFinancialNeuralNetwork::RunModel(FSFinancialData aFinancialInput)
28  {
29      // Check Neural Network has been set
30      if (m_simulationNN == nullptr || m_simulationNN->IsLoaded() == false)
31      {
32          UE_LOG(LogTemp, Error, TEXT("Cannot run Neural Network Model, it has not been set to valid value"));
33          return 0.0f;
34      }
35
36      // Scale the inputs to a range accepted by the Neural Network
37      float timeNanoSeconds = 0.0f;
38      FSFinancialData scaledData = m_dataScaler->GetMinMaxScaledData(aFinancialInput, timeNanoSeconds);
39
40      TArray<float> inputNN;
41
42      // Add the variables to the array, in the order in which they were set
43      // in the Neural Network model
44      inputNN.Emplace(timeNanoSeconds);
45      inputNN.Emplace(scaledData.m_operatingCashFlow);
46      inputNN.Emplace(scaledData.m_capitalExpenditure);
47      inputNN.Emplace(scaledData.m_grossProfit);
48      inputNN.Emplace(scaledData.m_totalRevenue);
49      inputNN.Emplace(scaledData.m_costOfRevenue);
50      inputNN.Emplace(scaledData.m_operatingExpenses);
51      inputNN.Emplace(scaledData.m_stockValue);
52
53      // Set the Neural Network and run it
54      m_simulationNN->SetInputFromArrayCopy(inputNN);
55      m_simulationNN->Run();
56
57      // Get the output from the Neural Network
58      float scaledOutput = 0.0f;
59      TArray<float> outputNN = m_simulationNN->GetOutputTensor().GetArrayCopy<float>();
60
61      // If it produced an output
62      if (outputNN.IsEmpty() == false)
63      {
64          // Unscale it
65          scaledOutput = m_dataScaler->UnscaleNetIncome(outputNN[0]);
66
67          // The NN should only produce one output, check this is the case
68          if (outputNN.Num() > 1)
69          {
70              UE_LOG(LogTemp, Warning, TEXT("The Neural Network Model is outputting more than 1 value"));
71          }
72      }
73      else
74      {
75          UE_LOG(LogTemp, Error, TEXT("The Neural Network Model did not produce any output"));
76      }
77
78      return scaledOutput;
79  }

```

Figure 63: Code to run the neural network in Unreal Engine

The wrapper class for the neural network hides the scaling of the data, so that in code it can be easily used with the unscaled values seen by the user.

7.2.1.2. Company/Game Manager

The core of the game simulation is centralised in a “Company” game object, which is stored and referenced in a general “Game Manager”.

The company class contains:

- The wrapper to the Neural Network, used to predict the net income.
- The financial data that serves as an input to this neural network and represents the current financial state of the company.
- Company specific gameplay variables, such as marketing and research investment bonuses, a cap to the amount that the net income can change each turn, etc.

This class handles any gameplay tweak related to the balance of the company simulation component of the game. This includes things like applying different gameplay boosts based on the investment levels of the various fields or rebalancing the current spending if the current budget can no longer cover the allocated monthly expenses. Besides this, the company class ensures that all the financial data is kept up to date per user actions and the neural network simulations.

The company object also provides access to the simulation functionality, overseeing the running of the simulation of the company performance through the neural network.

The monthly simulation consists of the following:

- Apply the marketing bonus: This increases the company stock price depending on the current bonus obtained from the marketing spending and serves as a direct neural network input.
- Calculate the monthly operation expenses: This direct neural network input is calculated based on the sum of all the monthly expenses that the company has.
- Calculate the net income: This runs the neural network simulation to obtain the net income. After this has been calculated, the net income is then capped to a certain amount of change each month to smooth any possible spike, and it is increased by a certain percentage based on the current research and development bonus unlocked.
- Update the date of the game: The fiscal date, which also serves as a direct input of the neural network, is advanced by one month for the next model run.
- Evaluate the end month stats: This updates the company's financial stats, like the monthly budget and cash available, based on the simulation of the previous month.

```
130 void UCompany::SimulateMonth()
131 {
132     // Calculate the stock value of the company based on its previous month performance
133     CalculateCompanyStockValue();
134
135     // Update the monthly operation expenses
136     CalculateMonthlyOperationExpenses();
137
138     // Calculate the new net income by running the Neural Network
139     // and applying any bonus/cap to it
140     CalculateNetIncome();
141
142     // Update the date
143     m_currentFinancialData.m_date = UCDateUtils::AdvanceMonth(m_currentFinancialData.m_date);
144
145     // Update any financial stat record
146     EvaluateEndMonthStats();
147 }
```

Figure 64: Code Simulate Month

As discussed in the design, the simulation of the neural network is affected by altering the inputs before the inference is done, or by altering the net income after it has been calculated by the neural network.

The Game Manager, in contrast, handles game-specific logic, like managing the turns in the games and checking if the conditions for the end of the game have been met. This provides a series of events, like when a game turn ends, which are broadcasted to any observer. Multiple objects can observe this event and trigger specific behaviour when the events happen.

Internally, this game manager inherits from the Unreal Engine “Game Mode” class. This allows the game manager to be widely available in different classes, per Unreal Engine architecture (static function to get the current game mode), allowing objects to observe the game state and subscribe to its events easily.

In the Unreal Engine editor, the game manager game mode is then inherited by a blueprint game mode class. The only purpose of this blueprint game mode is to, via visual scripting and game engine editor UI, simplify the access to gameplay settings for design purposes and to avoid having to recompile the code upon changes,

```
18 UCLASS(BlueprintType, Blueprintable, // Blueprintable so that it can be inherited from a Blueprint class, allowing it to customize the Default values
19 meta = (ToolTip = "Class for main game"))
20 class AUnprofitableStartupMainGameMode : public AGameModeBase
21 {
22     GENERATED_BODY()
23
24 public:
25     AUnprofitableStartupMainGameMode();
26     void InitGame(const FString& MapName, const FString& Options, FString& ErrorMessage) override;
27
28     /**
29      * Ends the current turn, advancing the company simulation
30      */
31     UFUNCTION(BlueprintCallable, Category = "Main Game Mode")
32     void EndTurn();
33
34     UFUNCTION(BlueprintPure, BlueprintCallable, Category = "Main Game Mode")
35     UCompany* GetPlayerCompany() const;
36
37     UFUNCTION(BlueprintPure, BlueprintCallable, Category = "Main Game Mode")
38     FDateTime GetStartDate() const;
39
40
41
42     UPROPERTY(BlueprintAssignable, BlueprintCallable,
43 meta = (ToolTip = "Event raised when the current turn is ended"))
44     FdelegTurnEnded OnTurnEnded;
45
46     UPROPERTY(BlueprintAssignable, BlueprintCallable,
47 meta = (ToolTip = "Event raised when the game ends"))
48     FdelegGameEnded OnEndGameConditionChanged;
```

Figure 65: Game Manager Public Members

To other classes, the game manager provides access to the player company and gameplay related events, like when the turn ends, or the game is finished.

As for usability purposes, to avoid the user having to worry about practically insignificant decimal points, the financial data is stored as an integer. But because the financial data handled in the game can quickly go into the hundred million or even billions, integer64 is used to avoid an integer overflow if the cap of 2,147,483,647 is ever reached. This also uses signed integers to maintain compatibility with the Blueprint system since it does not support unsigned types.

7.2.1.3 UI integration

All the UI functionality in the game was implemented using the Unreal Engine visual scripting Blueprint system. This was done to simplify the adjustment of the UI layout on the screen, without having to recompile the code, as well as streamlining the process of referencing the assets (sprites, sounds, etc.) since there can be reference with the engine explorer GUI, rather than specifying a file path manually through code.

Despite the use of Blueprints, most of the actual gameplay functionality is done via C++ functions that, using Unreal Engine specifiers, were exposed to this visual scripting system. This means that the UI subscribes to the same C++ events to trigger actions and uses the same C++ property accessors to get the gameplay data to display.

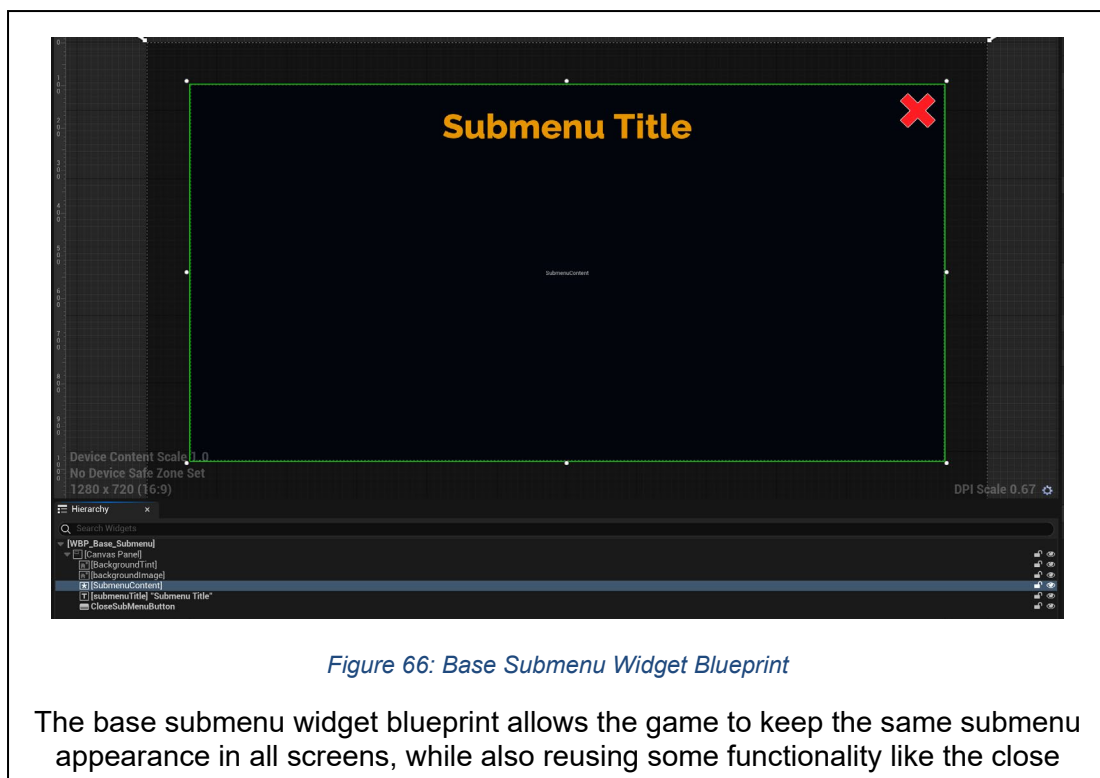
A significant concern of any UI development is always keeping the UI uniform throughout its different screens. In CSS, this could be done by creating a class that groups all the styling properties of the UI element, which can then be later used throughout the entire game.

Sadly, Unreal Engine does not have an accessible way to achieve this. Therefore, two different approaches were taken to preserve the uniformity of the UI:

- **Template Base Classes:** Creating a base class of the different UI elements (buttons, text, etc.) that contains the desired styling. This will then be set up inside another Unreal Engine widget for use, allowing customisation of the content of the UI element (e.g. changing the characters shown by a text element but templating its appearance). This had the disadvantage that the Unreal Engine class for these UI elements hides many of these properties in subsequent child classes. Therefore, for each different property or interaction event that needed to be customised in the “parent” widget, an accessor had to be created to set the original property.
- **Utility setter function:** In some cases, hiding the game interaction events in a child class, as done with the inheritance method, was impractical since access to these events could not be easily achieved. In these cases, a utility function to set the styling of that UI element was created. With this, all the elements where this function was applied will have the same appearance while preserving the accessibility to their properties. However, it has the obvious downside that this function had to be manually called for each UI element manually.

An example of the templated base class method of preserving UI is with the game buttons. A base class button was created to define its appearance, and this base class provided additional properties to set properties like the text of the button or binding to its interaction events (button clicked, etc.). Then, menus could use this base button appearance but tweak its functionality.

An example of the utility function approach was the use of HUD text to display financial information. Unreal Engine has a text binding event that refreshes the UI text whenever its value changes. However, since there is no easy way to override this event in a base class, the style of all these text elements is set via this utility function at startup. In that way, the text has all the same style and access to the binding event.



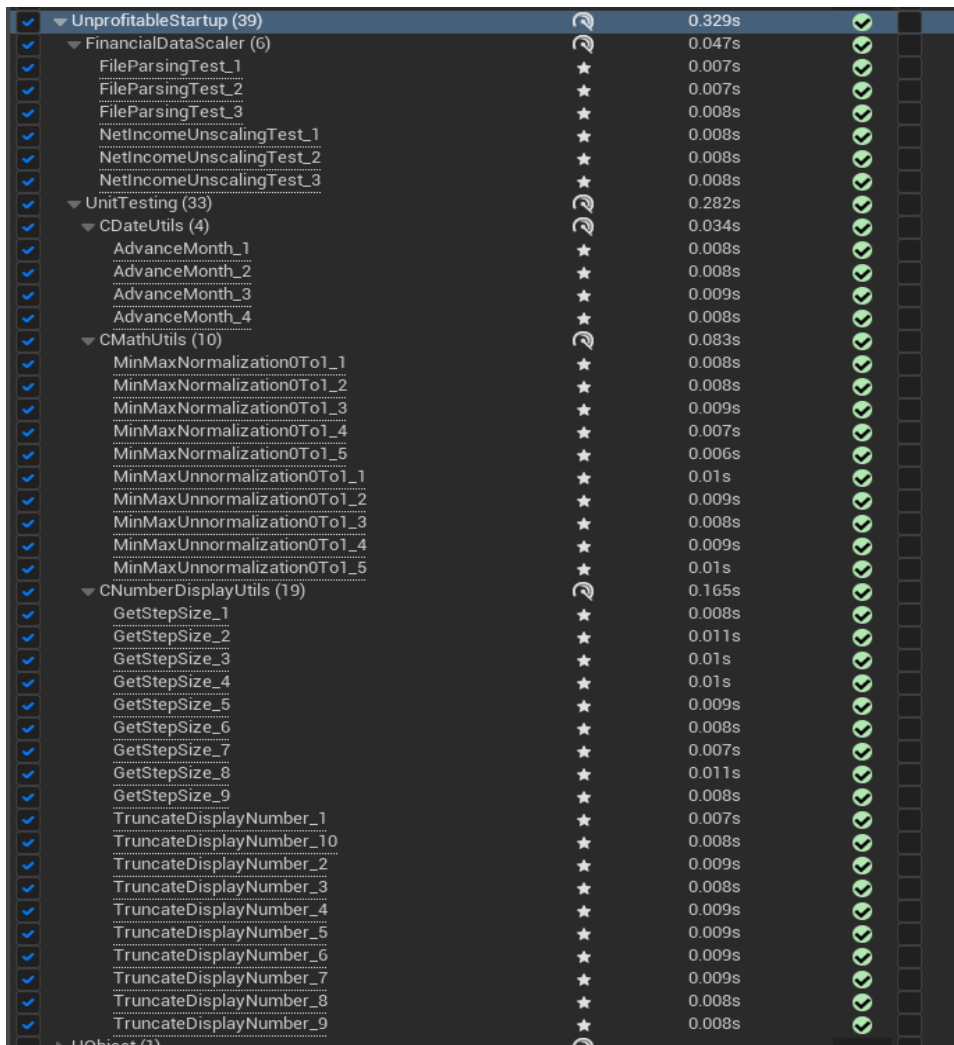
8. Testing

The project employed different types of testing to ensure that the features implemented successfully fulfilled their respective requirements and that the project was as bug-free as possible. As per the agile methodology, the project was tested at each iteration through development.

8.1. Test coverage

As a game development project, most of the testing covers the “front-end” or game component of the project. However, a significant portion of the testing was also done to check the neural network “back-end”. This includes testing the Python scripts used to get the financial data and to train the neural network, as well as the integration of the neural network with the game engine.

In its current iteration, the project is exclusive to the Windows platform. This means all the testing was done on a single platform, and no platform or hardware-specific testing was required.



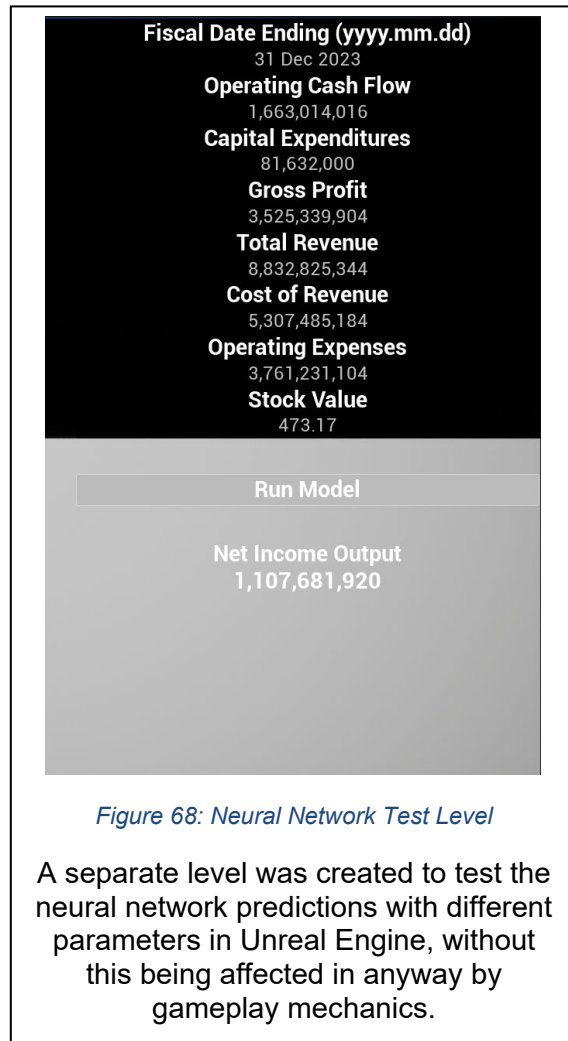
Test Name	Duration	Status
UnprofitableStartup (39)	0.329s	Pass
FinancialDataScaler (6)	0.047s	Pass
FileParsingTest_1	0.007s	Pass
FileParsingTest_2	0.007s	Pass
FileParsingTest_3	0.008s	Pass
NetIncomeUnscalingTest_1	0.008s	Pass
NetIncomeUnscalingTest_2	0.008s	Pass
NetIncomeUnscalingTest_3	0.008s	Pass
UnitTesting (33)	0.282s	Pass
CDateUtils (4)	0.034s	Pass
AdvanceMonth_1	0.008s	Pass
AdvanceMonth_2	0.008s	Pass
AdvanceMonth_3	0.009s	Pass
AdvanceMonth_4	0.008s	Pass
CMathUtils (10)	0.083s	Pass
MinMaxNormalization0To1_1	0.008s	Pass
MinMaxNormalization0To1_2	0.008s	Pass
MinMaxNormalization0To1_3	0.009s	Pass
MinMaxNormalization0To1_4	0.007s	Pass
MinMaxNormalization0To1_5	0.006s	Pass
MinMaxUnnormalization0To1_1	0.01s	Pass
MinMaxUnnormalization0To1_2	0.009s	Pass
MinMaxUnnormalization0To1_3	0.008s	Pass
MinMaxUnnormalization0To1_4	0.009s	Pass
MinMaxUnnormalization0To1_5	0.01s	Pass
CNumberDisplayUtils (19)	0.165s	Pass
GetStepSize_1	0.008s	Pass
GetStepSize_2	0.011s	Pass
GetStepSize_3	0.01s	Pass
GetStepSize_4	0.01s	Pass
GetStepSize_5	0.009s	Pass
GetStepSize_6	0.008s	Pass
GetStepSize_7	0.007s	Pass
GetStepSize_8	0.011s	Pass
GetStepSize_9	0.008s	Pass
TruncateDisplayNumber_1	0.007s	Pass
TruncateDisplayNumber_10	0.008s	Pass
TruncateDisplayNumber_2	0.009s	Pass
TruncateDisplayNumber_3	0.008s	Pass
TruncateDisplayNumber_4	0.009s	Pass
TruncateDisplayNumber_5	0.009s	Pass
TruncateDisplayNumber_6	0.009s	Pass
TruncateDisplayNumber_7	0.009s	Pass
TruncateDisplayNumber_8	0.008s	Pass
TruncateDisplayNumber_9	0.008s	Pass

Figure 67: Unreal Engine Unit Testing

Using Unreal Engine Automation System, multiple unit tests were created for core utility functions of the project.

The test plan covered the following categories of issues:

- Automated Unit Testing: Each custom utility function created for the game has its functionality tested for standard and edge cases.
- Appearance: This references anything to do with the 3D environment of the game (excludes any 2D UI). This involved testing that there were no major visual glitches and that the animated objects in the game behaved as expected.
- Financial Data: The Python script used to filter and export the financial data used to train the neural network had to be validated in concerns that both the export process and the interpolation of the quarterly data into monthly and weekly data worked correctly.
- Neural Network: The testing of the script to create, train, and export the neural network. This testing only covered the correct execution of the code, and it is not to be confused with evaluating the performance of the neural network done for 7.2.1.3 Evaluating the neural network. The testing of the export functionality covered both the neural network being exported as an ONNX model and the exporting of the scaling data JSON, used to scale/unscale the neural network parameters in-game.
- Builds: Unreal Engine has some unique configurations when packaging certain types of files. These files had to be exported in their original format, without being packaged/compressed into an Unreal Engine specific asset type. This included both the scaling data JSON that is parsed, as well as the mp4 videos that are played in the game.
- Performance: The game performance was measured to ensure there was no visible lag, stutters, or any other performance issue. The performance metrics were captured via the frames-per-second metric. This testing emphasised evaluating the performance the moment the neural network simulation was run, which should, in theory, be the more expensive task of the game.
- UI: These tests included checking that the UI displayed the correct information related to the game status and that it behaved as expected (e.g. opening the pause menu). These tests also included different usability aspects of the UI, like ensuring tooltips were shown when hovering over objects or displaying financial data in an easy-to-read format.
- Sound: The correct sound is played whenever specific actions are performed, including the continuous playback of background music.



- **Gameplay:** The main gameplay mechanics were tested to ensure that they performed as expected, do what they are supposed to; as well as testing the game does not do what is not supposed to, the player's actions are restricted by the game design.

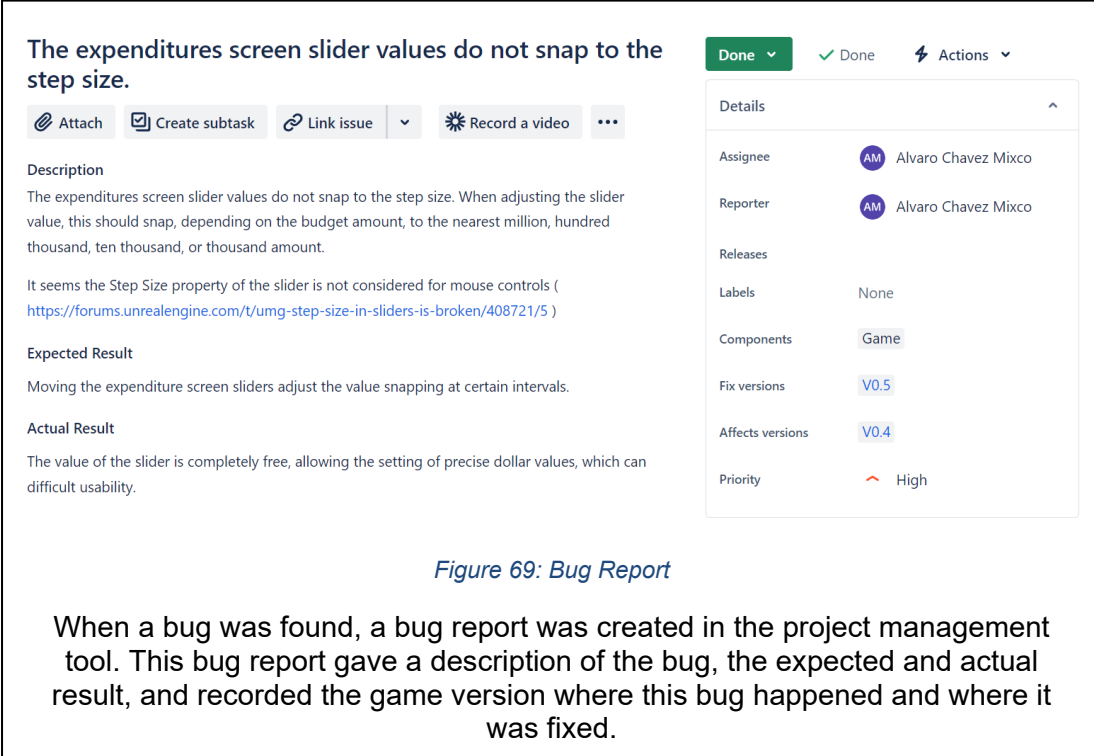
As one of the project's main features, special attention was also given to testing the correct integration of the neural network with the Unreal Engine inference system. This included testing:

- The neural network is correctly imported into Unreal Engine
- Outside of the "Unprofitable Startup" game setting, running the neural network without affecting it by any type of game design balance parameter gives similar results to running the neural network on the PyTorch environment. This involved running the neural network in the game engine and in the PyTorch training framework, with the same inputs, and evaluating the predicted net income output produced by both.
- The reproducibility of the neural network: running the neural network with the same inputs produces the same output. The neural network is a black-box algorithm with opaque inner workings. So, to ensure a consistent gameplay experience, it needs to be verified that the neural network behaves the same every time it is run.
- The neural network receives inputs that have been scaled to a zero-to-one range and produces a net income output that has also been scaled to a zero-to-one range. The game does not present these scaled values to the user or use them for gameplay. Therefore, it must be verified that the conversion of these unscaled values, used by the game, into scaled inputs used by the neural network, and vice versa, is accurate. This ensures that the neural network receives valid values from which to do the simulation and that the user sees the proper value produced by the neural network.

8.2. Test Methodology

The main types of testing for this project were white box testing, which I performed as a developer with knowledge of how the game works, including testing the functionality of any implemented feature as well as black box testing, which focused on user experience testing from people that do not have any knowledge of how the project works.

The testing methodology extensively used the Jira project management tool to keep track that new features were being correctly implemented and tested, as well as maintaining a record of any bug found during testing.



The expenditures screen slider values do not snap to the step size.

Attach Create subtask Link issue Record a video

Description
The expenditures screen slider values do not snap to the step size. When adjusting the slider value, this should snap, depending on the budget amount, to the nearest million, hundred thousand, ten thousand, or thousand amount.

It seems the Step Size property of the slider is not considered for mouse controls (<https://forums.unrealengine.com/t/umg-step-size-in-sliders-is-broken/408721/5>)

Expected Result
Moving the expenditure screen sliders adjust the value snapping at certain intervals.

Actual Result
The value of the slider is completely free, allowing the setting of precise dollar values, which can difficult usability.

Details

Assignee	AM Alvaro Chavez Mixco
Reporter	AM Alvaro Chavez Mixco
Releases	
Labels	None
Components	Game
Fix versions	V0.5
Affects versions	V0.4
Priority	High

Figure 69: Bug Report

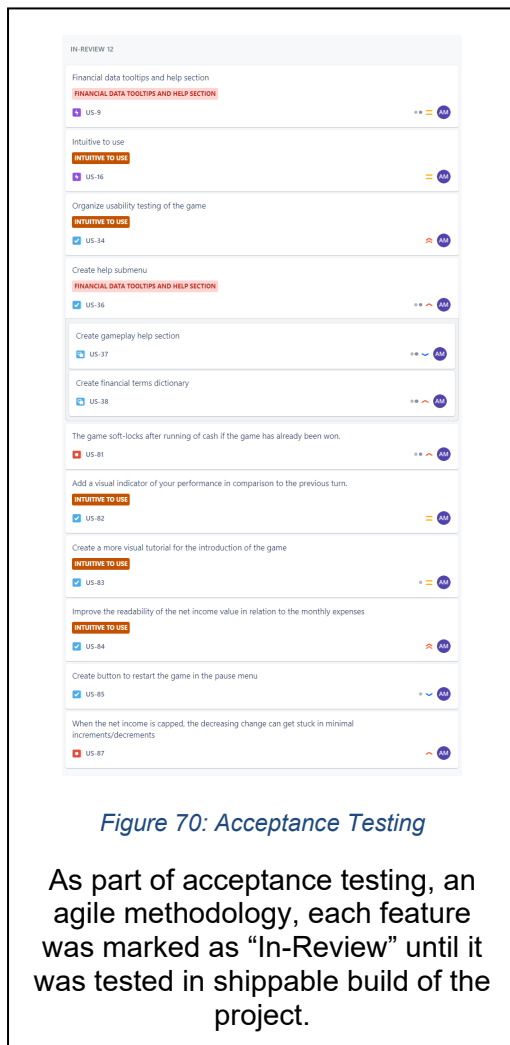
When a bug was found, a bug report was created in the project management tool. This bug report gave a description of the bug, the expected and actual result, and recorded the game version where this bug happened and where it was fixed.

8.2.1. Whitebox testing

The whitebox testing component of the project was done at two main points:

- When the feature is first implemented, edge cases for the feature are tested, and automated unit tests are created when needed.
- At the end of each weekly sprint, where the current state of the game was tested in a build of the project. As per agile methodology, this sprint build should, in theory, be a “shippable” version of the game, a version that, while not feature-complete, can be tested and evaluated.

Some of the forms of whitebox testing done during the project were:



- **Acceptance Testing:** After implementing each feature, before officially marking it as complete, this was set as “In-Review. The new feature would then be tested in the weekly sprint build. This ensured that the feature fulfilled its planned purposes and was bug-free.

- **Regression Testing:** A constantly growing list of test cases with each new feature added to it was tested for each sprint build of the game. This helped ensure that each subsequent game version kept the features implemented in previous versions working as expected.

- **Integration Testing:** A major component of the project was the integration of the neural network model with the game engine, Unreal. In practice, this meant testing that the neural network inside Unreal behaved in a similar way to as it behaved in the training framework.

- **Unit Testing:** Using Unreal Engine automated testing tools, different unit tests were created for the different utility functions (reusable class agnostic functions). These tests covered the behaviour of these functions with normal parameters, as well as with certain edge cases.

- **Verification Testing:** At the end of each sprint, the weekly build helped to verify that the project code implementation was

working correctly. This included checking that the code did its intended purpose and was bug-free.

- **Validation Testing:** The project's progress was frequently presented to the project supervisor to ensure that this met the requirements and original scope given in the project proposal.

Besides this, during development, different self-testing techniques were done to ensure that all the new code worked as expected, including:

- Using breakpoints in the code to ensure the new code gets called and analyse the call stack at that given moment.
- Logging messages to the Unreal Engine console to visualise the data or to notify about potential warnings or errors with the code.
- Creating test levels in the game to test specific features in isolation, most noticeably the scaling of the financial data and the predictions of the neural network.
- In the Python scripts for the neural network, multiple flags were created to control how verbose the printing of the information is. Besides, the scripts have graphs and more detailed comparisons to verify the correct functionality of the code.

Case ID	Description	Expected Result	Category	Pass/Fail
1	All custom-made automated tests, "UnprofitableStartup" category, in Unreal Engine (Tools->Test Automation) are run.	All automation tests in the Unreal Engine project are successful	Automated Testing	Pass
2	The script `NeuralNetworkTraining.py` works correctly, training the neural network and exporting it	The Neural Network model is exported an ONNX file, and the Neural Network performance evaluation is acceptable	Neural Network	Pass
3	The script `NeuralNetworkTraining.py` produces accurate data to scale/unscale the information from Unreal Engine.	The script produces `scaling_data_information.json` which correctly has the min and max values for each column.	Neural Network	Pass
4	The data scaling JSON is exported as an unpackaged JSON in builds. This opposite of most of the game data, which is packaged, preventing it from being parsed.	The data scaling JSON is copied as `Content/Data/scaling_data_information.json` in builds, remaining unpackaged.	Builds	Pass

5	The Neural Network, in isolation, produces consistent output results when the same input is given.	The Neural Network, in the Unreal Engine test scene, produces similar results to the PyTorch Neural Network script	Neural Network	Pass
6	When the end turn button is pressed the game progresses the next turn	The end turn button is clickable, and refreshes the game Hud with the latest data (next turn number, overall profits, etc.)	Gameplay	Pass
7	The financial data HUD shows tooltips when it is hovered over	Tooltips for the different UI elements are shown when they are hovered over	UI	Pass
8	The game HUD tracks the UI correctly	The game HUD accurately tracks the turn number, net income, and other financial data.	UI	Pass
9	The net income HUD text changes depending on its value	When the net income is positive, or zero, a '+' is prefixed and the text turns green; when negative a '-' is prefixed and text colour is set to red.	UI	Pass
10	In the UI, financial data is shown up as rounded up values	Values are shown rounded up, when above 100 units, for thousands (k), millions (m), and billions (b)	UI	Pass
11	The script "FinancialData-AlphaVantageData.py" correctly filters the data and exports it as quarterly, monthly, and weekly	Running the Python script produces 3 files, with only the reduced data fields, for "quarterlyFinancialData.csv", "monthlyFinancialData.csv", and "weeklyFinancialData.csv"	Financial Data	Pass
12	The interpolated monthly and weekly financial data matches closely the original quarterly data obtained from AlphaVantage.	When running "FinancialData-AlphaVantageData.py", the value ranges and the graph representation of the quarterly data is close to the monthly and weekly data.	Financial Data	Pass
13	Clicking the expenditures button opens the Expenditures submenu popup screen	When the expenditures button in the main game UI is clicked a new submenu screen, labelled Expenditures, is shown on top of the UI	Gameplay	Pass

14	Other UI elements in the main game UI cannot be interacted with when a submenu screen is open.	Open the Expenditures submenu screen, and click the end turn button or the Expenditures button (again), which are visible under the submenu. Observe that nothing happens	UI	Pass
15	Submenu screens are properly hidden when the close button is pressed	Open the Expenditures submenu screen and click the close submenu button on the top right, notice that the submenu is closed/hidden from view and no error shows up.	UI	Pass
16	In the expenditures screen, the Research and Marketing sliders cannot be below 0, or, add it up, cannot go above the current monthly budget.	In the expenditures screen, moving the slider to the left goes down to 0. The sum of both slider values when maxed, does not go above the monthly budget.	UI	Pass
17	The Game HUD is updated when the expenditures screen is closed.	Opening the expenditure screen and changing the budget allocation updates the monthly expenses amount, this change is reflected in the main game HUD when the submenu is closed	UI	Pass
18	When opened, the expenditure screen shows the most up to date values for monthly budget, available funds for the month, and marketing and research expenses	The expenditure screen shows the correct monthly budget amount, available funds (monthly budget - monthly expenses), and marketing and research expenses (sliders correctly placed, and text displaying correct values)	UI	Pass
19	Adjusting the sliders in the expenditures screen changes the company's budget allocation	When the marketing and the research sliders are moved, the budget allocation for that particular expense is updated in the company. These changes are reflected in both the text under the slider, and the currently available funds amount.	Gameplay	Pass
20	The expenditure screen sliders snap their value per the nearest rounded up value, depending on the current budget amount.	When moving the expenses sliders in the Expenditures screen, this snap in values to the corresponding step size of million, hundred thousand, etc. This step size on the current monthly budget.	UI	Pass

21	The expenditures allocations are rebalanced, proportionally, if the new budget (after the end of a turn) has been reduced and is no longer enough to cover the allocated expenses	Having the budget fully allocated, ending a turn that has a reduced budget, causes the expenditures to be re-allocated proportionally per the new budget. This use rounded down step size values and may not cover the 100% of expenses.	Gameplay	Pass
22	Clicking the Company Details button opens the Company Details submenu popup screen	When the Company Details button in the main game UI is clicked a new submenu screen, labelled Company Details, is shown on top of the UI	Gameplay	Pass
23	The company details screen shows the correct and up to date information when opened.	When the Company Details screen is opened, this shows the latest up to date information. This has the currency data shown as rounded up in thousands (k), millions (m) or billions (b) when appropriated.	UI	Pass
24	In the Company Details screen when at the max investment level, display the funding required for the next level as ` `	Open the company details when any expense has reached the max marketing expense level and notice the funding for the next level is shown as ` `	UI	Pass
25	Clicking the `Pause` button opens the Pause submenu popup screen	When the Pause button in the main game UI is clicked a new submenu screen, labelled Paused, is shown on top of the UI	Gameplay	Pass
26	Pressing `Escape` or `P` toggles the pause menu on and off	Pressing the `P` or `Escape` key on the keyboard toggles the pause menu on and off, showing it and hiding it	Gameplay	Pass
27	Pressing the `Resume` button in the pause menu closes the Pause menu. This considering that the Pause menu has no Close button like the other submenus.	Pressing the `Resume` button in the Pause menu closes the menu screen.	Gameplay	Pass
28	Pressing the `Exit Game` button in the pause menu closes the game	Pressing the `Exit Game` button in the Pause menu exits the game in builds and editor.	Gameplay	Pass

	(build) or stops the editor play (Editor)			
29	Despite the Neural Network prediction results, the Net Income of the company cannot change more than the restricted amount (default of 10%) in comparison to the previous month. This is before any bonus is applied, like the research investment level bonus.	The Net Income during different months never changes more (positive or negative) than 10% in comparison to the previous month net income. This is ignoring any bonus in the game.	Gameplay	Pass
30	Marketing and Research expenses can be improved based on investment levels	Open the company details screen and notice that the investment level is increased whenever the required amount of funding is met	Gameplay	Pass
31	Marketing bonus increases the stock value of the company	Open the Company Details screen and notice the current bonus of the Marketing investment. Advance turns and notice the bonus effects	Gameplay	Pass
32	Marketing bonus increases the net income of the company	Open the Company Details screen and notice the current bonus of the Research and Development investment. Advance turns and notice the bonus effects that increase the net income	Gameplay	Pass
33	The game performance, measured through its frame rate is good and steady, even on "expensive" actions like ending the turn/running the neural network simulation. This measured on a computer with performance similar to the ones in the university.	In the Unreal Engine editor, enabling the option to show FPS, and launching the game on fullscreen, the FPS remains steady.	Performance	Pass

34	The game assets present no major issue, like clipping, flickering or shadow artifacts.	Launching the game, the 3D environment presents no visual glitch	Appearance	Pass
35	There is a continuous day and night cycle.	The sun position changes throughout the game, simulating a day and night cycle	Appearance	Pass
36	The computers show animated videos at different intervals.	The computer screens in the game show different videos, at different intervals, where it is not obvious, they are the same videos. These are looping infinitely.	Appearance	Pass
37	The videos used in the game are correctly packed in the build.	The mp4 videos used in the game are in the "Content/Movies" folder of the project when packaged.	Builds	Pass
38	The buttons change their appearance when they are hovered or pressed.	Hovering over a button change its appearance, as well as when the button is pressed.	UI	Pass
39	There is a custom cursor image	The in-game cursor is always visible, and is shown with a custom image	UI	Pass
40	There is a sound effect when buttons are hovered or pressed	Different sound effects are played when a button is hovered, and when a button is pressed.	Sound	Pass
41	The sliders in the expenditures have sound effects for when the slider is grabbed, and for when the slider is released.	A sound is played when the slider is grabbed. If the slider is released on a valid range, not over the available budget, an ok sound is played. Otherwise, if the slider is released above the possible budget a "bad" sound is played	Sound	Pass
42	A sound is played when the close button is pressed.	Opening the company details screen, and pressing the close button plays a sound	Sound	Pass
43	A sound is played when the end turn button is pressed	When the end turn button is pressed a sound is played.	Sound	Pass
44	Looping background music is played throughout the game.	The game plays a constantly looping background music. The music continues playing even when the game is paused.	Sound	Pass
45	The day night cycle is stopped when the game is paused.	Opening the pause menu stops the current day night and cycle advance.	Appearance	Pass

46	When the game is started an introduction screen is shown	After starting the game, a welcome submenu with basic instructions is shown	UI	Pass
47	If the user runs out of cash available an end game screen is shown	If the user runs out of cash available, an end game submenu is shown, notifying the player he went bankrupt and giving him the option to restart the game	Gameplay	Pass
48	If the user survives (does not run out of cash) after the max number of turns he receives a victory screen	Advance the game to turn 60, without running out of cash. Observe a win game, notifying the player "survived" is shown. This gives the player the option to continue playing or restart the game.	Gameplay	Pass
49	If the company becomes profitable and it accrues a certain amount of cash, a victory screen is shown	If the player achieves a significantly positive net income and a good cash reserve, a victory screen is shown informing the player he made the company successful. This gives the player the option to continue playing or restart the game.	Gameplay	Pass
50	A sound effect is playing when the end game condition is played.	Different sound effects are played depending on the condition the game ended, bankruptcy, max turns, or profitability	Sound	Pass
51	The help menu can be opened	Clicking the help menu on the HUD opens the help popup screen. This display buttons with further subscreens of the help menu.	Gameplay	Pass
52	The help menu displays a how to play section	After opening the help menu, clicking the How to Play button shows a screen with instructions on how to play the game.	UI	Pass
53	The help menu displays a financial terms dictionary	After opening the help menu, clicking the Financial Dictionary button shows a screen with different financial terms used in the game and their definition.	UI	Pass
54	If you win the game, and decide to continue playing, you can still lose the game if you run out of cash.	Win the game, either by reaching the max turns or by earning enough money, and then progress the game until you run out of cash. Observe that the lose screen is shown.	Gameplay	Pass

55	A highlighted version of the HUD, with instructions, is shown at the welcome screen and at the help/how to play menu.	Notice in the introduction screen the image with instructions. Open the Help menu, How to Play, and notice the same screen with instruction is shown.	UI	Pass
56	The game can be restarted from the pause menu.	Open the pause menu, and click the Restart Game button, and the current level is reloaded.	UI	Pass
57	The net income has a minimum amount (by default 100,000) that can change each month before it is capped.	When the net income is reaching values near 0, notice that the net income is never capped to small, ever decreasing values.	Gameplay	Pass
58	There is an icon showing the change of the net income in relation to the previous month.	If, in comparison to the previous month, the net income has increased a green arrow is shown next to the net income HUD text. If this has decreased, a red arrow is shown. In the odd case that it stays the same, an equal sign is shown.	UI	Pass

Table 19: Version 0.8 Test Cases (Condensed)

See [Appendix 5 – Versions test cases](#) for the full test cases of all versions of the project.

8.2.1. Blackbox testing

The blackbox testing for the project involved presenting users, which had no knowledge about how the game works, with a playable demo version of the game, for a 10–15-minute playthrough, and then having them fill out a short form based on their experience.

Because of the increased barrier of entry, requiring users to download the demo of the game on a Windows computer, and play it; this form gathered much less responses than the initial requirement gathering survey.

In this case, the main 2 ways used to get users to test the game were:

- Through an in-person event at the University of Westminster, organised by the Computer Games Development course staff/student representatives.
- By forwarding the new form, with the link to download the demo, to the people that had previously, during the requirement gathering survey, left their email for further contacts about the project.

8.2.1.1. Playtesting Results

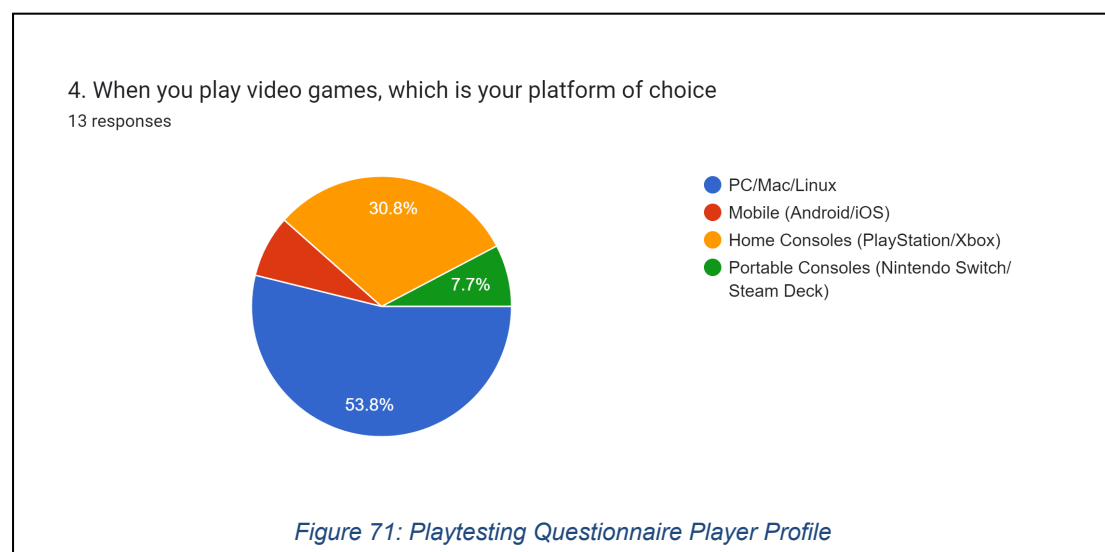
The results of the playtesting were gathered via a Google Form.

The playtesting gathered 13 responses, though as per the Participation Information Sheet, it was not required by the respondents to answer all the questions. This may lead to some questions having a smaller number of responses.

While the number of responses is relatively small, and could certainly be expanded, some trends can start to be observed from them.

To simplify the analysis of the responses, the form was divided into the following sections:

Player Profile

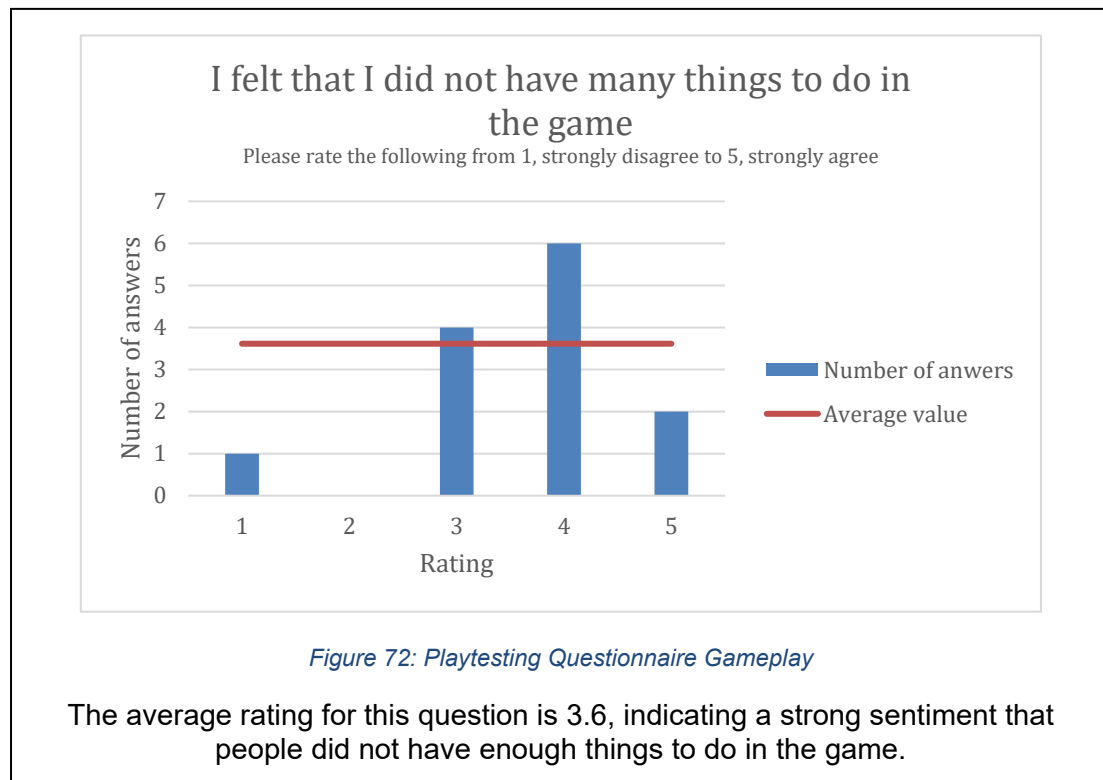


The responses on this section had the objective to create an anonymous player profile of the people answering the survey.

From these responses, we can see that the respondent's profile largely matches the game's target audience in both terms of age, with the age of 18-43 years old making up most of the respondents; and the platform of choice of the respondents being PC/Mac/Linux, which is the primarily target platform of the game (Yee, 2016) (AARKI, 2020).

The respondents' matching with the target audience highlights the importance of the respondents' opinions on the form since they are potential customers of the game.

Gameplay

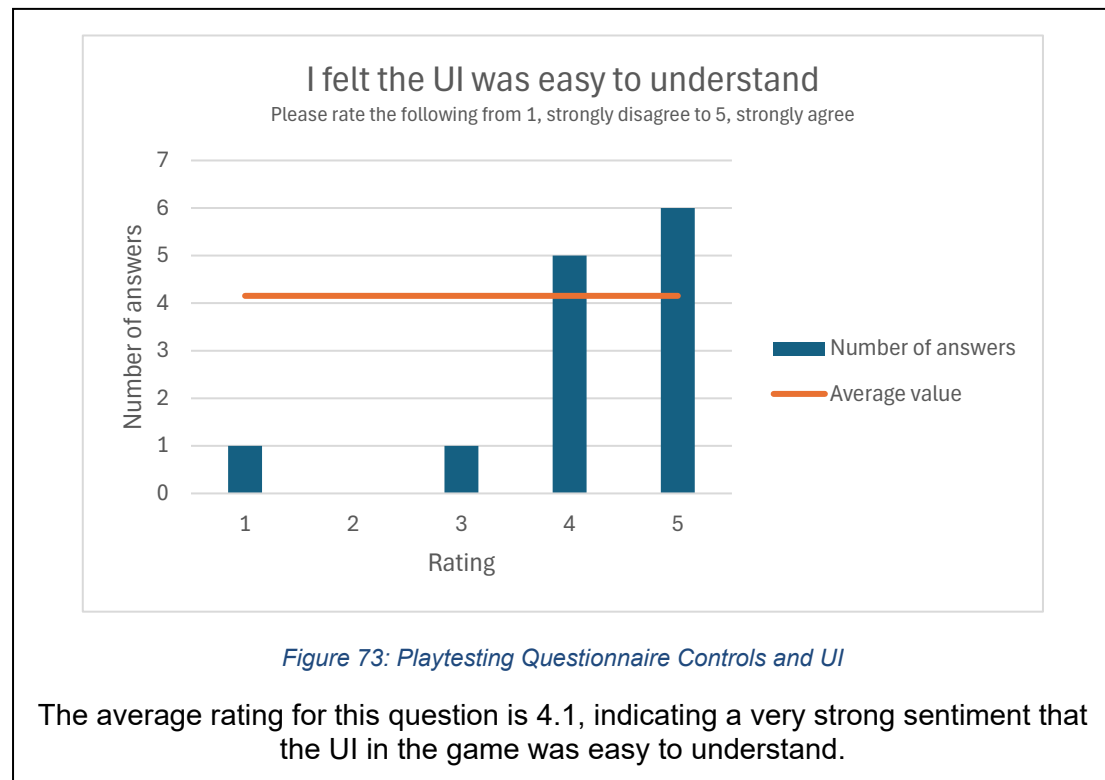


The objective of this section was to gauge respondents' opinion on how fun, fair, and innovative the gameplay of Unprofitable Startup was.

Though the majority of users found the gameplay unique, the responses on the game being fun and easy to understand were much more mixed, though still on the positive side.

However, the biggest pain point in these responses was that players felt there was not enough content or things to do in the game. This means that potential customers may not consider the game to be worth their money, since it does not provide enough things to do. This criticism is further highlighted by the fact that respondents only played the game for a short duration of time (suggested 10-15 minutes), rather than an extended play session.

Controls and UI

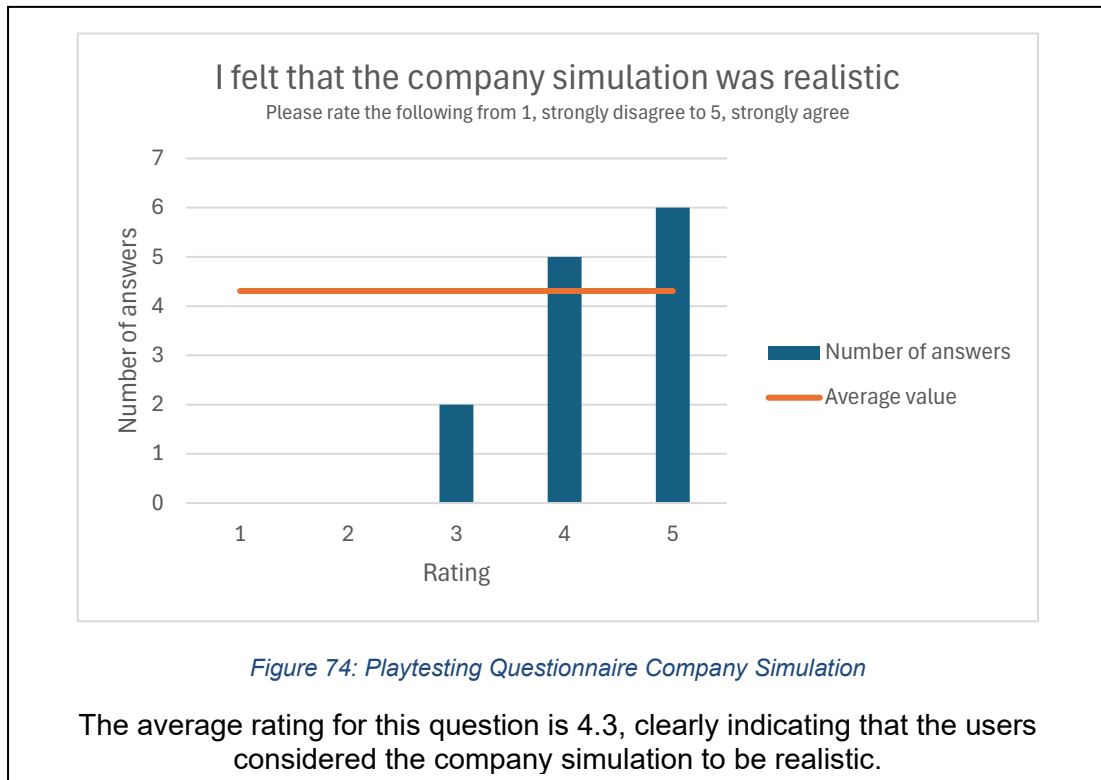


As it has been highlighted in the project design [6.6.1. User usability](#), user experience has been a major focus of the project, especially since this is a common weakness of the game genre. The responses in this section aim to measure how easy it is to play the game, and to understand the game.

The responses on this area were generally positive, with most respondents expressing they were able to understand the game controls and make of sense of the information being displayed on the screen.

In the additional feedback questions of this section, where respondents were able to freely type any extra comment regarding the Controls and UI, multiple respondents mentioned possible improvements. This were mostly focused on improving the tutorials and on creating notifications whenever certain events happened.

Company Simulation



As the integration of the neural network into the simulation of the game company is one of the main focuses of this project, this entire section was focused on measuring users' perception of it.

The responses mostly indicate that the neural network performed a realistic company simulation. However, since this is a game, the recreational aspects of the simulation also had to be evaluated, aspects in which the neural network performed worst.

Though still rated somewhat positively, the response to the simulation being fair, and being relatable to the player's action (users being able to understand how their actions affect the simulation) were certainly worst. This is most likely caused by the lack of transparency that the neural network algorithm has (Bourg & Seemann, 2004).

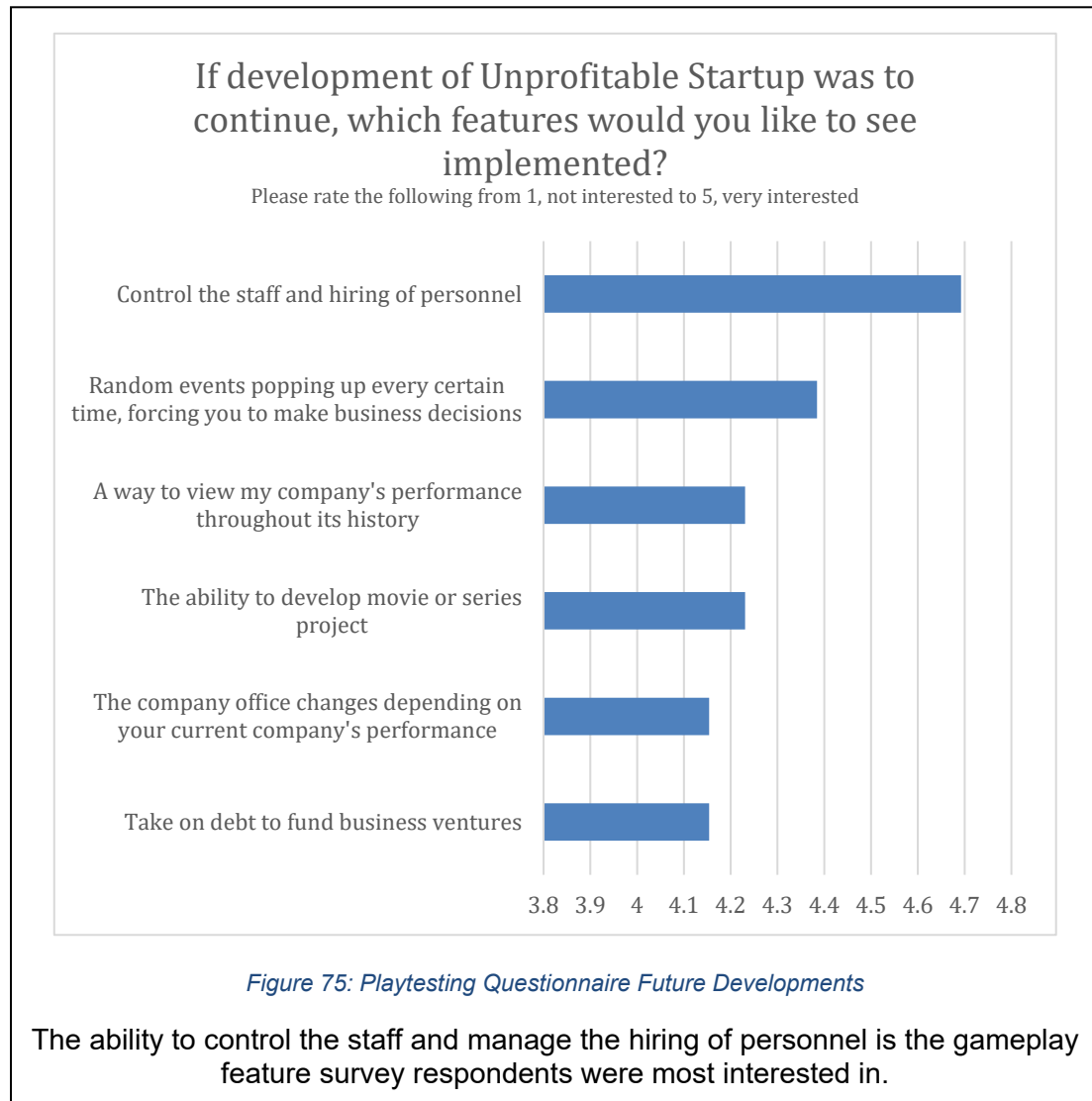
Future Developments

As a last point, the respondents were asked to rate their interest on several features that, due to time constraints of the project, were not implemented in the demo version of the game that was tested.

These ratings not only help to measure the potential interest of the audience in specific features, which can then be used to re-prioritise them; but it also could be used to deduct aspects of the game that were lacking.

For example, the most requested features had to do with adding more things for the players to do, rather than features to improve the usability or appearance of the game. This highlights the observations of the previous section responses, which indicated the lack of content.

This section of the survey would serve as a starting point for any further work on the project since it provides an analysis of what needs to be done next from a user perspective.



Full survey responses in [Appendix 3 – Playtesting responses](#)

8.2.1.1. Changes from playtesting

Based on usability testing the following new issues were created:

- US-82: Add a visual indicator of your performance in comparison to the previous turn. **DONE**
- US-83: Create a more visual tutorial for the introduction of the game. **DONE**
- US-85: Create button to restart the game in the pause menu. **DONE**
- US-86: Create a more in-depth tutorial. **BACKLOG**
- US-88: Create quality settings to better adjust to multiple devices. **BACKLOG**
- US-89: Create audio settings to adjust the volume of the game. **BACKLOG**

Also, for the record, these already schedule features were referenced:

- US-10: Performance representative animated background. **BACKLOG**
- US-73: Allow typing precise values in the expenditure screen. **BACKLOG**
- US-69: Create in-game notifications for certain important events. **BACKLOG**
- US-30: Create Market History screen. **BACKLOG**
- US-60: Warn the player when he is close to losing the game. **BACKLOG**

Figure 76: Blackbox Testing Requested Features

The new features to develop based on the blackbox testing were recorded in the project management tool for either immediate development or as future development.

The main focus of this blackbox testing was to test how much fun the game concept was and, as indicated in the design of the project, to improve the overall user experience in the game.

The feedback provided in this testing helped improve the game immediately with new features to improve the game usability (like adding a visual explanation of the game HUD and a visual indicator of the net income performance from month to month), as well as to guide potential future development of the project. For future development, this feedback raised new potential features to develop (an in-game tutorial level and more game settings) and reinforced ideas initially planned for the game, such as the luxury functional requirement of a “Performance representative animated background.”

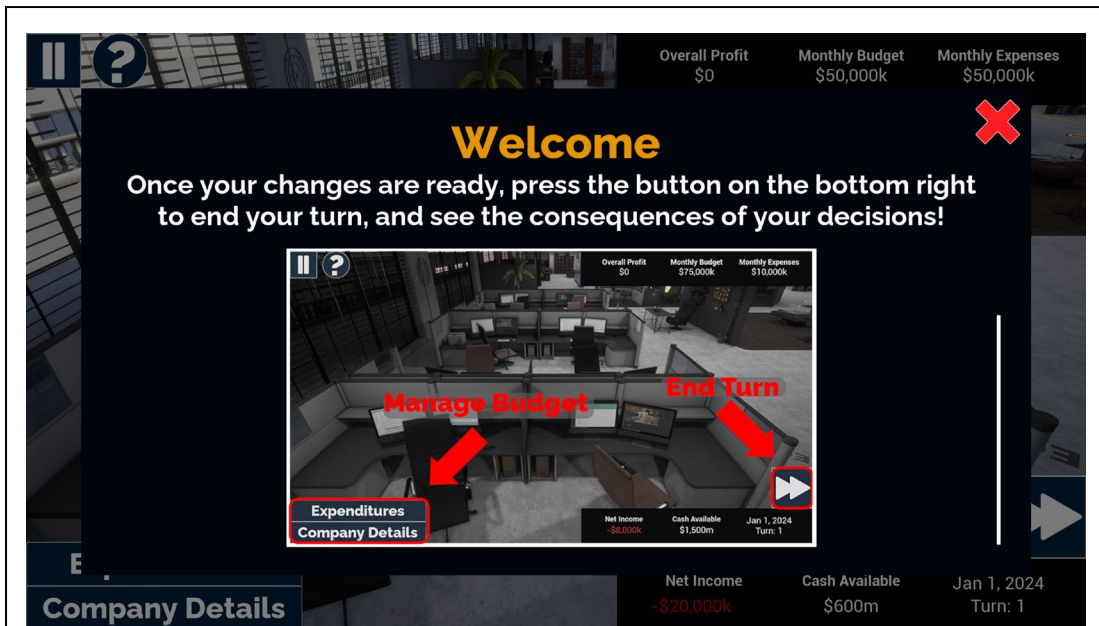


Figure 77: Game HUD Visual Instructions

As an immediate effect of the blackbox testing, a visual explanation of the game HUD was added, rather than relying purely on text to explain how to play the game.

9. Conclusions and reflections

Overall, the project could be considered a success since it met all its aims and their respective objectives. It managed to:

- Allow the simulation of a company's financial performance.
- Give players control of the company budget, allowing them to affect the company's net income.
- Create and train a neural network capable of predicting a company's net income and integrating it into a game engine.
- Create the basis of a simulation management game.

Per 3.3 List of requirements all the essential requirements of the project were met, with some additional desirable requirements fulfilled as well. However, even if the requirements were technically met and work had been developed for them, some of them could be further improved.

Such as the non-functional requirement for `Intuitive to use`. Despite much work being done to fulfil this requirement, with much emphasis on usability, more improvements could still be made. However, this process is iterative, requiring constant testing by users.

No.	Requirement	Status	Implementation	Priority
1	Net Income simulation based on player's actions	Met	The player's actions affect the company's net income.	Essential
2	Manage budget	Met	The player can manage the company budget in the Expenditures screen	Essential
3	Turn-based actions	Met	The player can end his turn to simulate a new month for the company	Essential
4	Animated background	Partially met	Animated computer screens and a day night cycle make the 3D background animated. However, no character models are present.	Desirable
5	Main Menu	Not met		Desirable
6	Pause Menu	Met	There is a pause menu allowing the player to pause, restart and quit the game.	Desirable

7	Movies and Series Projects	Not met		Desirable
8	Sounds and music	Met	Each UI element has a sound effect, and there is background music	Desirable
9	Financial data tooltips and help section	Met	The game HUD and menus have tooltips, and there is a help menu with a How to Play section and a Financial Dictionary.	Desirable
10	Performance representative animated background	Not met		Luxury
11	Debts issuance and repayment	Not met		Luxury
12	Auto-run mode	Not met		Luxury
13	Random events	Not met		Luxury
14	Options Menu	Not met		Luxury

Table 20: Functional Requirements Fulfilled

See [Appendix 7 – Completed work](#) for details on issues/requirements completed.

Because of time constraints, not all desirable and luxury requirements for the project could be met since the priority was obviously given to the essential requirements.

The agile methodology of the project proved effective, with the initial prototypes helping to test the viability of the game idea and the iterative agile process helping constantly adjust the priorities and scope of the project.

9.1. Weaknesses

From a technical standpoint, this project managed to test how the integration of a neural network in a game engine. So, besides the use case for this project of having a neural network to simulate a company, this has the potential for other applications of a neural network inside a game engine, like game AI or some form of editor tool to help with the development.

However, in terms of weaknesses, three main areas ended up as weak points in terms of development, balancing the game, and user experience. These are:

1. The neural network integration with the gameplay: The neural network proved challenging to balance from a gameplay perspective. The lack of transparency of the neural network made it hard to establish a correlation between the inputs of the neural network and their effect.

For example, from a gameplay perspective, reducing the cost to operate the company would be something good, since then the company would have more cash available and a bigger income. However, lowering the Cost of Revenue input for the neural network may not translate this same effect. Since the neural network could interpret that this lower cost is not the product of an optimisation in the company operations, but rather that the company has sold fewer products, which it would then reflect in the neural network predictions as a smaller net income.

2. Gameplay features: Simply put, the game lacks gameplay features to prevent it from becoming repetitive. This is a consequence of the scope reduction of the project due to the time spent developing and integrating the neural network. This resulted in the front-end game part of the project to be lacking in features. Consequently, the heuristic requirement for “The game does not stagnate, and the player feels the progress.” could not be appropriately met.
3. Unprofitable startup theme: Though this is primarily a programming focused project, and the overall graphics of the game are not a major issue. The game setting of managing an unprofitable startup does not really come across. The player does start with a negative net income, by design choice, however there is nothing else to reinforce this theme. This is even worse for the venture of the company, video streaming since there is nothing in the game that showcases or takes advantage of the fact that the player’s company is related to this business area.

Overall, while combining both machine learning and game development proved to be a challenging but rewarding learning opportunity, having to develop two different components did mean that not one of them could be fully polished or developed in-depth.

9.2. New knowledge and developed skills

9.2.1. Developed Skills

From a personal perspective, in terms of developing my software/game programming skills, I fulfilled all my goals. These required challenging myself in fields that I was not comfortable with and working with new technologies with limited documentation/tutorials.

The main skills I developed in this project, and which I had a personal goal to learn, were:

- Unreal Engine C++: Rather than just doing the C++ code for a section of a small-scale university coursework, I now have the experience developing a complete project with Unreal Engine C++. This required me to learn how to instantiate Unreal Engine objects, how their hierarchy works, and how to integrate my C++ code with other Unreal Engine systems, including their visual scripting system. This knowledge of Unreal Engine, complemented with my knowledge of Unity, means that now I am familiar with the two major game development engines (GDC, Game Developer, 2024).
- Python: I had always heard good things about Python and its potential, especially for machine learning and data science projects (Daigle & Staff, 2023). However, its limited usage in game development meant that I never had the opportunity to learn the language. However, with the rise in popularity of machine learning (Jones, 2024) this now appears to be a more important programming language to have in my personal toolkit.
- Neural Networks: Despite some basic familiarity with supervised machine learning from a University of Westminster module, my knowledge in the field was admittedly very limited. Despite still not considering myself anywhere near a machine learning specialist, I think I have gained enough knowledge in the field to understand most of the general concepts and terminology related to it. So, though I may not have developed enough skills to build a complex neural network system, I feel confident that I now have enough skills to research and further learn about this topic. Again, just having some basic knowledge of machine learning is becoming a more fundamental skill for software/game developers (Jones, 2024).

While I would not consider myself an expert on any of these skills, since this is still a relatively small-scope project, I do feel confident enough to write these skills in my resume and present them to employers.

9.2.2. New Knowledge

I think that one of the best outcomes of this project is the knowledge gathered regarding the integration of neural networks into Unreal Engine. The concept of neural networks in game development is not new, but the official support for this in game engines somewhat is. Especially with Unreal Engine, which released its first iteration of a neural network inference network with Unreal Engine 5.0 (Unreal Engine, 2022), and to this point, it is still considered an experimental feature.

This meant that besides a couple of tutorials and the official documentation for the plugin, there is practically no content regarding development using neural networks in Unreal Engine. Because of this, during development, I had to research the source code and documentation in detail to learn how to implement my intended game design with my custom neural network.

This leads me to believe that if this project were made public, it could potentially have a positive effect on the current shortage of information regarding this specific topic. Certainly not as an academic or research masterpiece, but as a simple example that other developers could look on for when they need to integrate their own neural networks with Unreal Engine, if just save them time for this process.

9.3. Further works

Due to time constraints and re-prioritization of tasks, multiple issues planned to satisfy the project's Desirable and Luxury requirements could not be implemented.

This, combined with [8.2.1.1. Playtesting Results](#) and an analysis of the current project conditions based on the [2.2 Review of projects / applications](#) and whitebox testing, guide some of the further work that could be done to improve the project.

9.3.1. Usability

Despite the vast work done to improve the user experience and the good results obtained in this area from [8.2.1.1. Playtesting Results](#) further work in this area is needed. This would require a continuous process, especially if new game features that add additional complexity are implemented in the game.

These changes could potentially involve:

- Create a more in-depth tutorial for the game that explains to the player step-by-step how to play. This could be through a separate tutorial level, or through a step-by-step guide in the main game.
- Improve how and when the player is notified of in-game events. This could involve things like warning the player when he is close to losing the game, advising him on how to improve, or a notification being shown when a particular bonus of an investment level has been unlocked.
- Some type of triggerable overlay can be used to indicate with arrows, text, or another method what each element in the screen UI is and what it does.



Figure 78: Example of Overlay Explaining UI [15]

Rome: Total War for iPad has an “advisor” overlay, which the player can tap to see a full breakdown of what each UI element in the screen is and what they do. A feature like this would help with the usability of Unprofitable Startup.

9.3.1. Neural Network Improvement

Because this project uses the neural network for recreational purposes and is not used to make any financial decisions with real-life consequences, the accuracy of the neural network is not a major concern as long as the predictions follow the general trend of the actual data.

However, due to my limited experience with this topic, experience which has grown considerably during the development of this project, I believe the performance of this could be further enhanced. This would probably require further research on more complex neural network architectures or some other form of machine learning algorithm.

Related to the neural network, a further area that could be improved is the gathering of data to train the neural network. The current data is only for a single video streaming company, Netflix. Ideally, to have more data and to get a general overview of the video streaming market overall, rather than a single company, the financial data of multiple companies would be required. But as mentioned in [6.3. Neural Network Design](#) this data would need to be filtered so that it only includes the video streaming vertical of those companies.

Besides this, the neural network would also need a more in-depth feature selection process to determine which inputs truly influence the net income output. This is not only to potentially improve the neural network accuracy but also to better establish the correlation between the inputs in the game. If an understanding of how each input correlates to the neural network is achieved, it will be easier to balance the game by tweaking these inputs. This would help improve the overall integration of the neural network with the game.

As a final point regarding further work in the neural network aspect, it is important to consider that the plug-in used by Unreal Engine to integrate with the neural network would likely have to be updated to its latest version (Unreal Engine, 2023). This is because the plug-in version currently used for the project has been deprecated, though it still works with older versions of the engine and the executables it produces. Even so, based on a preliminary inspection of the new plugin, their APIs should be similar.

9.3.2. Gameplay Features

As previously mentioned, one of the most significant weaknesses of the project is the lack of things to do in the game to prevent it from becoming stale or repetitive. To solve this, the following gameplay features could be implemented:

- Ability to manage company staff: This will give players control over the staff being managed in the company, including the skills of the staff hired and, when required, who to fire. The number of employees and their skills could vary widely depending on the player's choices, which would help add depth to the gameplay. Besides this, dealing with people rather than just the "company" could potentially help players feel more engaged with the game.
- Random events: Probably the best way to prevent a game from becoming repetitive is to add some degree of randomness to the game. So, events that force the player to make a financial decision, achieve a specific objective, or play a certain way at random intervals will help keep the game fresh and prevent all the turns from being the same.
- Movie or series projects: The player will be able to develop one-off projects that, if successful, can bring significant financial rewards to the player. The

ability to develop these multi-turn projects will give players an intermediate goal to aim for. This would also help reinforce the video streaming theme of the game, which was a significant weakness of the project.



Figure 79: Game Dev Story Project Feature [9]

Game Dev Story, a comparable simulation game, allows creating game projects as a feature. In it the player manages the parameter and direction of the game the company will produce.

9.3.3. Game Feel

As previously mentioned, the project's focus was on its functionality, not how the game looks or feels. Despite particular attention being given to the usability aspect of the game, more dedicated effort would be required to get the game to both look good and feel rewarding for the player to play.

This would come in three main components:

- Improving how the game looks: The assets in the current version of the game look good, especially when combined with Unreal Engine's well-regarded lighting system. However, this was still done using assets from the Unreal Engine Marketplace. So, to get a custom look for the game, personalised assets would be required (tooltip UI, animated characters, 3D models, etc.). These customised assets could help further reinforce the company's video streaming theme, which is one of the project's weaknesses.
- Make the game feel more rewarding: Either through sound effects, particles or some other type of feedback, the player should get some kind of "reward" or indicator when he has made a good choice, like increasing profits at the end of a turn.
- Company customisation: The player starts the game by managing a company; however, he has no personal connection to that company. So, there is no intrinsic incentive to make the company successful. Making the player feel like the company is his own, by allowing them to customise their name or logo, would intrinsically motivate the player to make the company successful. Which will make the player more engaged in his actual performance in the game.

Key	T	Summary	Status	P	Parent	Fix versions
US-69	✓	Create in-game notifications for certain important events	BACKLOG	☰	Accessible financial data	
US-77	✓	Add animated character models to game	BACKLOG	∨	Animated background	
US-41	✓	Create a custom appearance for the in-game tooltips	BACKLOG	⋮	Financial data tooltips and help section	
US-86	✓	Create a more in-depth tutorial	BACKLOG	⋮	Intuitive to use	
US-51	✓	Create animated background for the Main Menu	BACKLOG	∨	Main Menu	
US-50	✓	Create Main Menu level	BACKLOG	∨	Main Menu	
US-46	✓	Incorporate movie projects into the game	BACKLOG	☰	Movies and Series Projects	
US-48	✓	Add screen to start a new project	BACKLOG	∨	Movies and Series Projects	
US-47	✓	Add menu to see current projects in development	BACKLOG	∨	Movies and Series Projects	
US-89	✓	Create audio settings to adjust the volume of the game	BACKLOG	☰	Options Menu	
US-88	✓	Create quality settings to better adjust to multiple devices	BACKLOG	⋮	Options Menu	
US-44	✓	Add background music to the main menu	BACKLOG	∨	Sounds and music	
US-78	✗	Value overflow when reaching around 2,000,000,000 billions	BACKLOG	∨		
US-74	✓	Allow customising the player's company name	BACKLOG	∨		
US-73	✓	Allow typing precise values in the expenditure screen	BACKLOG	∨		
US-63	✓	Optimise how the Submenu Styling of the text elements is applied	BACKLOG	∨		
US-62	✓	Randomise the starting values of the Neural Network/Company for the game	BACKLOG	∨		
US-31	📄	US-30 Implement line-graphs to visualize financial data	BACKLOG	∨	Create Market History screen	
US-30	✓	Create Market History screen	BACKLOG	∨		
US-61	✗	The GBP £ symbol is not supported as a currency prefix in "TruncateDisplayNumber"	BACKLOG	⋮		
US-60	✓	Warn the player when he is close to losing the game	BACKLOG	⋮		

Figure 80: Backlog

At the end of the project, there is a backlog of 21 specific issues that can be implemented to improve the overall game; and this is not including some of the tasks required to fulfil the luxury requirements, which have not been broken down yet.

10. References

AARKI, 2020. *Role Playing and Strategy Games: User Demographics*. [Online] Available at: <https://www.aarki.com/insights/role-playing-and-strategy-games-user-demographics> [Accessed 03 February 2024].

Adams, E., 2013. *Fundamentals of construction and simulation game design*. 1st ed. s.l.:New Riders.

Alpha Vantage, n.d. *Alpha Vantage*. [Online] Available at: <https://www.alphavantage.co/> [Accessed 20 January 2024].

Aroussi, R., 2024. *yfinance*. [Online] Available at: <https://github.com/ranaroussi/yfinance> [Accessed 20 January 2024].

Ashton, H., 2020. *What does an Economy Designer in games do? Interview with Harry Ashton, Neon Play* [Interview] (7 October 2020).

Batchelor, J., 2023. *Has Unity repaired the damage done by its Runtime Fee plans?*. [Online] Available at: <https://www.gamesindustry.biz/has-unity-repaired-the-damage-done-by-its-runtime-fee-plans> [Accessed 15 February 2024].

Bobbitt, Z., 2021. *MSE vs. RMSE: Which Metric Should You Use?*. [Online] Available at: <https://www.statology.org/mse-vs-rmse/> [Accessed 06 February 2024].

Bourg, D. M. & Seemann, G., 2004. *AI for Game Developers*. 1st Edition ed. Sebastopol, CA, USA: O'Reilly Media, Inc..

Breviu, C., 2022. *Introduction to Deep Learning Models in Unreal*. [Online] Available at: <https://youtu.be/7aJQIOe1QTA?si=GsaetRAuEDZeUo0J> [Accessed 2023 December 18].

Brownlee, J., 2020. *Time Series Forecasting Performance Measures With Python*. [Online] Available at: <https://machinelearningmastery.com/time-series-forecasting-performance-measures-with-python/> [Accessed 05 February 2024].

Brownlee, J., 2021. *How to Choose an Activation Function for Deep Learning*. [Online] Available at: <https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/> [Accessed 01 February 2024].

Buffett, M. & Clark, D., 2008. *Warren Buffett and the Interpretation of Financial Statements: The Search for the Company with a Durable Competitive Advantage*. New York: Scribner.

Chugh, V., 2022. *Become UNSTOPPABLE with data*. [Online] Available at: <https://www.kdnuggets.com/2022/12/tuning-adam-optimizer-parameters->

[pytorch.html](#)

[Accessed 1 January 2024].

Daigle, K. & Staff, G., 2023. *Octoverse: The state of open source and rise of AI in 2023*. [Online]

Available at: <https://github.blog/2023-11-08-the-state-of-open-source-and-ai/>

[Accessed 23 April 2024].

Daisuke, M., Yuhei, M. & Perez, C., 2017. Forecasting Firm Performance with Machine Learning. *RIETI Discussion Paper Series 17-E-068*, May.

Daniel de Almeida Rocha, J. C. D., 2019. *Simulating Human Behaviour in Games using Machine Learning*. Rio de Janeiro, SBGames. Simpósio Brasileiro de Jogos e Entretenimento Digital..

DeepAi, n.d. *Feed Forward Neural Network*. [Online]

Available at: <https://deepai.org/machine-learning-glossary-and-terms/feed-forward-neural-network>

[Accessed 20 April 2024].

Durrani, A., 2024. *Top Streaming Statistics In 2024*. [Online]

Available at: <https://www.forbes.com/home-improvement/internet/streaming-stats/>

[Accessed 03 February 2024].

Engelbrecht, D., 2023. *Introduction to Unity ML-Agents : understand the interplay of neural networks and simulation space using the Unity ML-Agents Package*. New York: Apress L. P..

Epic Games, 2023. *Unreal Engine 5.2 Release Notes*. [Online]

Available at: <https://docs.unrealengine.com/5.2/en-US/unreal-engine-5.2-release-notes/>

[Accessed 29 January 2024].

Frames, W., 2022. *Bringing Deep Learning to UE5 — Pt. 2*. [Online]

Available at: <https://medium.com/@weirdframes/bringing-deep-learning-to-unreal-engine-5-pt-2-51c1a2a2c3>

[Accessed 20 December 2023].

Gaikwad, A. et al., 2023. *Profit Prediction for Businesses using Machine Learning Algorithms*. Coimbatore, India, IEEE.

GDC, Game Developer, 2024. *2024 State of the Industry*. San Francisco, CA, GDC, Game Developer.

Greenheart Games, 2012. *Greenheart Games*. [Online]

Available at: <https://www.greenheartgames.com/app/game-dev-tycoon/>

[Accessed 30 April 2024].

Hall, M., 2018. *Choosing A Project Management Tool For Game Development*. [Online]

Available at: <https://www.gamedeveloper.com/business/choosing-a-project-management-tool-for-game-development>

[Accessed 18 April 2024].

Hayes, A., 2021. *SEC Filings: Forms You Need To Know*. [Online] Available at: <https://www.investopedia.com/articles/fundamental-analysis/08/sec-forms.asp> [Accessed 20 April 2024].

Into Games, 2024. *Entry Level Skills Hub - Database - Junior Programmer*. [Online] Available at: <https://entrylevel.games/database/junior-programmer> [Accessed 26 March 2024].

Jeff Loucks, C. A. K. W. J. A., 2024. *Driven to tiers: Streaming video services look to up their profitability game with viewers*. [Online] Available at: <https://www2.deloitte.com/uk/en/insights/industry/technology/technology-media-and-telecom-predictions/2024/tmt-predictions-streaming-video-services-profitability-must-increase-in-2024.html> [Accessed 29 January 2024].

Jones, S., 2024. *AI and machine learning specialists to become the fastest growing jobs in the next four years*. [Online] Available at: <https://londonlovestech.com/ai-and-machine-learning-specialists-to-become-the-fastest-growing-jobs-in-the-next-four-years/> [Accessed 23 April 2024].

Kairosoft Co., Ltd, n.d. *Game Dev Story*. [Online] Available at: <https://kairosoft.net/game/appli/gamedev.html> [Accessed 30 April 2024].

Kelly, M., 2021. *Game Genres*. In: *Game Design & Development 2021*. Dublin, Ireland: Technological University Dublin.

Koeffel Christina, H. W. L. J. H. M. G. A. T. M., 2010. Using Heuristics to Evaluate the Overall User Experience of Video Games and Advanced Interaction Games. In: R. Bernhaupt, ed. *Evaluating User Experience in Games*. s.l.:Springer, pp. 233-256.

Lado-Sestayo, R. & Vivel-Búa, M., 2020. Hotel profitability: a multilayer neural network approach. *Journal of Hospitality and Tourism Technology*, 11(1), pp. 35-48.

Lanham, M., 2018. *Learn Unity ML-Agents : fundamentals of Unity machine learning : incorporate new powerful ML algorithms such as deep reinforcement learning for games*. Birmingham; Mumbai: Packt.

Lanham, M., 2019. *Hands-On Deep Learning for Games*. 1 ed. Birmingham, UK: Packt Publishing Limited.

Librande, S., 2013. *Simulating a City, One Page at a Time*. [Online] Available at: <https://www.gdcvault.com/play/1017708/Simulating-a-City-One-Page> [Accessed 27 April 2024].

Mariana Berga, P. C., 2024. *PyTorch vs TensorFlow: comparing deep learning frameworks*. [Online] Available at: <https://www.imaginarycloud.com/blog/pytorch-vs-tensorflow/> [Accessed 21 March 2024].

Mattar, M. et al., 2020. *Announcing ML-Agents Unity Package v1.0*. [Online] Available at: <https://blog.unity.com/engine-platform/announcing-ml-agents-unity->

package-v1-0

[Accessed 2024 April 27].

McAulay, C., 2019. *What Developers Need to Know about Making a Management/Simulator*. [Online]

Available at: <https://www.gamedeveloper.com/design/what-developers-need-to-know-about-making-a-management-simulator>

[Accessed 30 April 2024].

McCarlie, P. & Hunter, A., 2021. *Using Game AI to Control a Simulated Economy*. Burnaby, Canada, British Columbia Institute of Technology, pp. 629 - 634.

Millington, I. & Funge, J., 2009. *Artificial Intelligence for Games*,. 2nd Edition ed. s.l.:CRC Press.

Montesinos, A., 2022. Profit Prediction based on Financial Statements using Deep Neural Network. *2022 IEEE World AI IoT Congress*, June.pp. 533-537.

ONNX Runtime, n.d. *ONNX Runtime Execution Providers*. [Online]

Available at: <https://onnxruntime.ai/docs/execution-providers/>

[Accessed 03 November 2023].

OpenTTD, n.d. *OpenTTD: About*. [Online]

Available at: <https://www.openttd.org/about>

[Accessed 30 April 2024].

Paradox Interactive, n.d. *Cities Skylines II*. [Online]

Available at: <https://www.paradoxinteractive.com/games/cities-skylines-ii/about>

[Accessed 30 April 2024].

Robbins, M., 2013. Using Neural Networks to Control Agent Threat Response. In: S. Rabin, ed. *Game AI Pro: Collected Wisdom of Game AI Professionals*. 1st Edition ed. Boca Raton, FL, USA: A K Peters/CRC Press.

Samaya Madhavan, S. A. V. R. A. J., 2021. *Compare deep learning frameworks*. [Online]

Available at: <https://developer.ibm.com/articles/compare-deep-learning-frameworks/>

[Accessed 24 March 2024].

Sayantini, 2023. *Keras vs TensorFlow vs PyTorch : Comparison of the Deep Learning Frameworks*. [Online]

Available at: <https://www.edureka.co/blog/keras-vs-tensorflow-vs-pytorch/>

[Accessed 21 March 2024].

Schade, A., 2022. *5.0.3 Hotfix Released*. [Online]

Available at: <https://forums.unrealengine.com/t/5-0-3-hotfix-released/603149/1>

[Accessed 26 March 2024].

Sehrawat, A. & Raj, G., 2018. *Intelligent PC Games: Comparison of Neural Network Based AI against Pre-Scripted AI*. Paris, France, IEEE.

Semler, S., 2014. *Designing Business Simulation Games*. [Online]

Available at: <https://www.linkedin.com/pulse/20141115222121-2508500-designing-business-simulation-games>

[Accessed 28 April 2024].

Shuying Li, Y. Z. W. C. C. J. M. E. S. N. M. M. Q. Z., 2020. Adaptive Boosting (AdaBoost)-based multiwavelength spatial frequency domain imaging and characterization for ex vivo human colorectal tissue assessment. *Journal of biophotonics*, Volume 13.

Siddiq, S., 2024. *10 Best Streaming Service Stocks To Buy*. [Online] Available at: <https://uk.finance.yahoo.com/news/10-best-streaming-stocks-buy-221112460.html> [Accessed 26 April 2024].

Sitzmann, T., 2011. A meta-analytic examination of the instructional effectiveness of computer-based simulation games. *Personnel Psychology*, May, 64(2), pp. 489-528.

Thompson, T., 2021. *How AI upscaling can help remaster game art*. [Online] Available at: <https://www.gamedeveloper.com/art/how-ai-upscaling-can-help-remaster-game-art> [Accessed 29 April 2024].

Unity, 2023. *Barracuda 3.0.1 Manual*. [Online] Available at: <https://docs.unity3d.com/Packages/com.unity.barracuda@3.0/manual/index.html> [Accessed 26 March 2024].

Unity, 2024. *2024 Unity Gaming Report*. [Online] Available at: <https://unity.com/resources/gaming-report> [Accessed 30 April 2024].

Unreal Engine, 2022. *NeuralNetworkInference*. [Online] Available at: <https://docs.unrealengine.com/5.0/en-US/API/Plugins/NeuralNetworkInference/> [Accessed 19 December 2023].

Unreal Engine, 2023. *Unreal Engine 5.2 Release Notes*. [Online] Available at: <https://docs.unrealengine.com/5.2/en-US/unreal-engine-5.2-release-notes/> [Accessed 29 January 2024].

Vandeput, N., 2019. *Forecast KPIs: RMSE, MAE, MAPE & Bias*. [Online] Available at: <https://towardsdatascience.com/forecast-kpi-rmse-mae-mape-bias-cdc5703d242d> [Accessed 06 February 2024].

Vijh, M., Chandola, D., Tikkiwal, V. A. & Kumar, A., 2020. Stock Closing Price Prediction using Machine Learning Techniques. *Procedia Computer Science*. *Procedia Computer Science*, 167(4), pp. :99-606.

Viki, T., 2023. *Netflix is winding down its DVD-by-mail service for good*. [Online] Available at: <https://www.strategyzer.com/library/netflix-is-winding-down-its-dvd-by-mail-service-for-good> [Accessed 26 April 2024].

Wadawadagi, V., 2023. *TensorFlow vs PyTorch: Deep Learning Frameworks [2024]*. [Online] Available at: <https://www.knowledgehut.com/blog/data-science/pytorch-vs-tensorflow> [Accessed 23 March 2024].

Walsh, D., 2023. *The legal issues presented by generative AI*. [Online] Available at: <https://mitsloan.mit.edu/ideas-made-to-matter/legal-issues-presented-generative-ai> [Accessed 23 April 2024].

Wijman, T., 2024. *What are 2023's top game genres?*. [Online] Available at: <https://newzoo.com/resources/blog/top-game-genres-2023> [Accessed 24 January 2024].

Wube Software, n.d. *Factorio*. [Online] Available at: <https://www.factorio.com/> [Accessed 30 April 2024].

Yathish, V., 2022. *Loss Functions and Their Use In Neural Networks*. [Online] Available at: <https://towardsdatascience.com/loss-functions-and-their-use-in-neural-networks-a470e703f1e9> [Accessed 06 March 2024].

Yee, N., 2016. *Revisiting the Strategy Genre Map: Age, Audience Homogeneity, and the Lasso Effect*. [Online] Available at: <https://quanticfoundry.com/2016/03/23/revisiting-the-strategy-genre-map/> [Accessed 03 February 2024].

Zubek, R., 2015. Production Rules Implementation in 1849. In: S. Rabin, ed. *Game AI Pro 2: Collected Wisdom of Game AI Professionals*. Boca Raton, FL, USA: CRC Press.

Zukowski, C., 2022. *What genres are popular on Steam in 2022*. [Online] Available at: <https://howtomarketagame.com/2022/04/18/what-genres-are-popular-on-steam-in-2022/> [Accessed 30 April 2024].

Image references

[1] *Introduction to Deep Learning Models in Unreal* [Screenshot]. (2022). YouTube. Available at: URL <https://youtu.be/7aJQIOe1QTA?si=AZ-EcsbNhFD5CoCi> (Accessed 27 April 2024).

[2] Profit Prediction based on Financial Statements using Deep Neural Network. (2022). *IEEE Explore*. Available at: URL <https://ieeexplore.ieee.org/document/9817172> (Accessed 29 April 2024).

[3] SimCity - Increase Land Value Quickly and Easily [Screenshot]. (2013). *YouTube*. Available at: URL <https://youtu.be/nv99SstRGdl?si=AQK64LU84UrLiT-g> (Accessed 29 April 2024).

[4] Simulating a City, One Page at a Time [Screenshot]. (2013). *Stonetrnix*. Available at: URL <http://stonetrnix.com/gdc-2013/> (Accessed 29 April 2024).

[5] OpenAI Five – A team of 5 Algorithms is Beating Human Opponents in a Popular Game (2018). *Analytics Vidhya*. Available at: URL

<https://www.analyticsvidhya.com/blog/2018/06/openai-five-a-team-of-5-algorithms-is-beating-human-opponents-in-a-popular-game/> (Accessed 3 November 2024).

[6] Supreme Commander 2 (2010). Steam. Available at: URL https://store.steampowered.com/app/40100/Supreme_Commander_2/ (Accessed 29 April 2024).

[7] Top Rated Management Games (n.d.). Steam. Available at: URL https://store.steampowered.com/tags/en/Management/?flavor=contenthub_toprated (Accessed 30 April 2024).

[8] WHAT GENRES ARE POPULAR ON STEAM IN 2022. (2022). HOW TO MARKET A GAME. Available at: URL <https://howtomarketagame.com/2022/04/18/what-genres-are-popular-on-steam-in-2022/> (Accessed 30 April 2024).

[9] Kairosoft (2022) Game Dev Story [Video game]. Kairosoft. Available at: URL https://store.steampowered.com/app/1847240/Game_Dev_Story/ (Accessed: 30 April 2024).

[10] Greenheart Games (2012) Game Dev Tycoon [Video game]. Greenheart Games. Available at: URL https://store.steampowered.com/app/239820/Game_Dev_Tycoon/ (Accessed: 30 April 2024).

[11] OpenTTD Team (2004) OpenTTD [Video game]. OpenTTD Team. Available at: URL <https://store.steampowered.com/app/1536610/OpenTTD/> (Accessed: 30 April 2024).

[12] Cities: Skylines II [Screenshot]. (2022). Steam. Available at: URL https://store.steampowered.com/app/949230/Cities_Skylines_II/ (Accessed 30 April 2024).

[13] Factorio [Screenshot]. (2022). Factorio. Available at: URL <https://www.factorio.com/game/screenshots> (Accessed 30 April 2024).

[14] Netflix. (2014). Google Play Store. Available at: URL https://play.google.com/store/apps/details?id=com.netflix.mediaclient&hl=en_GB&gl=US (Accessed 23 April 2024).

[15] Rome Total War, out now on iOS & Android. (2018). Feral Interactive. Available at: URL <https://www.feralinteractive.com/en/mobile-games/rometw/help/#advisor> (Accessed 28 April 2024).

11. Bibliography

AARKI, 2020. *Role Playing and Strategy Games: User Demographics*. [Online] Available at: <https://www.aarki.com/insights/role-playing-and-strategy-games-user-demographics> [Accessed 03 February 2024].

Adams, E., 2013. *Fundamentals of construction and simulation game design*. 1st ed. s.l.:New Riders.

Alpha Vantage, n.d. *Alpha Vantage*. [Online] Available at: <https://www.alphavantage.co/> [Accessed 20 January 2024].

Aroussi, R., 2024. *yfinance*. [Online] Available at: <https://github.com/ranaroussi/yfinance> [Accessed 20 January 2024].

Ashton, H., 2020. *What does an Economy Designer in games do? Interview with Harry Ashton, Neon Play* [Interview] (7 October 2020).

Batchelor, J., 2023. *Has Unity repaired the damage done by its Runtime Fee plans?*. [Online] Available at: <https://www.gamesindustry.biz/has-unity-repaired-the-damage-done-by-its-runtime-fee-plans> [Accessed 15 February 2024].

Bobbitt, Z., 2021. *MSE vs. RMSE: Which Metric Should You Use?*. [Online] Available at: <https://www.statology.org/mse-vs-rmse/> [Accessed 06 February 2024].

Bourg, D. M. & Seemann, G., 2004. *AI for Game Developers*. 1st Edition ed. Sebastopol, CA, USA: O'Reilly Media, Inc..

Breviu, C., 2022. *Introduction to Deep Learning Models in Unreal*. [Online] Available at: <https://youtu.be/7aJQIOe1QTA?si=GsaetRAuEDZeUo0J> [Accessed 2023 December 18].

Brownlee, J., 2020. *Time Series Forecasting Performance Measures With Python*. [Online] Available at: <https://machinelearningmastery.com/time-series-forecasting-performance-measures-with-python/> [Accessed 05 February 2024].

Brownlee, J., 2021. *How to Choose an Activation Function for Deep Learning*. [Online] Available at: <https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/> [Accessed 01 February 2024].

Brown, M., 2022. *How Video Game Economies are Designed*. [Online] Available at: https://youtu.be/Zrf1cou_yVo?si=WezMlqpq53IEDF8v [Accessed 07 November 2023].

Buffett, M. & Clark, D., 2008. *Warren Buffett and the Interpretation of Financial Statements: The Search for the Company with a Durable Competitive Advantage*. New York: Scribner.

Chugh, V., 2022. *Become UNSTOPPABLE with data*. [Online] Available at: <https://www.kdnuggets.com/2022/12/tuning-adam-optimizer-parameters-pytorch.html> [Accessed 1 January 2024].

Daigle, K. & Staff, G., 2023. *Octoverse: The state of open source and rise of AI in 2023*. [Online] Available at: <https://github.blog/2023-11-08-the-state-of-open-source-and-ai/> [Accessed 23 April 2024].

Daisuke, M., Yuhei, M. & Perez, C., 2017. Forecasting Firm Performance with Machine Learning. *RIETI Discussion Paper Series 17-E-068*, May.

Daniel de Almeida Rocha, J. C. D., 2019. *Simulating Human Behaviour in Games using Machine Learning*. Rio de Janeiro, SBGames. Simpósio Brasileiro de Jogos e Entretenimento Digital..

Dar, P., 2018. *OpenAI Five – A team of 5 Algorithms is Beating Human Opponents in a Popular Game*. [Online] Available at: <https://www.analyticsvidhya.com/blog/2018/06/openai-five-a-team-of-5-algorithms-is-beating-human-opponents-in-a-popular-game/> [Accessed 03 November 2023].

DeepAi, n.d. *Feed Forward Neural Network*. [Online] Available at: <https://deepai.org/machine-learning-glossary-and-terms/feed-forward-neural-network> [Accessed 20 April 2024].

Durrani, A., 2024. *Top Streaming Statistics In 2024*. [Online] Available at: <https://www.forbes.com/home-improvement/internet/streaming-stats/> [Accessed 03 February 2024].

Engelbrecht, D., 2023. *Introduction to Unity ML-Agents : understand the interplay of neural networks and simulation space using the Unity ML-Agents Package*. New York: Apress L. P..

Epic Games, 2022. *Unreal Engine 5.0 Release Notes*. [Online] Available at: [https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5.0-release-notes?application version=5.0](https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5.0-release-notes?application%20version=5.0) [Accessed 26 March 2024].

Epic Games, 2023. *Unreal Engine 5.2 Release Notes*. [Online] Available at: <https://docs.unrealengine.com/5.2/en-US/unreal-engine-5.2-release-notes/> [Accessed 29 January 2024].

Extra Credits, 2016. *MMO Economies - Hyperinflation, Reserve Currencies & You! - Extra Credits*. [Online] Available at: <https://youtu.be/sumZLwFXJqE?si=P8-payAp1oCV9dJ0> [Accessed 27 April 2024].

Fangasadha, E. F., Soeroredjo, S., Anderies & Gunawan, A. A. S., 2022. *Literature Review of OpenAI Five's Mechanisms in Dota 2's Bot Player*. Semarang, Indonesia,, IEEE, pp. 183-190.

Frames, W., 2022. *Bringing Deep Learning to UE5 — Pt. 2*. [Online] Available at: <https://medium.com/@weirdframes/bringing-deep-learning-to-unreal-engine-5-pt-2-51c1a2a2c3> [Accessed 20 December 2023].

Gaikwad, A. et al., 2023. *Profit Prediction for Businesses using Machine Learning Algorithms*. Coimbatore, India, IEEE.

GDC, Game Developer, 2024. *2024 State of the Industry*. San Francisco, CA, GDC, Game Developer.

Gilmore, J. L., 2020. <https://xyretail.medium.com/unprofitable-unicorns-3170d8c682e1>. [Online] Available at: <https://xyretail.medium.com/unprofitable-unicorns-3170d8c682e1> [Accessed 03 November 2023].

Greenheart Games, 2012. *Greenheart Games*. [Online] Available at: <https://www.greenheartgames.com/app/game-dev-tycoon/> [Accessed 30 April 2024].

Hall, M., 2018. *Choosing A Project Management Tool For Game Development*. [Online] Available at: <https://www.gamedeveloper.com/business/choosing-a-project-management-tool-for-game-development> [Accessed 18 April 2024].

Hayes, A., 2021. *SEC Filings: Forms You Need To Know*. [Online] Available at: <https://www.investopedia.com/articles/fundamental-analysis/08/sec-forms.asp> [Accessed 20 April 2024].

Into Games, 2024. *Entry Level Skills Hub - Database - Junior Programmer*. [Online] Available at: <https://entrylevel.games/database/junior-programmer> [Accessed 26 March 2024].

Jeff Loucks, C. A. K. W. J. A., 2024. *Driven to tiers: Streaming video services look to up their profitability game with viewers*. [Online] Available at: <https://www2.deloitte.com/uk/en/insights/industry/technology/technology-media-and-telecom-predictions/2024/tmt-predictions-streaming-video-services-profitability-must-increase-in-2024.html> [Accessed 29 January 2024].

Jones, S., 2024. *AI and machine learning specialists to become the fastest growing jobs in the next four years*. [Online] Available at: <https://londonlovestech.com/ai-and-machine-learning-specialists-to-become-the-fastest-growing-jobs-in-the-next-four-years/> [Accessed 23 April 2024].

Kairosoft Co., Ltd, n.d. *Game Dev Story*. [Online] Available at: <https://kairosoft.net/game/appli/gamedev.html> [Accessed 30 April 2024].

Kelly, M., 2021. Game Genres. In: *Game Design & Development 2021*. Dublin, Ireland: Technological University Dublin.

Koeffel Christina, H. W. L. J. H. M. G. A. T. M., 2010. Using Heuristics to Evaluate the Overall User Experience of Video Games and Advanced Interaction Games. In: R. Bernhaupt, ed. *Evaluating User Experience in Games*. s.l.:Springer, pp. 233-256.

Kraj, N., 2021. *Keys to Economic Systems*. [Online] Available at: <https://gdkeys.com/keys-to-economic-systems/> [Accessed 20 April 2024].

Lado-Sestayo, R. & Vivel-Búa, M., 2020. Hotel profitability: a multilayer neural network approach. *Journal of Hospitality and Tourism Technology*, 11(1), pp. 35-48.

Lanham, M., 2018. *Learn Unity ML-Agents : fundamentals of Unity machine learning : incorporate new powerful ML algorithms such as deep reinforcement learning for games*. Birmingham; Mumbai: Packt.

Lanham, M., 2019. *Hands-On Deep Learning for Games*. 1 ed. Birmingham, UK: Packt Publishing Limited.

Librande, S., 2013. *Simulating a City, One Page at a Time*. [Online] Available at: <https://www.gdcvault.com/play/1017708/Simulating-a-City-One-Page> [Accessed 27 April 2024].

Mariana Berga, P. C., 2024. *PyTorch vs TensorFlow: comparing deep learning frameworks*. [Online] Available at: <https://www.imaginarycloud.com/blog/pytorch-vs-tensorflow/> [Accessed 21 March 2024].

Mattar, M. et al., 2020. *Announcing ML-Agents Unity Package v1.0*. [Online] Available at: <https://blog.unity.com/engine-platform/announcing-ml-agents-unity-package-v1-0> [Accessed 2024 April 27].

McAulay, C., 2019. *What Developers Need to Know about Making a Management/Simulator*. [Online] Available at: <https://www.gamedeveloper.com/design/what-developers-need-to-know-about-making-a-management-simulator> [Accessed 30 April 2024].

McCarlie, P. & Hunter, A., 2021. *Using Game AI to Control a Simulated Economy*. Burnaby, Canada, British Columbia Institute of Technology, pp. 629 - 634.

Millington, I. & Funge, J., 2009. *Artificial Intelligence for Games*,. 2nd Edition ed. s.l.:CRC Press.

Montesinos, A., 2022. Profit Prediction based on Financial Statements using Deep Neural Network. *2022 IEEE World AI IoT Congress*, June, pp. 533-537.

ONNX Runtime, n.d. *ONNX Runtime Execution Providers*. [Online] Available at: <https://onnxruntime.ai/docs/execution-providers/> [Accessed 03 November 2023].

OpenTTD, n.d. *OpenTTD: About.* [Online]
Available at: <https://www.openttd.org/about>
[Accessed 30 April 2024].

Paradox Interactive, n.d. *Cities Skylines II.* [Online]
Available at: <https://www.paradoxinteractive.com/games/cities-skylines-ii/about>
[Accessed 30 April 2024].

Robbins, M., 2013. Using Neural Networks to Control Agent Threat Response. In: S. Rabin, ed. *Game AI Pro: Collected Wisdom of Game AI Professionals*. 1st Edition ed. Boca Raton, FL, USA: A K Peters/CRC Press.

Samaya Madhavan, S. A. V. R. A. J., 2021. *Compare deep learning frameworks.* [Online]
Available at: <https://developer.ibm.com/articles/compare-deep-learning-frameworks/>
[Accessed 24 March 2024].

Sayantini, 2023. *Keras vs TensorFlow vs PyTorch : Comparison of the Deep Learning Frameworks.* [Online]
Available at: <https://www.edureka.co/blog/keras-vs-tensorflow-vs-pytorch/>
[Accessed 21 March 2024].

Schade, A., 2022. *5.0.3 Hotfix Released.* [Online]
Available at: <https://forums.unrealengine.com/t/5-0-3-hotfix-released/603149/1>
[Accessed 26 March 2024].

Sehrawat, A. & Raj, G., 2018. *Intelligent PC Games: Comparison of Neural Network Based AI against Pre-Scripted AI*. Paris, France, IEEE.

Semler, S., 2014. *Designing Business Simulation Games.* [Online]
Available at: <https://www.linkedin.com/pulse/20141115222121-2508500-designing-business-simulation-games>
[Accessed 28 April 2024].

Shuying Li, Y. Z. W. C. C. J. M. E. S. N. M. M. Q. Z., 2020. Adaptive Boosting (AdaBoost)-based multiwavelength spatial frequency domain imaging and characterization for ex vivo human colorectal tissue assessment. *Journal of biophotonics*, Volume 13.

Siddiq, S., 2024. *10 Best Streaming Service Stocks To Buy.* [Online]
Available at: <https://uk.finance.yahoo.com/news/10-best-streaming-stocks-buy-221112460.html>
[Accessed 26 April 2024].

Singh, R., Mehra, N. & Dhamija, A., 2022. *Natural Selection Simulator using Machine Learning*. Sonapat, India, IEEE.

Sitzmann, T., 2011. A meta-analytic examination of the instructional effectiveness of computer-based simulation games. *Personnel Psychology*, May, 64(2), pp. 489-528.

Terra, J., 2024. *Keras vs Tensorflow vs Pytorch: Key Differences Among Deep Learning.* [Online]
Available at: <https://www.simplilearn.com/keras-vs-tensorflow-vs-pytorch-article>
[Accessed 21 March 2024].

Thompson, T., 2021. *How AI upscaling can help remaster game art*. [Online] Available at: <https://www.gamedeveloper.com/art/how-ai-upscaling-can-help-remaster-game-art> [Accessed 29 April 2024].

Unity, 2023. *Barracuda 3.0.1 Manual*. [Online] Available at: <https://docs.unity3d.com/Packages/com.unity.barracuda@3.0/manual/index.html> [Accessed 26 March 2024].

Unity, 2024. *2024 Unity Gaming Report*. [Online] Available at: <https://unity.com/resources/gaming-report> [Accessed 30 April 2024].

Unreal Engine, 2022. *NeuralNetworkInference*. [Online] Available at: <https://docs.unrealengine.com/5.0/en-US/API/Plugins/NeuralNetworkInference/> [Accessed 19 December 2023].

Unreal Engine, 2023. *Unreal Engine 5.2 Release Notes*. [Online] Available at: <https://docs.unrealengine.com/5.2/en-US/unreal-engine-5.2-release-notes/> [Accessed 29 January 2024].

Vandeput, N., 2019. *Forecast KPIs: RMSE, MAE, MAPE & Bias*. [Online] Available at: <https://towardsdatascience.com/forecast-kpi-rmse-mae-mape-bias-cdc5703d242d> [Accessed 06 February 2024].

Vijh, M., Chandola, D., Tikkiwal, V. A. & Kumar, A., 2020. Stock Closing Price Prediction using Machine Learning Techniques. *Procedia Computer Science*, 167(4), pp. :99-606.

Viki, T., 2023. *Netflix is winding down its DVD-by-mail service for good*. [Online] Available at: <https://www.strategyzer.com/library/netflix-is-winding-down-its-dvd-by-mail-service-for-good> [Accessed 26 April 2024].

Wadawadagi, V., 2023. *TensorFlow vs PyTorch: Deep Learning Frameworks [2024]*. [Online] Available at: <https://www.knowledgehut.com/blog/data-science/pytorch-vs-tensorflow> [Accessed 23 March 2024].

Walsh, D., 2023. *The legal issues presented by generative AI*. [Online] Available at: <https://mitsloan.mit.edu/ideas-made-to-matter/legal-issues-presented-generative-ai> [Accessed 23 April 2024].

Wijman, T., 2024. *What are 2023's top game genres?*. [Online] Available at: <https://newzoo.com/resources/blog/top-game-genres-2023> [Accessed 24 January 2024].

Woei-Jiunn Tsaur, C. H. T. C. C.-L., 2022. Effective Bots' Detection for Online Smartphone Game Using Multilayer Perceptron Neural Networks. *Security and Communication Networks*, Volume 2022.

Wube Software, n.d. *Factorio*. [Online]
Available at: <https://www.factorio.com/>
[Accessed 30 April 2024].

Yathish, V., 2022. *Loss Functions and Their Use In Neural Networks*. [Online]
Available at: <https://towardsdatascience.com/loss-functions-and-their-use-in-neural-networks-a470e703f1e9>
[Accessed 06 March 2024].

Yee, N., 2016. *Revisiting the Strategy Genre Map: Age, Audience Homogeneity, and the Lasso Effect*. [Online]
Available at: <https://quanticfoundry.com/2016/03/23/revisiting-the-strategy-genre-map/>
[Accessed 03 February 2024].

Zubek, R., 2015. Production Rules Implementation in 1849. In: S. Rabin, ed. *Game AI Pro 2: Collected Wisdom of Game AI Professionals*. Boca Raton, FL, USA: CRC Press.

Zukowski, C., 2022. *What genres are popular on Steam in 2022*. [Online]
Available at: <https://howtomarketagame.com/2022/04/18/what-genres-are-popular-on-steam-in-2022/>
[Accessed 30 April 2024].

Appendix 1 – Project Links

GitHub

- Project Repository: https://github.com/chav0028/Unprofitable-Startup_Public

YouTube

- Demo Video: <https://youtu.be/c9blcmceiUw>
- Neural Network Inference Unreal Engine Prototype: <https://youtu.be/niKxpE9DWhI>
- Financial neural network prototype: <https://youtu.be/vrPkeIIAYyU>

Appendix 2 – Elicitation requirements responses

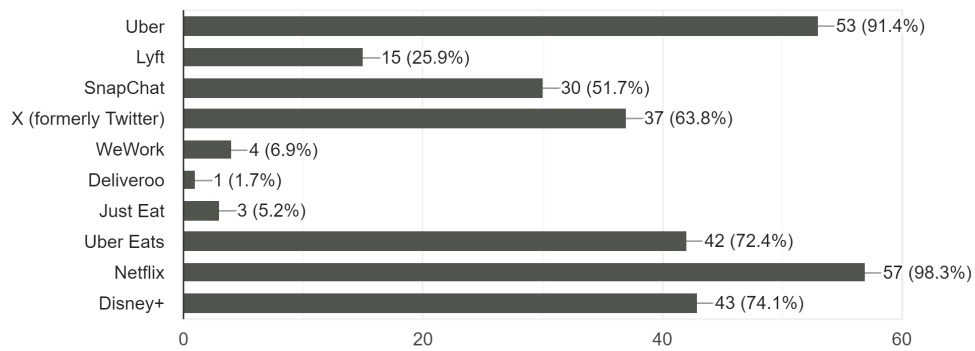
Unprofitable Startup is a business simulation game where you manage an already well-established company, that despite its size and market share, has never managed to make a profit, as is the case with many of today's best-known companies.

As the player, your goal is to make the best financial decisions to make the company profitable. The effect of these decisions will be seen through the result of a machine learning simulation algorithm.

Business Setting

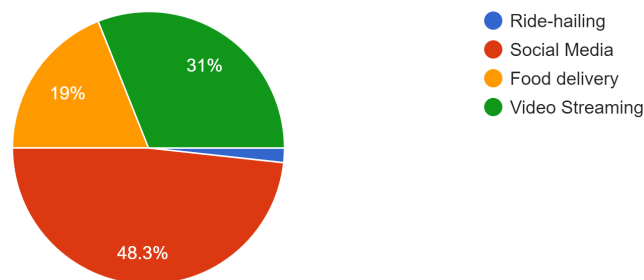
1. With which of these companies/products are you familiar with?

58 responses



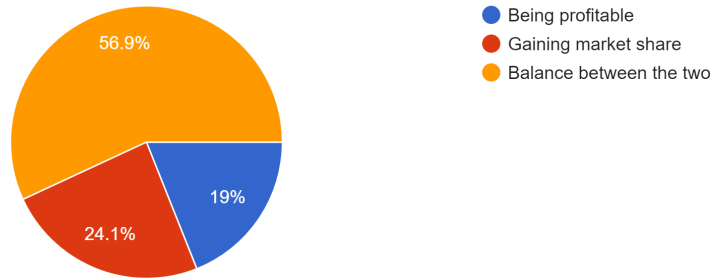
2. Which of this business ventures do you consider to be the most profitable?

58 responses



3. At their beginning, do you think start-up companies should focus on gaining market share or in being profitable?

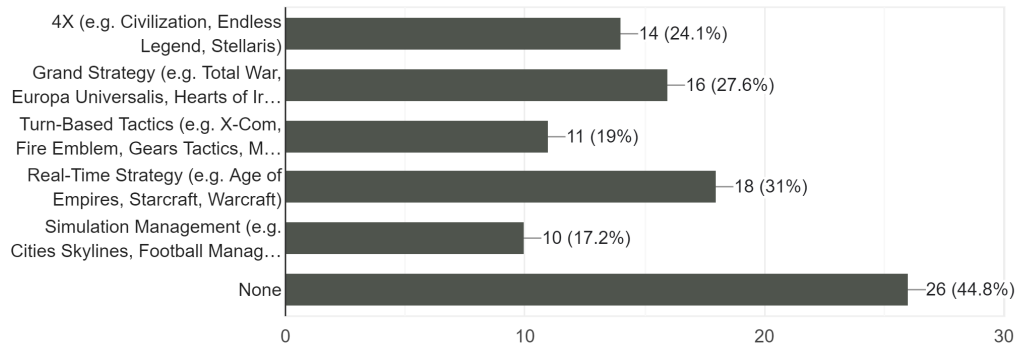
58 responses



Comparable Games

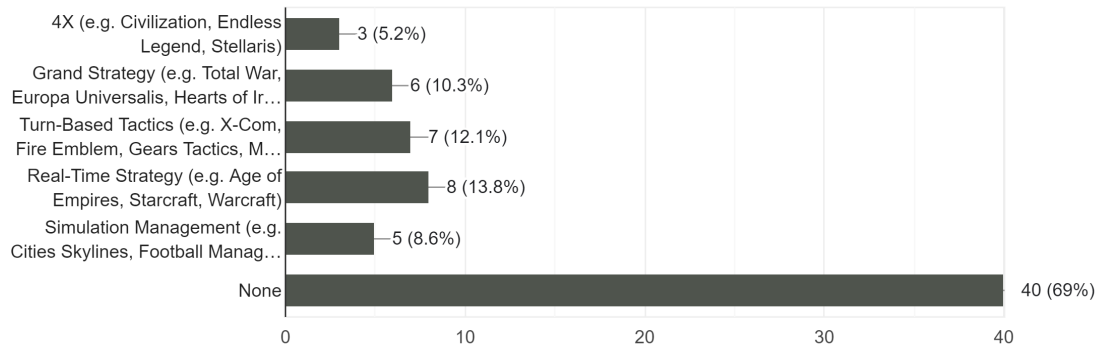
4. With which of the following video game genres are you familiar with?

58 responses



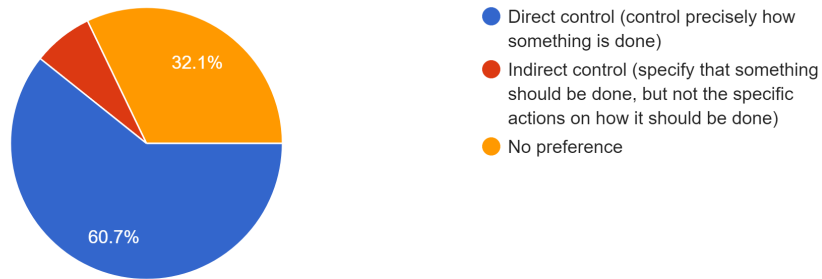
5. Which of the following video game genre have you played in the last 6 months?

58 responses



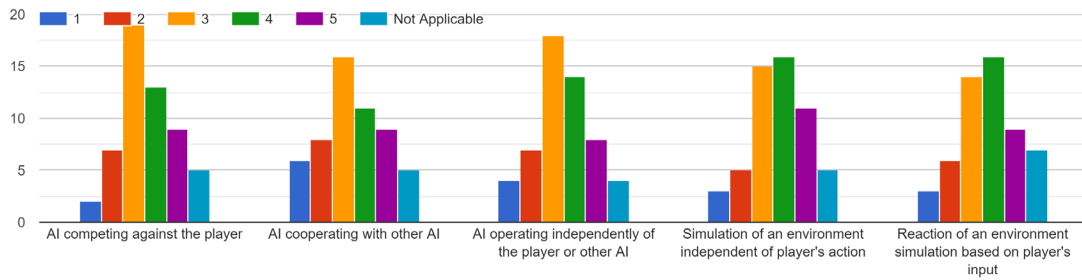
6. When making a strategic choice in a video game do you prefer:

56 responses



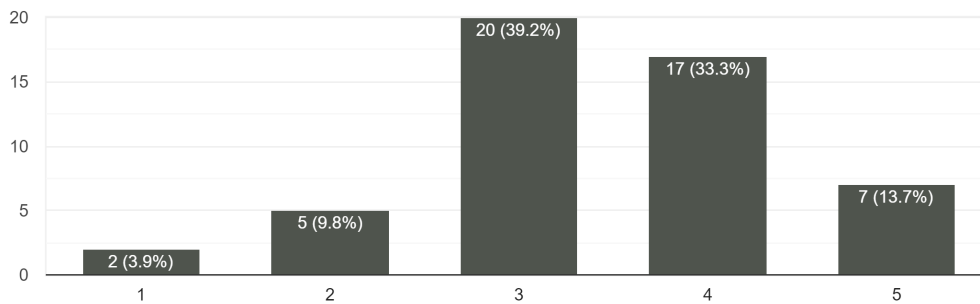
Artificial Intelligence

7. How would you rate the quality of the Artificial Intelligence in the following situations, based on your recent video game experience



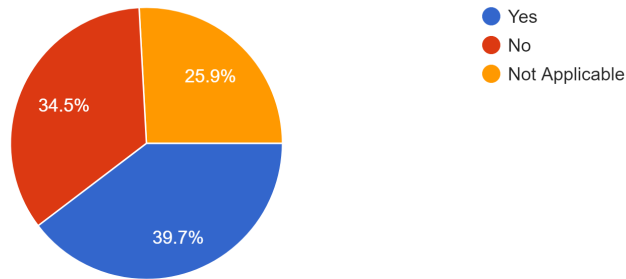
8. Based on your most recent experience, how realistic do you consider games simulate the virtual economy?

51 responses



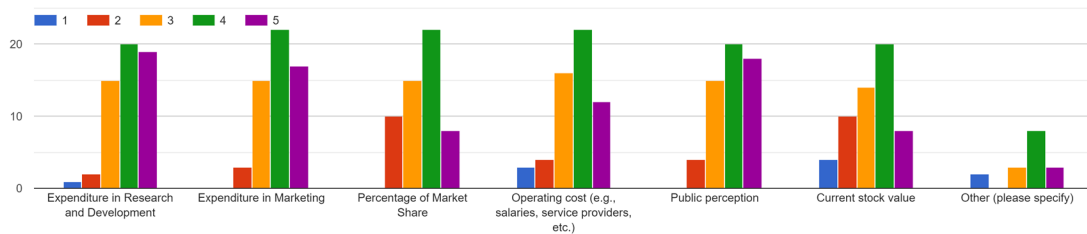
9. Does the simulation of a game's economy affect your immersion in the game?

58 responses



Simulation Inputs

10. Based on your real life knowledge, which factors do you think should have the most effect on a company's profit (financial gain, after any production, buying or operating cost)?



If you selected other, please specify it here.

The value of the brand

Revenues

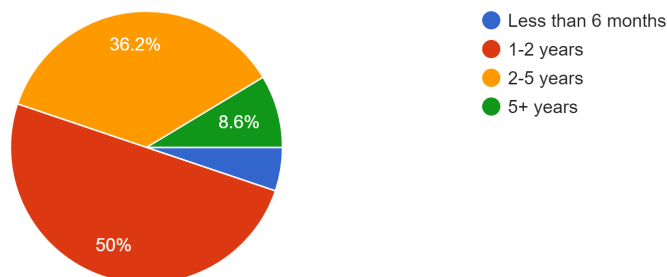
Its employees and training

Renombre de la marca si no es una marca muy conocida no la comprará mucha gente

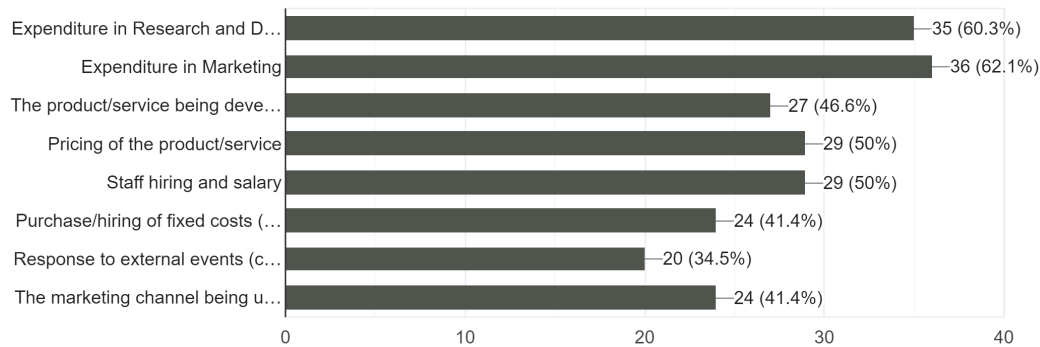
Competence

11. On your opinion, what will be a reasonable time for a start-up company to start making a profit

58 responses



12. If you were managing a company in a game, which factors would you like to control? Supposing that the company is already in a business industry... social media company to a food delivery company)
58 responses



Please leave any additional feedback or comment, regarding the questions in this form or the study, here.7 responses

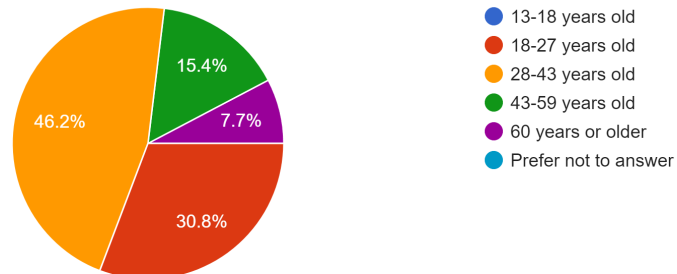
- None
- I think you need to include your target demographics
- It is a New knowledge in a future system facility
- ITS a good idea that proyect, i will buy this kind of video game because sounds very interesting.
- Las respuesta son muy simples y algunas tienen limitaciones, pero todo está bien
- Everything is ok
- One of the most difficult thing to do for an startup to be a float until it has enough relevance to be profitable

Appendix 3 – Playtesting responses

Player Profile

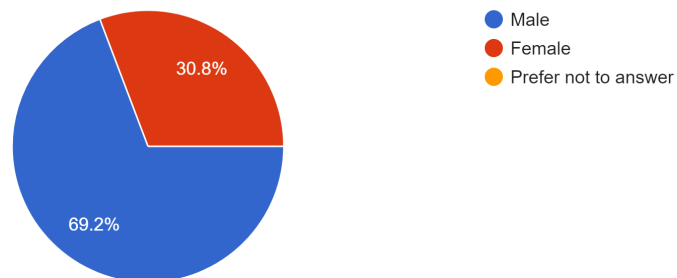
1. What is your age?

13 responses



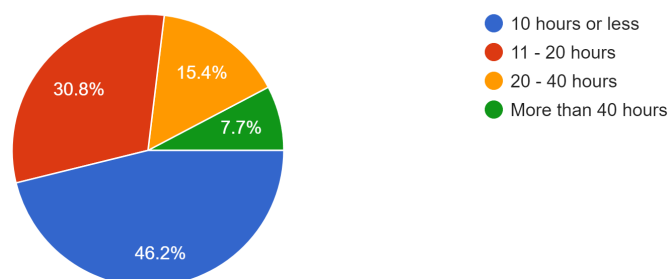
2. What is your gender?

13 responses



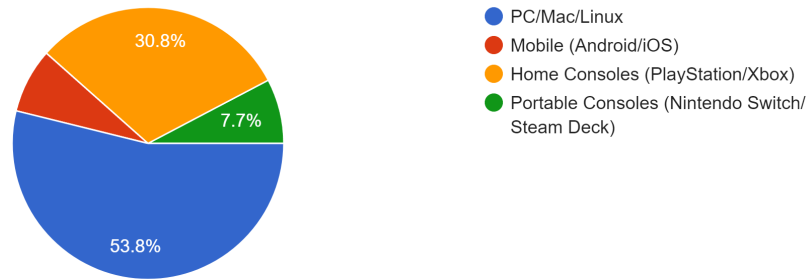
3. How many hours per week do you play videogames

13 responses



4. When you play video games, which is your platform of choice

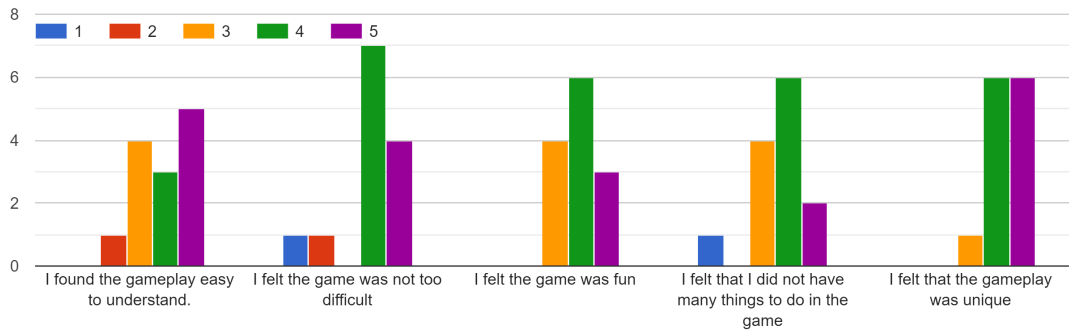
13 responses



Gameplay

Please rate the following from 1, strongly disagree to 5, strongly agree

5. Gameplay



Please feel free to leave any additional feedback about the Gameplay here

responses

Gameplay could be more immersive. The environment, lighting and graphics were excellent.

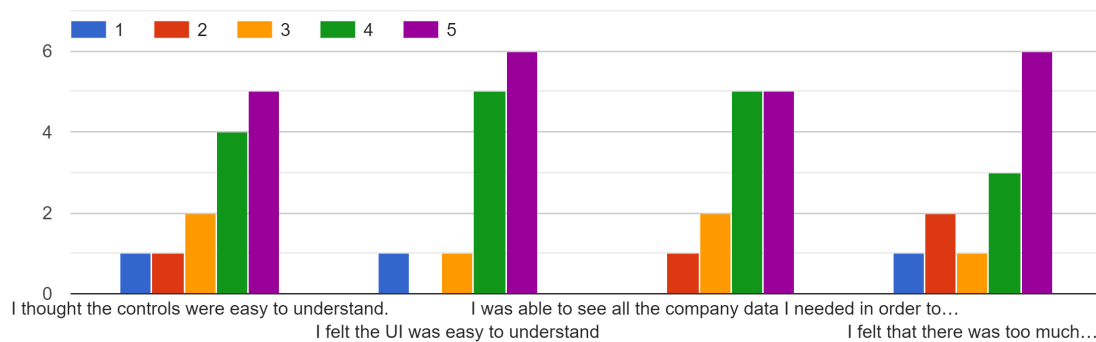
- The initial prompt says "Run of of Cash" - I won in about a few seconds by simply clicking on the "next turn" button. - My CPU was at around 95 degrees. My laptop is old, but the only games that make the CPU go that high are games like Halo MCC on max settings or Dark Souls III on high settings.

Interesting game design.

Controls and UI

Please rate the following from 1, strongly disagree to 5, strongly agree

6. Controls and UI



Please feel free to leave any additional feedback about the Controls and the UI here⁵ responses

Add a simple UI tutorial, just so that I can understand better the game

Controls could have better tutorial messages for easier understanding. Information is very extensive, but could be structured to be more engaging.

- Once my budget became large enough, it was hard putting in the exact value I wanted to spend on Marketing and Development. - While the basics of the gameplay are explained, the nuance behind it never is. I don't know what I should be looking at when deciding what to put my money on. - There should be a small prompt that explains what is what withing the company details window.

* pop-up / alert when an expenditure reaches the next level * graphs that show the trend of the variables at play, not just numbers * pop-up / alert that warn the player is getting close to bank-corrupt (before it's too late) and show some tips on how overcome the crisis

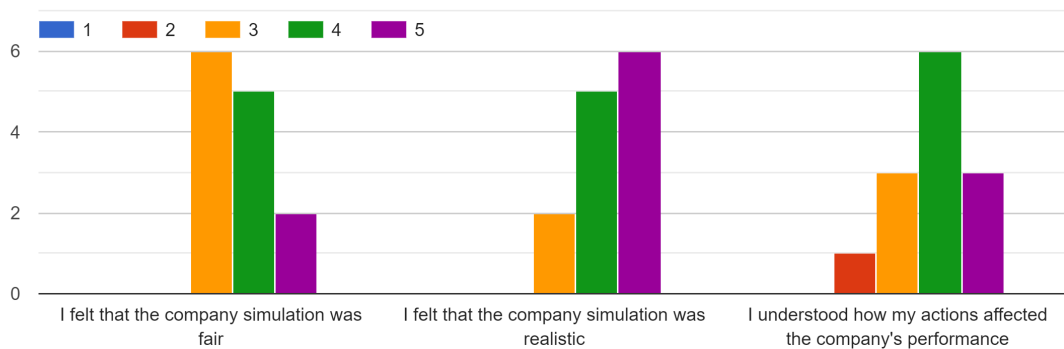
I like that there is a widget to explain things further if I forgot something.

Company Simulation

One of the main features of Unprofitable Startup is the use of a Neural Network to predict the net income a company has, based on its different financial data.

Please rate the following from 1, strongly disagree to 5, strongly agree

7. Company Simulation



Please feel free to leave any additional feedback about the simulation of the company here² responses

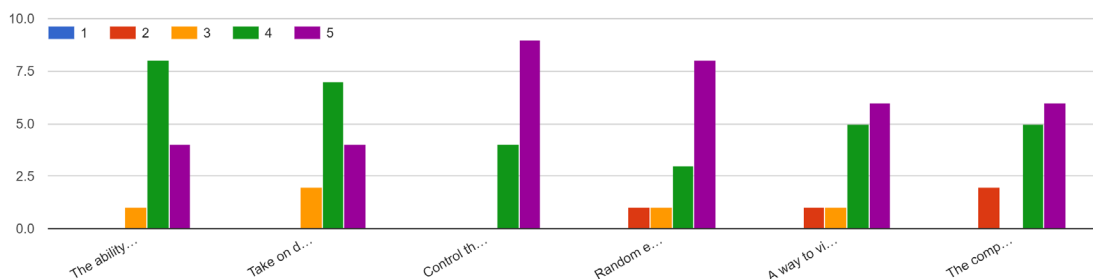
Really nice concept. With additional features, the game could be very engaging and unique. Had trouble understanding the game's objectives, could use engaging tutorial prompts for this. Some side objectives could also improve the experience. Overall, a good alpha version with dynamic lighting and good graphics.

- There's too little information. And whatever information there is on screen I have no idea what it is, as most of it are just labels and numbers. - The company simulation wasn't fair because all I did to win was press "next turn" a few times really fast and I won. - It didn't feel realistic because there was no visual change other than the numbers and the prompt at the end. Not to mention how fast I won.

Future Developments

Please rate the following from 1, not interested to 5, very interested

8. If development of Unprofitable Startup was to continue, which features would you like to see implemented?



1. The ability to develop movie or series project
2. Take on debt to fund business ventures
3. Control the staff and hiring of personnel
4. Random events popping up every certain time, forcing you to make business decisions
5. A way to view my company's performance throughout its history

6. The company office changes depending on your current company's performance

Please feel free to leave any additional feedback about future improvements here^{3 responses}

Add a restart game button. Controlling staff NPCs could be very good.
- I'd like to be able to visit different parts of the building/office as well as SEE the employees that are working for me. - I'd like to have more input in choosing what the company does with its money other than 2 sliders and a next turn button.
Would love to see the future development of Unprofitable Startup

If you found anything that you consider an error, or something that was not working as you expected, in the demo, please comment it here^{4 responses}

No errors.
After a certain point, no matter what I did my profits started to go down. I tried investing many times in different ways even looking at the information I did understand and from what I could gather I was doing everything right but my profits were still going down.
noticed two typos :) * welcome screen: before you run of of cash * how-to-play screen: visualize the in detail the current
I could not find

Please leave any additional feedback or comment regarding the questions in this form or the demo, here.

Once again, thanks for your valuable feedback.

4 responses

Interesting game and idea
Please improve the performance. My CPU has no right to get that hot. Also, please provide options for the graphics.
great work!
Thank you as well - good luck!

Appendix 4 – Versions release notes

After each weekly sprint, a build of the game was made and tested. These release notes highlight the different features and bug fixes implemented for each game version.

Release notes - Unprofitable Startup - V0.1

Fiscal Date Ending (yyyy.mm.dd)
Dec 31, 2023
Operating Cash Flow
1,663,014,016
Capital Expenditures
81,632,000
Gross Profit
3,525,339,904
Total Revenue
8,832,825,344
Cost of Revenue
5,307,485,184
Operating Expenses
3,761,231,104
Stock Value
473.17

Run Model
Net Income Output
1,242,245,760

Epic

US-19 The Neural Network model is stored as an ONNX file.

Task

US-21 Scale the financial data inputted and outputted from the Neural Network

US-22 Create Unreal Engine test scene for the Neural Network

US-23 Create Net Income Inference Neural Network in Unreal Engine

Release notes - Unprofitable Startup - V0.2



Task

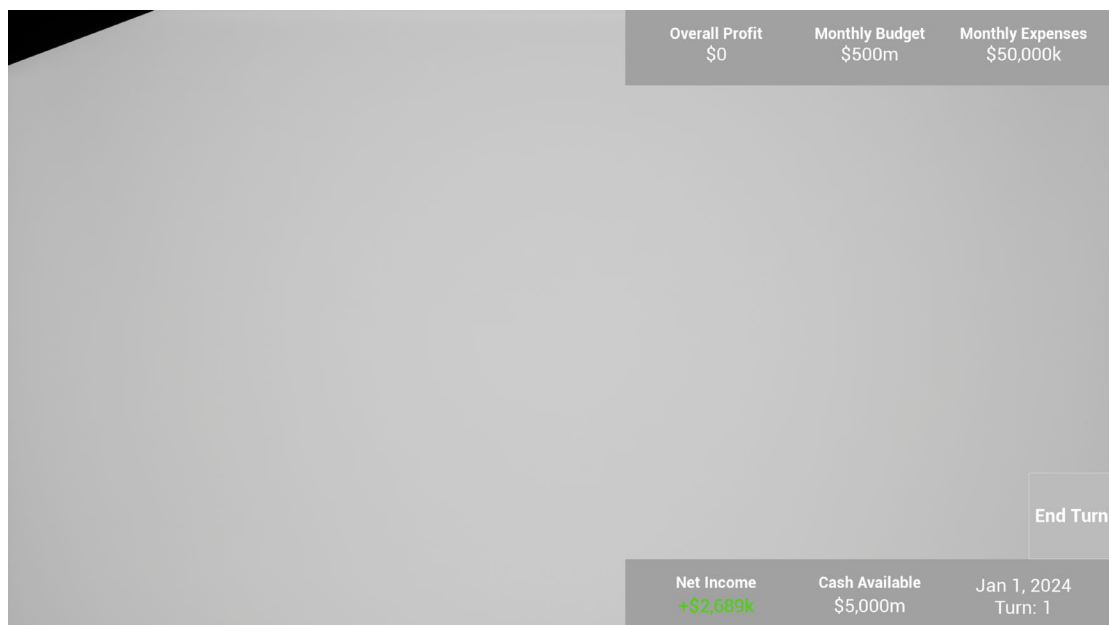
US-24 Create Company object in Unreal Engine

US-25 Create wireframe HUD to visualize the player's company financial data.

US-26 Create button to end the current turn.

US-27 Create core game manager functionality.

Release notes - Unprofitable Startup - V0.3



Task

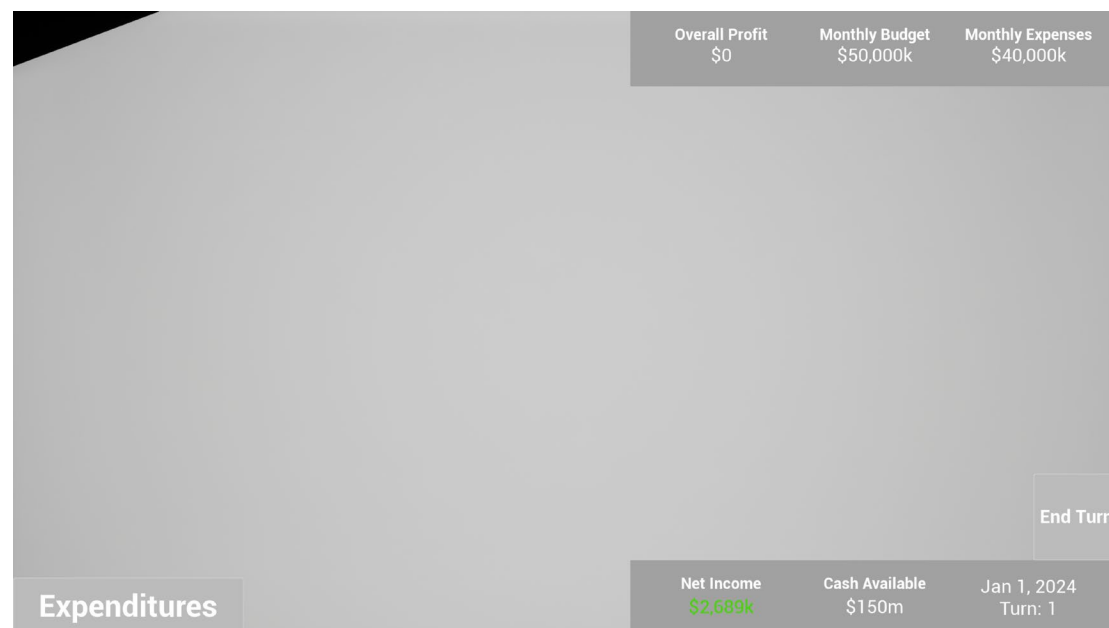
US-35 Round the financial data into easy-to-read values

Bug

US-56 The monthly budget is bigger than the overall cash available.

US-57 Moving the mouse cursor moves the camera.

Release notes - Unprofitable Startup - V0.4



Epic

US-2 Manage budget.

US-3 Turn-based actions.

US-20 Gathering Financial Data

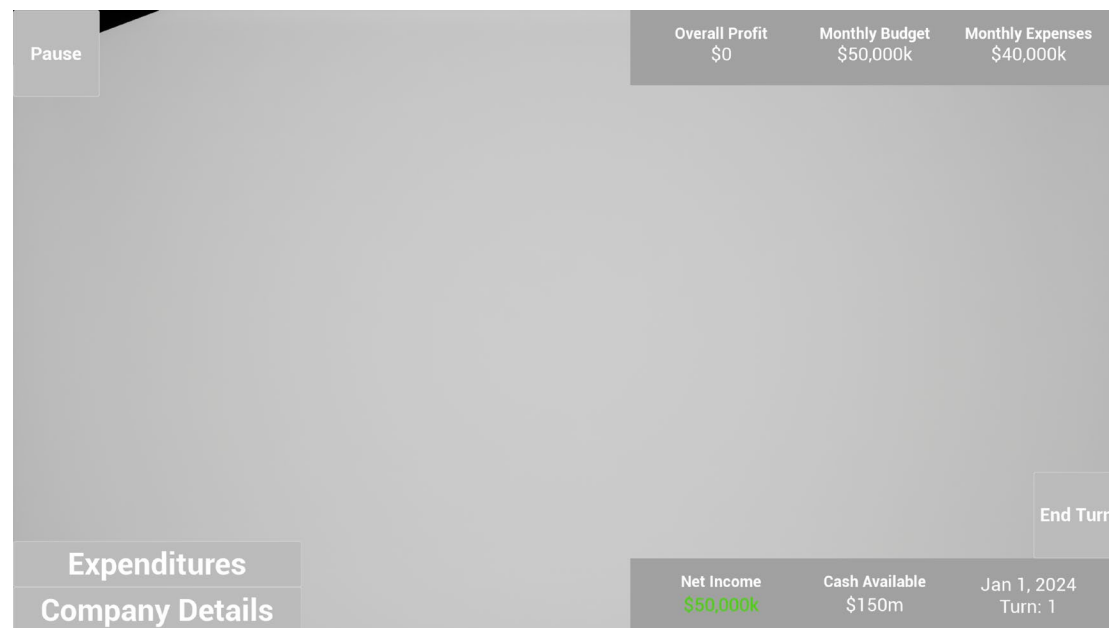
Task

US-28 Interpolate the Neural Network financial data to be weekly.

US-32 Add more up to date financial data.

US-53 Create expenditures screen.

Release notes - Unprofitable Startup - V0.5



Epic

US-1 Net Income simulation based on player's actions.

US-6 Pause Menu

US-15 Accessible financial data

Task

US-40 Add tooltips to the different UI elements explaining what they do.

US-49 Create pause menu screen.

US-54 Cap the amount that the Net Income can change between months.

US-65 Increase the effect of the Marketing and Research expenses in the game.

US-70 Create company details submenu.

Bug

US-64 The expenditures screen slider values do not snap to the step size.

US-71 The budget allocation for each expense type is not updated if the monthly budget is reduced.

Release notes - Unprofitable Startup - V0.6



Epic

US-4 Animated background

US-8 Sounds and music.

US-17 Optimal performance

Task

US-33 Create a way to measure the game's performance.

US-42 Add sound effects for the game UI.

US-43 Add background music to the main game.

US-52 Create 3D animated background for Main Game level.

US-55 Create custom mouse cursor icon.

US-68 Finalise the game UI assets to improve the appearance of the game.

Bug

US-75 The overall monthly expenses is not displayed in the Company Details screen.

US-76 Leftover available funds may not be assignable for a topped-up expense type.

Release notes - Unprofitable Startup - V0.7



Epic

US-18 Neural Network is used for simulations.

Task

US-29 Improve the performance of the Neural Network

US-39 Create a start of the game pop-up explaining the objective of the game.

US-58 Create a lose end game condition.

US-59 Create a win game condition.

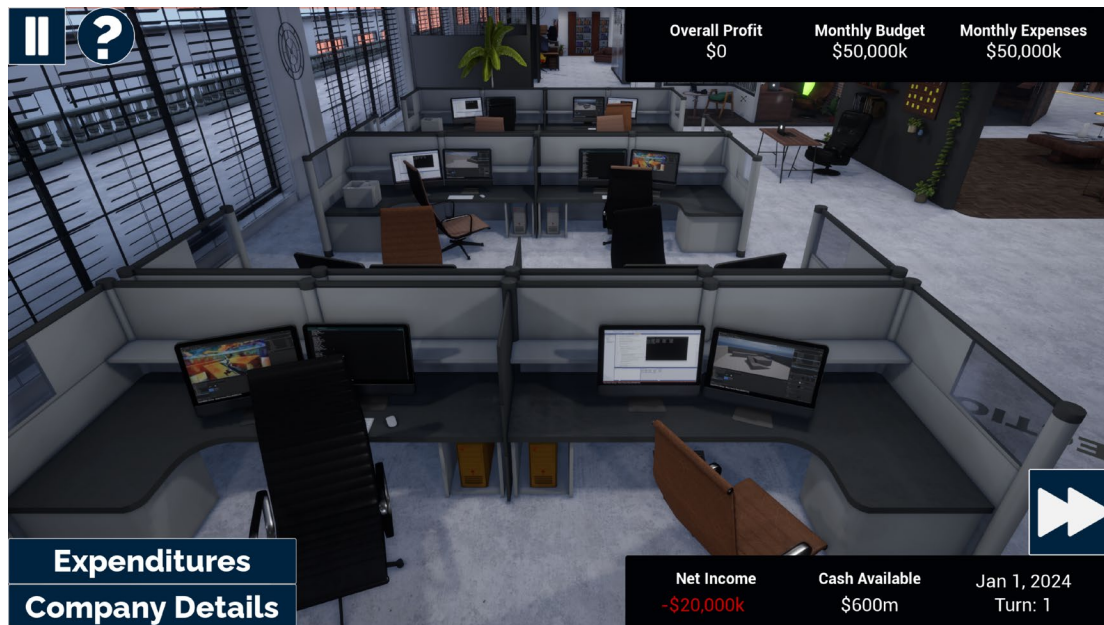
US-72 Improve the game balance.

Bug

US-79 The custom mouse cursor image lags when low framerates

US-80 Mismatch between the monthly budget shown in the company details and the in-game HUD.

Release notes - Unprofitable Startup - V0.8



Epic

US-9 Financial data tooltips and help section

US-16 Intuitive to use

Sub-task

US-37 Create gameplay help section

US-38 Create financial terms dictionary

Task

US-34 Organize usability testing of the game

US-36 Create help submenu

US-45 Create demo video of the game

US-82 Add a visual indicator of your performance in comparison to the previous turn.

US-83 Create a more visual tutorial for the introduction of the game

US-84 Improve the readability of the net income value in relation to the monthly expenses

US-85 Create button to restart the game in the pause menu

Appendix 5 – Versions test cases

Version 0.1

Test

Date:

Wednesday, March 13th, 2024

Test

Version

:

V0.1

Case ID	Description	Expected Result	Pass/ Fail	Actual Result	Comment	Jira Issue
1	All custom-made automated tests, "UnprofitableStartup" category, in Unreal Engine (Tools->Test Automation) are run.	All automation tests in the Unreal Engine project are successful	Pass	All tests passed		
2	The script `NeuralNetworkTraining.py` used to train a Neural Network and export it works correctly	The Neural Network model is exported an ONNX file, and the Neural Network performance evaluation is acceptable	Pass	The model 'netIncome_neural_net.onnx'		
3	The script `NeuralNetworkTraining.py` produces accurate data to scale/unscale the information from Unreal Engine.	The script produces `scaling_data_information.json` which correctly has the min and max values for each column.	Pass	Scaling data is outputted to 'scaling_data_information.json'		
4	The data scaling JSON is exported as an unpackaged JSON in builds. This opposite of most of the game data, which is packaged, preventing it from being parsed.	The data scaling JSON is copied as `Content/Data/scaling_data_information.json` in builds, remaining unpackaged.	Pass	Scaling data file copied to `..\Windows\UnprofitableStartup\Content\Data\`		
5	The Neural Network, in isolation, produces consistent output results when the same input is given. When running the 'Test_NeuralNetwork' level in Unreal Engine, with the default values, it produces a value within a reasonable range of the one produced by the Financial Neural Network Python script, with the last row of data.	Unreal Engine 'Test_NeuralNetwork' level with default value predicts a Net Income output of 1,242,245,760.	Pass		Consistent result produced in build and Editor	

Version 0.2

Test Date:

Wednesday, March 20th, 2024

Test

Version:

V0.2

Case ID	Description	Expected Result	Pass/Fail	Actual Result	Comment	Jira Issue
1	All custom-made automated tests, "UnprofitableStartup" category, in Unreal Engine (Tools->Test Automation) are run.	All automation tests in the Unreal Engine project are successful	Pass			
2	The script `NeuralNetworkTraining.py` used to train a Neural Network and export it works correctly	The Neural Network model is exported as an ONNX file, and the Neural Network performance evaluation is acceptable	Pass			
3	The script `NeuralNetworkTraining.py` produces accurate data to scale/unscale the information from Unreal Engine.	The script produces `scaling_data_information.json` which correctly has the min and max values for each column.	Pass			
4	The data scaling JSON is exported as an unpackaged JSON in builds. This is opposite of most of the game data, which is packaged, preventing it from being parsed.	The data scaling JSON is copied as `Content/Data/scaling_data_information.json` in builds, remaining unpackaged.	Pass			
5	The Neural Network, in isolation, produces consistent output results when the same input is given.	The Neural Network, in the Unreal Engine test scene, produces similar results to the PyTorch Neural Network script	Pass			
6	When the end turn button is pressed the game progresses the next turn	The end turn button is clickable, and refreshes the game HUD with the latest data (next turn number, overall profits, etc.)	Pass			
7	The financial data HUD shows tooltips when it is hovered over	Tooltips for the different UI elements are shown when they are hovered over	Pass			
8	The game HUD tracks the UI correctly	The game HUD accurately tracks the turn number, net income, and other financial data.	Fail	The monthly budget is bigger than the cash available		US-56
9	The net income HUD text changes depending on its value	When the net income is positive, or zero, a '+' is prefixed and the text turns green; when negative a '-' is prefixed and text colour is set to red.	Pass			

Other Bugs

Description	Expected Result	Actual Result	Comment	Jira Issue
When moving the mouse cursor the camera pans around, like if it was a first-person camera, moving alongside the mouse cursor.	The camera is on a fixed position, regardless of mouse cursor movement.	The camera pans around with mouse cursor movements	Likely caused by Unreal Engine setup (e.g. not having a player controller or similar)	US-57

Version 0.3

Test

Date:

Wednesday, March 27th, 2024

Test

Version:

V0.3

Case ID	Description	Expected Result	Pass/ Fail	Actual Result	Comment	Jira Issue
1	All custom-made automated tests, "UnprofitableStartup" category, in Unreal Engine (Tools->Test Automation) are run.	All automation tests in the Unreal Engine project are successful	Pass			
2	The script `NeuralNetworkTraining.py` used to train a Neural Network and export it works correctly	The Neural Network model is exported an ONNX file, and the Neural Network performance evaluation is acceptable	Pass			
3	The script `NeuralNetworkTraining.py` produces accurate data to scale/unscale the information from Unreal Engine.	The script produces `scaling_data_information.json` which correctly has the min and max values for each column.	Pass			
4	The data scaling JSON is exported as an unpackaged JSON in builds. This opposite of most of the game data, which is packaged, preventing it from being parsed.	The data scaling JSON is copied as `Content/Data/scaling_data_information.json` in builds, remaining unpackaged.	Pass			
5	The Neural Network, in isolation, produces consistent output results when the same input is given.	The Neural Network, in the Unreal Engine test scene, produces similar results to the PyTorch Neural Network script	Pass		Python: 1,242,245,562.2878075 Unreal Engine: 1,242,245,760	
6	When the end turn button is pressed the game progresses the next turn	The end turn button is clickable, and refreshes the game Hud with the latest data (next turn number, overall profits, etc.)	Pass			
7	The financial data HUD shows tooltips when it is hovered over	Tooltips for the different UI elements are shown when they are hovered over	Pass			

8	The game HUD tracks the UI correctly	The game HUD accurately tracks the turn number, net income, and other financial data.	Pass
9	The net income HUD text changes depending on its value	When the net income is positive ,or zero, a '+' is prefixed and the text turns green; when negative a '-' is prefixed and text colour is set to red.	Pass
10	In the UI, financial data is shown up as rounded up values	Values are shown rounded up, when above 100 units, for thousands (k), millions (m), and billions (b)	Pass

Version 0.4

Test

Date:

Friday, April 4th, 2024

Test

Version:

V0.4

Case ID	Description	Expected Result	Category	Pass/Fail	Actual Result	Comment	Jira Issue
1	All custom-made automated tests, "UnprofitableStart up" category, in Unreal Engine (Tools->Test Automation) are run.	All automation tests in the Unreal Engine project are successful	Automated Testing	Pass			
2	The script `NeuralNetworkTraining.py` works correctly, training the neural network and exporting it	The Neural Network model is exported an ONNX file, and the Neural Network performance evaluation is acceptable	Neural Network	Pass			
3	The script `NeuralNetworkTraining.py` produces accurate data to scale/unscale the information from Unreal Engine.	The script produces `scaling_data_information.json` which correctly has the min and max values for each column.	Neural Network	Pass			
4	The data scaling JSON is exported as an unpackaged JSON in builds. This opposite of most of the game data, which is packaged, preventing it from being parsed.	The data scaling JSON is copied as `Content/Data/scaling_data_information.json` in builds, remaining unpackaged.	Builds	Pass			
5	The Neural Network, in isolation, produces consistent output results when the same input is given.	The Neural Network, in the Unreal Engine test scene, produces similar results to the PyTorch Neural Network script	Neural Network	Pass		Python: 1,242,245,562. 2878075 Unreal Engine: 1,242,245,760	

6	When the end turn button is pressed the game progresses the next turn	The end turn button is clickable, and refreshes the game Hud with the latest data (next turn number, overall profits, etc.)	Game play	Pass	
7	The financial data HUD shows tooltips when it is hovered over	Tooltips for the different UI elements are shown when they are hovered over	UI	Fail	The close button in the expenditure's submenu does not have a tooltip US-40
8	The game HUD tracks the UI correctly	The game HUD accurately tracks the turn number, net income, and other financial data.	UI	Pass	
9	The net income HUD text changes depending on its value	When the net income is positive, or zero, a '+' is prefixed and the text turns green; when negative a '-' is prefixed and text colour is set to red.	UI	Pass	
10	In the UI, financial data is shown up as rounded up values	Values are shown rounded up, when above 100 units, for thousands (k), millions (m), and billions (b)	UI	Pass	
11	The script "FinancialData-AlphaVantageData.py" correctly filters the data and exports it as quarterly, monthly, and weekly	Running the Python script produces 3 files, with only the reduced data fields, for "quarterlyFinancialData.csv", "monthlyFinancialData.csv", and "weeklyFinancialData.csv"	Financial Data	Pass	
12	The interpolated monthly and weekly financial data matches closely the original quarterly data obtained from AlphaVantage.	When running "FinancialData-AlphaVantageData.py", the value ranges and the graph representation of the quarterly data is close to the monthly and weekly data.	Financial Data	Pass	
13	The interpolated monthly and weekly financial data matches closely the original quarterly data obtained from AlphaVantage.	When running "FinancialData-AlphaVantageData.py", the value ranges and the graph representation of the quarterly data is close to the monthly and weekly data.	Financial Data	Pass	
14	Clicking the expenditures button opens the Expenditures submenu popup screen	When the expenditures button in the main game UI is clicked a new submenu screen, labelled Expenditures, is shown on top of the UI	Game play	Pass	

15	Other UI elements in the main game UI cannot be interacted with when a submenu screen is open.	Open the Expenditures submenu screen, and click the end turn button or the Expenditures button (again), which are visible under the submenu. Observe that nothing happens	UI	Pass
16	Submenu screens are properly hidden when the close button is pressed	Open the Expenditures submenu screen and click the close submenu button on the top right, notice that the submenu is closed/hidden from view and no error shows up.	UI	Pass
17	In the expenditures screen, the Research and Marketing sliders cannot be below 0, or, add it up, cannot go above the current monthly budget.	In the expenditures screen, moving the slider to the left goes down to 0. The sum of both slider values when maxed, does not go above the monthly budget.	UI	Pass
18	The Game HUD is updated when the expenditures screen is closed.	Opening the expenditure screen and changing the budget allocation updates the monthly expenses amount, this change is reflected in the main game HUD when the submenu is closed	UI	Pass
19	When opened, the expenditure screen shows the most up to date values for monthly budget, available funds for the month, and marketing and research expenses	The expenditure screen shows the correct monthly budget amount, available funds (monthly budget - monthly expenses), and marketing and research expenses (sliders correctly placed, and text displaying correct values)	UI	Pass
20	Adjusting the sliders in the expenditures screen changes the company's budget allocation	When the marketing and the research sliders are moved, the budget allocation for that particular expense is updated in the company. These changes are reflected in both the text under the slider, and the currently available funds amount.	Game play	Pass

Other Bugs

Description	Expected Result	Actual Result	Comment	Jira Issue
-------------	-----------------	---------------	---------	------------

<p>The expenditures screen slider values do not snap to the step size. When adjusting the slider value, this should snap, depending on the budget amount, to the nearest million, hundred thousand, ten thousand, or thousand amount.</p>	<p>Moving the expenditure screen sliders adjust the value snapping at certain intervals.</p>	<p>The value of the slider is completely free, allowing the setting of precise dollar values, which can difficult usability.</p>	<p>It seems the Step Size property of the slider is not considered for mouse controls (https://forums.unrealengine.com/t/umg-step-size-in-sliders-is-broken/408721/5)</p>	<p>US-64</p>
---	--	--	---	------------------------------

Version 0.5

Test
Date: **Wednesday, April 10th, 2024**
Test
Version: **V0.5**

Case ID	Description	Expected Result	Category	Pass/Fail	Actual Result	Comment	Jira Issue
1	All custom-made automated tests, "UnprofitableStart up" category, in Unreal Engine (Tools->Test Automation) are run.	All automation tests in the Unreal Engine project are successful	Automated Testing	Pass			
2	The script `NeuralNetworkTraining.py` works correctly, training the neural network and exporting it	The Neural Network model is exported an ONNX file, and the Neural Network performance evaluation is acceptable	Neural Network	Pass			
3	The script `NeuralNetworkTraining.py` produces accurate data to scale/unscale the information from Unreal Engine.	The script produces `scaling_data_information.json` which correctly has the min and max values for each column.	Neural Network	Pass			
4	The data scaling JSON is exported as an unpackaged JSON in builds. This opposite of most of the game data, which is packaged, preventing it from being parsed.	The data scaling JSON is copied as `Content/Data/scaling_data_information.json` in builds, remaining unpackaged.	Builds	Pass			
5	The Neural Network, in isolation, produces consistent output results when the	The Neural Network, in the Unreal Engine test scene, produces similar results to the PyTorch Neural Network script	Neural Network	Pass		Python: 1,242,245,562.2878075 Unreal Engine: 1,242,245,760	

	same input is given.				
6	When the end turn button is pressed the game progresses the next turn	The end turn button is clickable, and refreshes the game Hud with the latest data (next turn number, overall profits, etc.)	Game play	Pass	
7	The financial data HUD shows tooltips when it is hovered over	Tooltips for the different UI elements are shown when they are hovered over	UI	Pass	
8	The game HUD tracks the UI correctly	The game HUD accurately tracks the turn number, net income, and other financial data.	UI	Pass	
9	The net income HUD text changes depending on its value	When the net income is positive ,or zero, the text turns green; when negative a '-' is prefixed and text colour is set to red.	UI	Pass	
10	In the UI, financial data is shown up as rounded up values	Values are shown rounded up, when above 100 units, for thousands (k), millions (m), and billions (b)	UI	Pass	
11	The script "FinancialData-AlphaVantageData.py" correctly filters the data and exports it as quarterly, monthly, and weekly	Running the Python script produces 3 files, with only the reduced data fields, for "quarterlyFinancialData.csv" , "monthlyFinancialData.csv", and "weeklyFinancialData.csv"	Financial Data	Pass	
12	The interpolated monthly and weekly financial data matches closely the original quarterly data obtained from AlphaVantage.	When running "FinancialData-AlphaVantageData.py", the value ranges and the graph representation of the quarterly data is close to the monthly and weekly data.	Financial Data	Pass	
13	Clicking the expenditures button opens the Expenditures submenu popup screen	When the expenditures button in the main game UI is clicked a new submenu screen, labelled Expenditures, is shown on top of the UI	Game play	Pass	
14	Other UI elements in the main game UI cannot be interacted with when a submenu screen is open.	Open the Expenditures submenu screen, and click the end turn button or the Expenditures button (again), which are visible under the submenu. Observe that nothing happens	UI	Pass	

15	Submenu screens are properly hidden when the close button is pressed	Open the Expenditures submenu screen and click the close submenu button on the top right, notice that the submenu is closed/hidden from view and no error shows up.	UI	Pass
16	In the expenditures screen, the Research and Marketing sliders cannot be below 0, or, add it up, cannot go above the current monthly budget.	In the expenditures screen, moving the slider to the left goes down to 0. The sum of both slider values when maxed, does not go above the monthly budget.	UI	Pass
17	The Game HUD is updated when the expenditures screen is closed.	Opening the expenditure screen and changing the budget allocation updates the monthly expenses amount, this change is reflected in the main game HUD when the submenu is closed	UI	Pass
18	When opened, the expenditure screen shows the most up to date values for monthly budget, available funds for the month, and marketing and research expenses	The expenditure screen shows the correct monthly budget amount, available funds (monthly budget - monthly expenses), and marketing and research expenses (sliders correctly placed, and text displaying correct values)	UI	Pass
19	Adjusting the sliders in the expenditures screen changes the company's budget allocation	When the marketing and the research sliders are moved, the budget allocation for that particular expense is updated in the company. These changes are reflected in both the text under the slider, and the currently available funds amount.	Game play	Pass
20	The expenditure screen sliders snap their value per the nearest rounded up value, depending on the current budget amount.	When moving the expenses sliders in the Expenditures screen, this snap in values to the corresponding step size of million, hundred thousand, etc. This step size on the current monthly budget.	UI	Pass

21	The expenditures allocations are rebalanced, proportionally, if the new budget (after the end of a turn) has been reduced and is no longer enough to cover the allocated expenses	Having the budget fully allocated, ending a turn that has a reduced budget, causes the expenditures to be re-allocated proportionally per the new budget. These uses rounded down step size values and may not cover the 100% of expenses.	Game play	Pass	
22	Clicking the Company Details button opens the Company Details submenu popup screen	When the Company Details button in the main game UI is clicked a new submenu screen, labelled Company Details, is shown on top of the UI	Game play	Pass	
23	The company details screen shows the correct and up to date information when opened.	When the Company Details screen is opened, this shows the latest up to date information. This has the currency data shown as rounded up in thousands (k), millions (m) or billions (b) when appropriate.	UI	Fail	There is no Monthly Expense field in Company Details, only the broken-down expenses for Marketing and Research US-75
24	In the Company Details screen when at the max investment level, display the funding required for the next level as ` `	Open the company details when any expense has reached the max marketing expense level and notice the funding for the next level is shown as ` `	UI	Pass	
25	Clicking the `Pause` button opens the Pause submenu popup screen	When the Pause button in the main game UI is clicked a new submenu screen, labelled Paused, is shown on top of the UI	Game play	Pass	
26	Pressing `Escape` or `P` toggles the pause menu on and off	Pressing the `P` or `Escape` key on the keyboard toggles the pause menu on and off, showing it and hiding it	Game play	Pass	The Pause menu can be toggled on when there is another submenu being shown, but as a pause menu this is acceptable and does not affect gameplay
27	Pressing the `Resume` button in the pause menu closes the Pause menu. This considering that	Pressing the `Resume` button in the Pause menu closes the menu screen.	Game play	Pass	

	the Pause menu has no Close button like the other submenus.				
28	Pressing the `Exit Game` button in the pause menu closes the game (build) or stops the editor play (Editor)	Pressing the `Exit Game` button in the Pause menu exits the game in builds and editor.	Game play	Pass	
29	Despite the Neural Network prediction results, the Net Income of the company cannot change more than the restricted amount (default of 10%) in comparison to the previous month. This is before any bonus is applied, like the research investment level bonus.	The Net Income during different months never changes more (positive or negative) than 10% in comparison to the previous month net income. This is ignoring any bonus in the game.	Game play	Pass	The bonus to the net income due to research investment, can make the net income change up to 20%, which can be confusing at points
30	Marketing and Research expenses can be improved based on investment levels	Open the company details screen and notice that the investment level is increased whenever the required amount of funding is met	Game play	Pass	
31	Marketing bonus increases the stock value of the company	Open the Company Details screen and notice the current bonus of the Marketing investment. Advance turns and notice the bonus effects	Game play	Pass	
32	Marketing bonus increases the net income of the company	Open the Company Details screen and notice the current bonus of the Research and Development investment. Advance turns and notice the bonus effects that increase the net income	Game play	Pass	

Version 0.6

Test

Date:

Wednesday, April 17th, 2024

Test

Version:

V0.6

Case ID	Description	Expected Result	Category	Pass/Fail	Actual Result	Comment	Jira Issue
1	All custom-made automated tests, "UnprofitableStart up" category, in Unreal Engine (Tools->Test Automation) are run.	All automation tests in the Unreal Engine project are successful	Automated Testing	Pass			

2	The script `NeuralNetworkTraining.py` works correctly, training the neural network and exporting it	The Neural Network model is exported as an ONNX file, and the Neural Network performance evaluation is acceptable	Neural Network	Pass	
3	The script `NeuralNetworkTraining.py` produces accurate data to scale/unscale the information from Unreal Engine.	The script produces `scaling_data_information.json` which correctly has the min and max values for each column.	Neural Network	Pass	
4	The data scaling JSON is exported as an unpackaged JSON in builds. This opposite of most of the game data, which is packaged, preventing it from being parsed.	The data scaling JSON is copied as `Content/Data/scaling_data_information.json` in builds, remaining unpackaged.	Builds	Pass	
5	The Neural Network, in isolation, produces consistent output results when the same input is given.	The Neural Network, in the Unreal Engine test scene, produces similar results to the PyTorch Neural Network script	Neural Network	Pass	Unreal Engine: 1,242,245,760 Python: 1,242,245,562.2878075
6	When the end turn button is pressed the game progresses the next turn	The end turn button is clickable, and refreshes the game Hud with the latest data (next turn number, overall profits, etc.)	Gameplay	Pass	
7	The financial data HUD shows tooltips when it is hovered over	Tooltips for the different UI elements are shown when they are hovered over	UI	Pass	
8	The game HUD tracks the UI correctly	The game HUD accurately tracks the turn number, net income, and other financial data.	UI	Fail	The monthly budget amount in the HUD differs from the Monthly Budget shown in the Company Details US-80
9	The net income HUD text changes depending on its value	When the net income is positive, or zero, a '+' is prefixed and the text turns green; when negative a '-' is prefixed and text colour is set to red.	UI	Pass	
10	In the UI, financial data is shown up as rounded up values	Values are shown rounded up, when above 100 units, for thousands (k), millions (m), and billions (b)	UI	Pass	

11	The script "FinancialData-AlphaVantageData.py" correctly filters the data and exports it as quarterly, monthly, and weekly	Running the Python script produces 3 files, with only the reduced data fields, for "quarterlyFinancialData.csv", "monthlyFinancialData.csv", and "weeklyFinancialData.csv"	Financial Data	Pass
12	The interpolated monthly and weekly financial data matches closely the original quarterly data obtained from AlphaVantage.	When running "FinancialData-AlphaVantageData.py", the value ranges and the graph representation of the quarterly data is close to the monthly and weekly data.	Financial Data	Pass
13	Clicking the expenditures button opens the Expenditures submenu popup screen	When the expenditures button in the main game UI is clicked a new submenu screen, labelled Expenditures, is shown on top of the UI	Gameplay	Pass
14	Other UI elements in the main game UI cannot be interacted with when a submenu screen is open.	Open the Expenditures submenu screen, and click the end turn button or the Expenditures button (again), which are visible under the submenu. Observe that nothing happens	UI	Pass
15	Submenu screens are properly hidden when the close button is pressed	Open the Expenditures submenu screen and click the close submenu button on the top right, notice that the submenu is closed/hidden from view and no error shows up.	UI	Pass
16	In the expenditures screen, the Research and Marketing sliders cannot be below 0, or, add it up, cannot go above the current monthly budget.	In the expenditures screen, moving the slider to the left goes down to 0. The sum of both slider values when maxed, does not go above the monthly budget.	UI	Pass
17	The Game HUD is updated when the expenditures screen is closed.	Opening the expenditure screen and changing the budget allocation updates the monthly expenses amount, this change is reflected in the main game HUD when the submenu is closed	UI	Pass

18	When opened, the expenditure screen shows the most up to date values for monthly budget, available funds for the month, and marketing and research expenses	The expenditure screen shows the correct monthly budget amount, available funds (monthly budget - monthly expenses), and marketing and research expenses (sliders correctly placed, and text displaying correct values)	UI	Pass
19	Adjusting the sliders in the expenditures screen changes the company's budget allocation	When the marketing and the research sliders are moved, the budget allocation for that particular expense is updated in the company. These changes are reflected in both the text under the slider, and the currently available funds amount.	Gameplay	Pass
20	The expenditure screen sliders snap their value per the nearest rounded up value, depending on the current budget amount.	When moving the expenses sliders in the Expenditures screen, this snap in values to the corresponding step size of million, hundred thousand, etc. This step size on the current monthly budget.	UI	Pass
21	The expenditures allocations are rebalanced, proportionally, if the new budget (after the end of a turn) has been reduced and is no longer enough to cover the allocated expenses	Having the budget fully allocated, ending a turn that has a reduced budget, causes the expenditures to be re-allocated proportionally per the new budget. These uses rounded down step size values and may not cover the 100% of expenses.	Gameplay	Pass
22	Clicking the Company Details button opens the Company Details submenu popup screen	When the Company Details button in the main game UI is clicked a new submenu screen, labelled Company Details, is shown on top of the UI	Gameplay	Pass
23	The company details screen shows the correct and up to date information when opened.	When the Company Details screen is opened, this shows the latest up to date information. This has the currency data shown as rounded up in thousands (k), millions (m) or billions (b) when appropriated.	UI	Fail

[US-80](#)

24	In the Company Details screen when at the max investment level, display the funding required for the next level as ` `	Open the company details when any expense has reached the max marketing expense level and notice the funding for the next level is shown as ` `	UI	Pass
25	Clicking the `Pause` button opens the Pause submenu popup screen	When the Pause button in the main game UI is clicked a new submenu screen, labelled Paused, is shown on top of the UI	Gameplay	Pass
26	Pressing `Escape` or `P` toggles the pause menu on and off	Pressing the `P` or `Escape` key on the keyboard toggles the pause menu on and off, showing it and hiding it	Gameplay	Pass
27	Pressing the `Resume` button in the pause menu closes the Pause menu. This considering that the Pause menu has no Close button like the other submenus.	Pressing the `Resume` button in the Pause menu closes the menu screen.	Gameplay	Pass
28	Pressing the `Exit Game` button in the pause menu closes the game (build) or stops the editor play (Editor)	Pressing the `Exit Game` button in the Pause menu exits the game in builds and editor.	Gameplay	Pass
29	Despite the Neural Network prediction results, the Net Income of the company cannot change more than the restricted amount (default of 10%) in comparison to the previous month. This is before any bonus is applied, like the research investment level bonus.	The Net Income during different months never changes more (positive or negative) than 10% in comparison to the previous month net income. This is ignoring any bonus in the game.	Gameplay	Pass
30	Marketing and Research expenses can be improved based on investment levels	Open the company details screen and notice that the investment level is increased whenever the required amount of funding is met	Gameplay	Pass
31	Marketing bonus increases the stock value of the company	Open the Company Details screen and notice the current bonus of the Marketing investment. Advance turns and notice the bonus effects	Gameplay	Pass

32	Marketing bonus increases the net income of the company	Open the Company Details screen and notice the current bonus of the Research and Development investment. Advance turns and notice the bonus effects that increase the net income	Gameplay	Pass	
33	The game performance, measured through its frame rate is good and steady, even on "expensive" actions like ending the turn/running the neural network simulation. This measured on a computer with performance similar to the ones in the university.	In the Unreal Engine editor, enabling the option to show FPS, and launching the game on fullscreen, the FPS remains steady.	Performance	Pass	Around 120 FPS
34	The game assets present no major issue, like clipping, flickering or shadow artifacts.	Launching the game, the 3D environment presents no visual glitch	Appearance	Pass	
35	There is a continuous day and night cycle.	The sun position changes throughout the game, simulating a day and night cycle	Appearance	Pass	
36	The computers show animated videos at different intervals.	The computer screens in the game show different videos, at different intervals, where it is not obvious, they are the same videos. These are looping infinitely.	Appearance	Pass	
37	The videos used in the game are correctly packed in the build.	The mp4 videos used in the game are in the "Content/Movies" folder of the project when packaged.	Builds	Pass	
38	The buttons change their appearance when they are hovered or pressed.	Hovering over a button change its appearance, as well as when the button is pressed.	UI	Pass	
39	There is a custom cursor image	The in-game cursor is always visible, and is shown with a custom image	UI	Fail	The custom cursor image is shown, but it moves weird/slowly in low framerates Likely issue caused by using Software Cursor instead of Hardware cursor US-79

40	There is a sound effect when buttons are hovered or pressed	Different sound effects are played when a button is hovered, and when a button is pressed.	Sound	Pass
41	The sliders in the expenditures have sound effects for when the slider is grabbed, and for when the slider is released.	A sound is played when the slider is grabbed. If the slider is released on a valid range, not over the available budget, an ok sound is played. Otherwise, if the slider is released above the possible budget a "bad" sound is played	Sound	Pass
42	A sound is played when the close button is pressed.	Opening the company details screen, and pressing the close button plays a sound	Sound	Pass
43	A sound is played when the end turn button is pressed	When the end turn button is pressed a sound is played.	Sound	Pass
44	Looping background music is played throughout the game.	The game plays a constantly looping background music. The music continues playing even when the game is paused.	Sound	Pass
45	The day night cycle is stopped when the game is paused.	Opening the pause menu stops the current day night and cycle advance.	Appearance	Pass

Other Bugs

Description	Expected Result	Actual Result	Comment	Jira Issue
Value overflow when reaching around 2,000,000,000 billion	Values never overflow to negative, and always show the consistent result.	When reaching around 2,000,000,000 billion the HUD display for Overall profit and Cash Available overflows, turning negative	Since this required around 200 turns and will likely be preceded by a win end game condition (before that amount is reached), this is low priority.	US-78

Version 0.7

Test

Date:

Tuesday, April 23rd, 2024

Test

Version:

V0.7

Case ID	Description	Expected Result	Category	Pass/Fail	Actual Result	Comment	Jira Issue
1	All custom-made automated tests, "UnprofitableStart up" category, in Unreal Engine (Tools->Test Automation) are run.	All automation tests in the Unreal Engine project are successful	Automated Testing	Pass			
2	The script `NeuralNetworkTraining.py` works correctly, training the neural network and exporting it	The Neural Network model is exported as an ONNX file and the Neural Network performance evaluation is acceptable	Neural Network	Pass			
3	The script `NeuralNetworkTraining.py` produces accurate data to scale/unscale the information from Unreal Engine.	The script produces `scaling_data_information.json` which correctly has the min and max values for each column.	Neural Network	Pass			
4	The data scaling JSON is exported as an unpackaged JSON in builds. This opposite of most of the game data, which is packaged, preventing it from being parsed.	The data scaling JSON is copied as `Content/Data/scaling_data_information.json` in builds, remaining unpackaged.	Builds	Pass			
5	The Neural Network, in isolation, produces consistent output results when the same input is given.	The Neural Network, in the Unreal Engine test scene, produces similar results to the PyTorch Neural Network script	Neural Network	Pass		Python: 1,107,681,955. 1634789 Unreal Engine: 1,107,681,920	
6	When the end turn button is pressed the game progresses the next turn	The end turn button is clickable, and refreshes the game Hud with the latest data (next turn number, overall profits, etc.)	Gameplay	Pass			
7	The financial data HUD shows tooltips when it is hovered over	Tooltips for the different UI elements are shown when they are hovered over	UI	Pass			
8	The game HUD tracks the UI correctly	The game HUD accurately tracks the turn number, net income, and other financial data.	UI	Pass			
9	The net income HUD text changes depending on its value	When the net income is positive, or zero, a '+' is prefixed and the text turns green; when negative a '-' is prefixed and text colour is set to red.	UI	Pass			

10	In the UI, financial data is shown up as rounded up values	Values are shown rounded up, when above 100 units, for thousands (k), millions (m), and billions (b)	UI	Pass
11	The script "FinancialData-AlphaVantageData.py" correctly filters the data and exports it as quarterly, monthly, and weekly	Running the Python script produces 3 files, with only the reduced data fields, for "quarterlyFinancialData.csv", "monthlyFinancialData.csv", and "weeklyFinancialData.csv"	Financial Data	Pass
12	The interpolated monthly and weekly financial data matches closely the original quarterly data obtained from AlphaVantage.	When running "FinancialData-AlphaVantageData.py", the value ranges and the graph representation of the quarterly data is close to the monthly and weekly data.	Financial Data	Pass
13	Clicking the expenditures button opens the Expenditures submenu popup screen	When the expenditures button in the main game UI is clicked a new submenu screen, labelled Expenditures, is shown on top of the UI	Gameplay	Pass
14	Other UI elements in the main game UI cannot be interacted with when a submenu screen is open.	Open the Expenditures submenu screen, and click the end turn button or the Expenditures button (again), which are visible under the submenu. Observe that nothing happens	UI	Pass
15	Submenu screens are properly hidden when the close button is pressed	Open the Expenditures submenu screen and click the close submenu button on the top right, notice that the submenu is closed/hidden from view and no error shows up.	UI	Pass
16	In the expenditures screen, the Research and Marketing sliders cannot be below 0, or, add it up, cannot go above the current monthly budget.	In the expenditures screen, moving the slider to the left goes down to 0. The sum of both slider values when maxed, does not go above the monthly budget.	UI	Pass
17	The Game HUD is updated when the expenditures screen is closed.	Opening the expenditure screen and changing the budget allocation updates the monthly expenses amount, this change is reflected in the main game HUD when the submenu is closed	UI	Pass

18	When opened, the expenditure screen shows the most up to date values for monthly budget, available funds for the month, and marketing and research expenses	The expenditure screen shows the correct monthly budget amount, available funds (monthly budget - monthly expenses), and marketing and research expenses (sliders correctly placed, and text displaying correct values)	UI	Pass
19	Adjusting the sliders in the expenditures screen changes the company's budget allocation	When the marketing and the research sliders are moved, the budget allocation for that particular expense is updated in the company. These changes are reflected in both the text under the slider, and the currently available funds amount.	Gameplay	Pass
20	The expenditure screen sliders snap their value per the nearest rounded up value, depending on the current budget amount.	When moving the expenses sliders in the Expenditures screen, this snap in values to the corresponding step size of million, hundred thousand, etc. This step size on the current monthly budget.	UI	Pass
21	The expenditures allocations are rebalanced, proportionally, if the new budget (after the end of a turn) has been reduced and is no longer enough to cover the allocated expenses	Having the budget fully allocated, ending a turn that has a reduced budget, causes the expenditures to be re-allocated proportionally per the new budget. These use rounded down step size values, and may not cover the 100% of expenses.	Gameplay	Pass
22	Clicking the Company Details button opens the Company Details submenu popup screen	When the Company Details button in the main game UI is clicked a new submenu screen, labelled Company Details, is shown on top of the UI	Gameplay	Pass
23	The company details screen shows the correct and up to date information when opened.	When the Company Details screen is opened, this shows the latest up to date information. This has the currency data shown as rounded up in thousands (k), millions (m) or billions (b) when appropriated.	UI	Pass

24	In the Company Details screen when at the max investment level, display the funding required for the next level as ` `	Open the company details when any expense has reached the max marketing expense level and notice the funding for the next level is shown as ` `	UI	Pass
25	Clicking the `Pause` button opens the Pause submenu popup screen	When the Pause button in the main game UI is clicked a new submenu screen, labelled Paused, is shown on top of the UI	Gameplay	Pass
26	Pressing `Escape` or `P` toggles the pause menu on and off	Pressing the `P` or `Escape` key on the keyboard toggles the pause menu on and off, showing it and hiding it	Gameplay	Pass
27	Pressing the `Resume` button in the pause menu closes the Pause menu. This considering that the Pause menu has no Close button like the other submenus.	Pressing the `Resume` button in the Pause menu closes the menu screen.	Gameplay	Pass
28	Pressing the `Exit Game` button in the pause menu closes the game (build) or stops the editor play (Editor)	Pressing the `Exit Game` button in the Pause menu exits the game in builds and editor.	Gameplay	Pass
29	Despite the Neural Network prediction results, the Net Income of the company cannot change more than the restricted amount (default of 10%) in comparison to the previous month. This is before any bonus is applied, like the research investment level bonus.	The Net Income during different months never changes more (positive or negative) than 10% in comparison to the previous month net income. This is ignoring any bonus in the game.	Gameplay	Pass
30	Marketing and Research expenses can be improved based on investment levels	Open the company details screen and notice that the investment level is increased whenever the required amount of funding is met	Gameplay	Pass
31	Marketing bonus increases the stock value of the company	Open the Company Details screen and notice the current bonus of the Marketing investment. Advance turns and notice the bonus effects	Gameplay	Pass

32	Marketing bonus increases the net income of the company	Open the Company Details screen and notice the current bonus of the Research and Development investment. Advance turns and notice the bonus effects that increase the net income	Gameplay	Pass	
33	The game performance, measured through its frame rate is good and steady, even on "expensive" actions like ending the turn/running the neural network simulation. This measured on a computer with performance similar to the ones in the university.	In the Unreal Engine editor, enabling the option to show FPS, and launching the game on fullscreen, the FPS remains steady.	Performance	Pass	Steady 120 FPS in the game using university PCs
34	The game assets present no major issue, like clipping, flickering or shadow artifacts.	Launching the game, the 3D environment presents no visual glitch	Appearance	Pass	
35	There is a continuous day and night cycle.	The sun position changes throughout the game, simulating a day and night cycle	Appearance	Pass	
36	The computers show animated videos at different intervals.	The computer screens in the game show different videos, at different intervals, where it is not obvious, they are the same videos. These are looping infinitely.	Appearance	Pass	
37	The videos used in the game are correctly packed in the build.	The mp4 videos used in the game are in the "Content/Movies" folder of the project when packaged.	Builds	Pass	
38	The buttons change their appearance when they are hovered or pressed.	Hovering over a button change its appearance, as well as when the button is pressed.	UI	Pass	
39	There is a custom cursor image	The in-game cursor is always visible, and is shown with a custom image	UI	Pass	
40	There is a sound effect when buttons are hovered or pressed	Different sound effects are played when a button is hovered, and when a button is pressed.	Sound	Pass	

41	The sliders in the expenditures have sound effects for when the slider is grabbed, and for when the slider is released.	A sound is played when the slider is grabbed. If the slider is released on a valid range, not over the available budget, an ok sound is played. Otherwise, if the slider is released above the possible budget a "bad" sound is played	Sound	Pass
42	A sound is played when the close button is pressed.	Opening the company details screen, and pressing the close button plays a sound	Sound	Pass
43	A sound is played when the end turn button is pressed	When the end turn button is pressed a sound is played.	Sound	Pass
44	Looping background music is played throughout the game.	The game plays a constantly looping background music. The music continues playing even when the game is paused.	Sound	Pass
45	The day night cycle is stopped when the game is paused.	Opening the pause menu stops the current day night and cycle advance.	Appearance	Pass
46	When the game is started an introduction screen is shown	After starting the game, a welcome submenu with basic instructions is shown	UI	Pass
47	If the user runs out of cash available an end game screen is shown	If the user runs out of cash available, an end game submenu is shown, notifying the player he went bankrupt and giving him the option to restart the game	Gameplay	Pass
48	If the user survives (does not run out of cash) after the max number of turns he receives a victory screen	Advance the game to turn 60, without running out of cash. Observe a win game, notifying the player "survived" is shown. This gives the player the option to continue playing or restart the game.	Gameplay	Pass
49	If the company becomes profitable and it accrues a certain amount of cash, a victory screen is shown	If the player achieves a significantly positive net income and a good cash reserve, a victory screen is shown informing the player he made the company successful. This gives the player the option to continue playing or restart the game.	Gameplay	Pass

50	A sound effect is playing when the end game condition is played.	Different sound effects are played depending on the condition the game ended, bankruptcy, max turns, or profitability	Sound	Pass
----	--	---	-------	------

Other Bugs

Description	Expected Result	Actual Result	Comment	Jira Issue
If the game is continued to be played after a victory end game condition has been reached, and the player runs out of cash available the game soft locks	If the player runs out of cash after he wins the game and decides to continue playing, the standard game over the screen is shown.	The player has no cash left, so he has no budget to allocate or any other action besides ending his turn.		US-81

Version 0.8

Test

Date:

Friday, April 26th, 2024

Test

Version:

V0.8

Case ID	Description	Expected Result	Category	Pass/Fail	Actual Result	Comment	Jira Issue
1	All custom-made automated tests, "UnprofitableStartup" category, in Unreal Engine (Tools->Test Automation) are run.	All automation tests in the Unreal Engine project are successful	Automated Testing	Pass			
2	The script `NeuralNetworkTraining.py` works correctly, training the neural network and exporting it	The Neural Network model is exported an ONNX file and the Neural Network performance evaluation is acceptable	Neural Network	Pass			
3	The script `NeuralNetworkTraining.py` produces accurate data to scale/unscale the information from Unreal Engine.	The script produces `scaling_data_information.json` which correctly has the min and max values for each column.	Neural Network	Pass			

4	The data scaling JSON is exported as an unpackaged JSON in builds. This is opposite of most of the game data, which is packaged, preventing it from being parsed.	The data scaling JSON is copied as <code>`Content/Data/scaling_data_information.json`</code> in builds, remaining unpackaged.	Builds	Pass	
5	The Neural Network, in isolation, produces consistent output results when the same input is given.	The Neural Network, in the Unreal Engine test scene, produces similar results to the PyTorch Neural Network script	Neural Network	Pass	Python: 1,107,681,955 Unreal Engine: 1,107,681,920
6	When the end turn button is pressed the game progresses the next turn	The end turn button is clickable, and refreshes the game Hud with the latest data (next turn number, overall profits, etc.)	Gameplay	Pass	
7	The financial data HUD shows tooltips when it is hovered over	Tooltips for the different UI elements are shown when they are hovered over	UI	Pass	
8	The game HUD tracks the UI correctly	The game HUD accurately tracks the turn number, net income, and other financial data.	UI	Pass	
9	The net income HUD text changes depending on its value	When the net income is positive, or zero, a '+' is prefixed and the text turns green; when negative a '-' is prefixed and text colour is set to red.	UI	Pass	
10	In the UI, financial data is shown up as rounded up values	Values are shown rounded up, when above 100 units, for thousands (k), millions (m), and billions (b)	UI	Pass	
11	The script "FinancialData-AlphaVantageData.py" correctly filters the data and exports it as quarterly, monthly, and weekly	Running the Python script produces 3 files, with only the reduced data fields, for "quarterlyFinancialData.csv", "monthlyFinancialData.csv", and "weeklyFinancialData.csv"	Financial Data	Pass	
12	The interpolated monthly and weekly financial data matches closely the original quarterly data obtained from AlphaVantage.	When running "FinancialData-AlphaVantageData.py", the value ranges and the graph representation of the quarterly data is close to the monthly and weekly data.	Financial Data	Pass	

13	Clicking the expenditures button opens the Expenditures submenu popup screen	When the expenditures button in the main game UI is clicked a new submenu screen, labelled Expenditures, is shown on top of the UI	Gameplay	Pass
14	Other UI elements in the main game UI cannot be interacted with when a submenu screen is open.	Open the Expenditures submenu screen, and click the end turn button or the Expenditures button (again), which are visible under the submenu. Observe that nothing happens	UI	Pass
15	Submenu screens are properly hidden when the close button is pressed	Open the Expenditures submenu screen and click the close submenu button on the top right, notice that the submenu is closed/hidden from view and no error shows up.	UI	Pass
16	In the expenditures screen, the Research and Marketing sliders cannot be below 0, or, add it up, cannot go above the current monthly budget.	In the expenditures screen, moving the slider to the left goes down to 0. The sum of both slider values when maxed, does not go above the monthly budget.	UI	Pass
17	The Game HUD is updated when the expenditures screen is closed.	Opening the expenditure screen and changing the budget allocation updates the monthly expenses amount, this change is reflected in the main game HUD when the submenu is closed	UI	Pass
18	When opened, the expenditure screen shows the most up to date values for monthly budget, available funds for the month, and marketing and research expenses	The expenditure screen shows the correct monthly budget amount, available funds (monthly budget - monthly expenses), and marketing and research expenses (sliders correctly placed, and text displaying correct values)	UI	Pass
19	Adjusting the sliders in the expenditures screen changes the company's budget allocation	When the marketing and the research sliders are moved, the budget allocation for that particular expense is updated in the company. These changes are reflected in both the text under the slider, and the currently available funds amount.	Gameplay	Pass

20	The expenditure screen sliders snap their value per the nearest rounded up value, depending on the current budget amount.	When moving the expenses sliders in the Expenditures screen, this snap in values to the corresponding step size of million, hundred thousand, etc. This step size on the current monthly budget.	UI	Pass
21	The expenditures allocations are rebalanced, proportionally, if the new budget (after the end of a turn) has been reduced and is no longer enough to cover the allocated expenses	Having the budget fully allocated, ending a turn that has a reduced budget, causes the expenditures to be re-allocated proportionally per the new budget. These use rounded down step size values, and may not cover the 100% of expenses.	Gameplay	Pass
22	Clicking the Company Details button opens the Company Details submenu popup screen	When the Company Details button in the main game UI is clicked a new submenu screen, labelled Company Details, is shown on top of the UI	Gameplay	Pass
23	The company details screen shows the correct and up to date information when opened.	When the Company Details screen is opened, this shows the latest up to date information. This has the currency data shown as rounded up in thousands (k), millions (m) or billions (b) when appropriated.	UI	Pass
24	In the Company Details screen when at the max investment level, display the funding required for the next level as ` `	Open the company details when any expense has reached the max marketing expense level and notice the funding for the next level is shown as ` `	UI	Pass
25	Clicking the `Pause` button opens the Pause submenu popup screen	When the Pause button in the main game UI is clicked a new submenu screen, labelled Paused, is shown on top of the UI	Gameplay	Pass
26	Pressing `Escape` or `P` toggles the pause menu on and off	Pressing the `P` or `Escape` key on the keyboard toggles the pause menu on and off, showing it and hiding it	Gameplay	Pass
27	Pressing the `Resume` button in the pause menu closes the Pause menu. This considering that the Pause menu	Pressing the `Resume` button in the Pause menu closes the menu screen.	Gameplay	Pass

	has no Close button like the other submenus.				
28	Pressing the `Exit Game` button in the pause menu closes the game (build) or stops the editor play (Editor)	Pressing the `Exit Game` button in the Pause menu exits the game in builds and editor.	Gameplay	Pass	
29	Despite the Neural Network prediction results, the Net Income of the company cannot change more than the restricted amount (default of 10%) in comparison to the previous month. This is before any bonus is applied, like the research investment level bonus.	The Net Income during different months never changes more (positive or negative) than 10% in comparison to the previous month net income. This is ignoring any bonus in the game.	Gameplay	Pass	
30	Marketing and Research expenses can be improved based on investment levels	Open the company details screen and notice that the investment level is increased whenever the required amount of funding is met	Gameplay	Pass	
31	Marketing bonus increases the stock value of the company	Open the Company Details screen and notice the current bonus of the Marketing investment. Advance turns and notice the bonus effects	Gameplay	Pass	
32	Marketing bonus increases the net income of the company	Open the Company Details screen and notice the current bonus of the Research and Development investment. Advance turns and notice the bonus effects that increase the net income	Gameplay	Pass	
33	The game performance, measured through its frame rate is good and steady, even on "expensive" actions like ending the turn/running the neural network simulation. This measured on a computer with performance similar to the ones in the university.	In the Unreal Engine editor, enabling the option to show FPS, and launching the game on fullscreen, the FPS remains steady.	Performance	Pass	Consistent 120.0 FPS in university computers

34	The game assets present no major issue, like clipping, flickering or shadow artifacts.	Launching the game, the 3D environment presents no visual glitch	Appearance	Pass
35	There is a continuous day and night cycle.	The sun position changes throughout the game, simulating a day and night cycle	Appearance	Pass
36	The computers show animated videos at different intervals.	The computer screens in the game show different videos, at different intervals, where it is not obvious, they are the same videos. These are looping infinitely.	Appearance	Pass
37	The videos used in the game are correctly packed in the build.	The mp4 videos used in the game are in the "Content/Movies" folder of the project when packaged.	Builds	Pass
38	The buttons change their appearance when they are hovered or pressed.	Hovering over a button change its appearance, as well as when the button is pressed.	UI	Pass
39	There is a custom cursor image	The in-game cursor is always visible, and is shown with a custom image	UI	Pass
40	There is a sound effect when buttons are hovered or pressed	Different sound effects are played when a button is hovered, and when a button is pressed.	Sound	Pass
41	The sliders in the expenditures have sound effects for when the slider is grabbed, and for when the slider is released.	A sound is played when the slider is grabbed. If the slider is released on a valid range, not over the available budget, an ok sound is played. Otherwise, if the slider is released above the possible budget a "bad" sound is played	Sound	Pass
42	A sound is played when the close button is pressed.	Opening the company details screen, and pressing the close button plays a sound	Sound	Pass
43	A sound is played when the end turn button is pressed	When the end turn button is pressed a sound is played.	Sound	Pass
44	Looping background music is played throughout the game.	The game plays a constantly looping background music. The music continues playing even when the game is paused.	Sound	Pass

45	The day night cycle is stopped when the game is paused.	Opening the pause menu stops the current day night and cycle advance.	Appearance	Pass
46	When the game is started an introduction screen is shown	After starting the game, a welcome submenu with basic instructions is shown	UI	Pass
47	If the user runs out of cash available an end game screen is shown	If the user runs out of cash available, an end game submenu is shown, notifying the player he went bankrupt and giving him the option to restart the game	Gameplay	Pass
48	If the user survives (does not run out of cash) after the max number of turns he receives a victory screen	Advance the game to turn 60, without running out of cash. Observe a win game, notifying the player "survived" is shown. This gives the player the option to continue playing or restart the game.	Gameplay	Pass
49	If the company becomes profitable and it accrues a certain amount of cash, a victory screen is shown	If the player achieves a significantly positive net income and a good cash reserve, a victory screen is shown informing the player he made the company successful. This gives the player the option to continue playing or restart the game.	Gameplay	Pass
50	A sound effect is playing when the end game condition is played.	Different sound effects are played depending on the condition the game ended, bankruptcy, max turns, or profitability	Sound	Pass
51	The help menu can be opened	Clicking the help menu on the HUD opens the help popup screen. This display buttons with further subscreens of the help menu.	Gameplay	Pass
52	The help menu displays a how to play section	After opening the help menu, clicking the How To Play button shows a screen with instructions on how to play the game.	UI	Pass

53	The help menu displays a financial terms dictionary	After opening the help menu, clicking the Financial Dictionary button shows a screen with different financial terms used in the game and their definition.	UI	Pass
54	If you win the game, and decide to continue playing, you can still lose the game if you run out of cash.	Win the game, either by reaching the max turns or by earning enough money, and then progress the game until you run out of cash. Observe that the lose screen is shown.	Gameplay	Pass
55	A highlighted version of the HUD, with instructions, is shown at the welcome screen and at the help/how to play menu.	Notice in the introduction screen the image with instructions. Open the Help menu, How To Play, and notice the same screen with instruction is shown.	UI	Pass
56	The game can be restarted from the pause menu.	Open the pause menu, and click the Restart Game button, and the current level is reloaded.	UI	Pass
57	The net income has a minimum amount (by default 100,000) that can change each month before it is capped.	When the net income is reaching values near 0, notice that the net income is never capped to small, ever decreasing values.	Gameplay	Pass
58	There is an icon showing the change of the net income in relation to the previous month.	If, in comparison to the previous month, the net income has increased a green arrow is shown next to the net income HUD text. If this has decreased, a red arrow is shown. In the odd case that it stays the same, an equal sign is shown.	UI	Pass

Appendix 6 – Alpha Vantage Netflix financial statements samples

Cashflow

```
{
  "symbol": "NFLX",
  "annualReports": [
    {
      "fiscalDateEnding": "2023-12-31",
      "reportedCurrency": "USD",
      "operatingCashflow": "7274301000",
      "paymentsForOperatingActivities": "1839477000",
      "proceedsFromOperatingActivities": "None",
      "changeInOperatingLiabilities": "375775000",
      "changeInOperatingAssets": "181003000",
      "depreciationDepletionAndAmortization": "356947000",
      "capitalExpenditures": "348552000",
      "changeInReceivables": "None",
      "changeInInventory": "None",
      "profitLoss": "5407990000",
      "cashflowFromInvestment": "541751000",
      "cashflowFromFinancing": "-5950803000",
      "proceedsFromRepaymentsOfShortTermDebt": "None",
      "paymentsForRepurchaseOfCommonStock": "6045347000",
      "paymentsForRepurchaseOfEquity": "6045347000",
      "paymentsForRepurchaseOfPreferredStock": "None",
      "dividendPayout": "None",
      "dividendPayoutCommonStock": "None",
      "dividendPayoutPreferredStock": "None",
      "proceedsFromIssuanceOfCommonStock": "None",
      "proceedsFromIssuanceOfLongTermDebtAndCapitalSecuritiesNet":
"None",
      "proceedsFromIssuanceOfPreferredStock": "None",
      "proceedsFromRepurchaseOfEquity": "-5875357000",
      "proceedsFromSaleOfTreasuryStock": "None",
      "changeInCashAndCashEquivalents": "None",
      "changeInExchangeRate": "None",
      "netIncome": "5407990000"
    }
  ]
}
```

Earnings

```
{
  "symbol": "NFLX",
  "annualEarnings": [
    {
      "fiscalDateEnding": "2023-12-31",
      "reportedEPS": "12.01"
    }
  ]
}
```


Income Statement

```
{
  "symbol": "NFLX",
  "annualReports": [
    {
      "fiscalDateEnding": "2023-12-31",
      "reportedCurrency": "USD",
      "grossProfit": "14007929000",
      "totalRevenue": "33723297000",
      "costOfRevenue": "19715368000",
      "costofGoodsAndServicesSold": "19715368000",
      "operatingIncome": "6954003000",
      "sellingGeneralAndAdministrative": "6110168000",
      "researchAndDevelopment": "2675758000",
      "operatingExpenses": "8785926000",
      "investmentIncomeNet": "None",
      "netInterestIncome": "-699826000",
      "interestIncome": "None",
      "interestExpense": "699826000",
      "nonInterestIncome": "-176296000",
      "otherNonOperatingIncome": "None",
      "depreciation": "None",
      "depreciationAndAmortization": "None",
      "incomeBeforeTax": "6205405000",
      "incomeTaxExpense": "797415000",
      "interestAndDebtExpense": "699826000",
      "netIncomeFromContinuingOperations": "5407990000",
      "comprehensiveIncomeNetOfTax": "5401351000",
      "ebit": "6905231000",
      "ebitda": "6954003000",
      "netIncome": "5407990000"
    }
  ]
}
```

Monthly Time Series

```
{
  "Meta Data": {
    "1. Information": "Monthly Prices (open, high, low, close) and  
Volumes",
    "2. Symbol": "nflx",
    "3. Last Refreshed": "2024-01-26",
    "4. Time Zone": "US/Eastern"
  },
  "Monthly Time Series": {
    "2024-01-26": {
      "1. open": "483.1850",
      "2. high": "579.6400",
      "3. low": "461.8600",
      "4. close": "570.4200",
      "5. volume": "127854895"
    }
  }
}
```

```
}  
}
```

Balance Sheet

```
{  
  "symbol": "NFLX",  
  "annualReports": [  
    {  
      "fiscalDateEnding": "2023-12-31",  
      "reportedCurrency": "USD",  
      "totalAssets": "48731992000",  
      "totalCurrentAssets": "9918133000",  
      "cashAndCashEquivalentsAtCarryingValue": "7116913000",  
      "cashAndShortTermInvestments": "7139488000",  
      "inventory": "408936000",  
      "currentNetReceivables": "1639257000",  
      "totalNonCurrentAssets": "9232702000",  
      "propertyPlantEquipment": "1491444000",  
      "accumulatedDepreciationAmortizationPPE": "855043000",  
      "intangibleAssets": "31658056000",  
      "intangibleAssetsExcludingGoodwill": "31658056000",  
      "goodwill": "None",  
      "investments": "30934000",  
      "longTermInvestments": "None",  
      "shortTermInvestments": "20973000",  
      "otherCurrentAssets": "2780247000",  
      "otherNonCurrentAssets": "5664359000",  
      "totalLiabilities": "28143679000",  
      "totalCurrentLiabilities": "8860655000",  
      "currentAccountsPayable": "747412000",  
      "deferredRevenue": "1442969000",  
      "currentDebt": "400000000",  
      "shortTermDebt": "399844000",  
      "totalNonCurrentLiabilities": "18751652000",  
      "capitalLeaseObligations": "None",  
      "longTermDebt": "14478000000",  
      "currentLongTermDebt": "399844000",  
      "longTermDebtNoncurrent": "14143417000",  
      "shortLongTermDebtTotal": "14878000000",  
      "otherCurrentLiabilities": "6270430000",  
      "otherNonCurrentLiabilities": "2561434000",  
      "totalShareholderEquity": "20588313000",  
      "treasuryStock": "6922200000",  
      "retainedEarnings": "22589286000",  
      "commonStock": "5145172000",  
      "commonStockSharesOutstanding": "432759584"  
    },  
  ]  
}
```

Appendix 7 – Completed work

Completed Issues

Key	T	Summary	Status	P	Parent	Fix versions
US-70	✓	Create company details submenu	DONE	✓	Accessible financial data	V0.5
US-35	✓	Round the financial data into easy to read values	DONE	✓	Accessible financial data	V0.3
US-52	✓	Create 3D animated background for Main Game level	DONE	✓	Animated background	V0.6
US-38	✓	US-36 Create financial terms dictionary	DONE	✓	Create help submenu	V0.8
US-37	✓	US-36 Create gameplay help section	DONE	✓	Create help submenu	V0.8
US-40	✓	Add tooltips to the different UI elements explaining what they do	DONE	✓	Financial data tooltips and help section	V0.5
US-36	✓	Create help submenu	DONE	✓	Financial data tooltips and help section	V0.8
US-52	✓	Add more up-to-date financial data	DONE	✓	Gathering Financial Data	V0.4
US-28	✓	Interpolate the Neural Network financial data to be weekly	DONE	✓	Gathering Financial Data	V0.4
US-21	✓	Scale the financial data equated and subtracted from the Neural Network	DONE	✓	Gathering Financial Data	V0.1
US-84	✓	Improve the readability of the net income value in relation to the monthly expenses	DONE	✓	Intuitive to use	V0.8
US-83	✓	Create a more visual label for the introduction of the game	DONE	✓	Intuitive to use	V0.8
US-82	✓	Add a visual indicator of your performance in comparison to the previous turn	DONE	✓	Intuitive to use	V0.8
US-59	✓	Create a start of the game pop-up explaining the objective of the game	DONE	✓	Intuitive to use	V0.7
US-34	✓	Organize usability testing of the game	DONE	✓	Intuitive to use	V0.8
US-53	✓	Create expenditures screen	DONE	✓	Manage budget	V0.4
US-65	✓	Increase the effect of the Marketing and Research expenses in the game	DONE	✓	Net Income simulation based on player's actions	V0.5
US-54	✓	Cap the amount that the Net Income can change between months	DONE	✓	Net Income simulation based on player's actions	V0.5
US-27	✓	Create core game manager functionality	DONE	✓	Net Income simulation based on player's actions	V0.2
US-25	✓	Create workflow HUD to visualize the player's company financial data	DONE	✓	Net Income simulation based on player's actions	V0.2
US-24	✓	Create Company object in Unreal Engine	DONE	✓	Net Income simulation based on player's actions	V0.2
US-22	✓	Create Unreal Engine test scene for the Neural Network	DONE	✓	Net Income simulation based on player's actions	V0.1
US-29	✓	Improve the performance of the Neural Network	DONE	✓	Neural Network is used for simulations	V0.7
US-23	✓	Create Net Income Inference Neural Network in Unreal Engine	DONE	✓	Neural Network is used for simulations	V0.1
US-33	✓	Create a way to measure the game's performance	DONE	✓	Optimal performance	V0.6
US-48	✓	Create pause menu screen	DONE	✓	Pause Menu	V0.5
US-43	✓	Add background music to the main game	DONE	✓	Sounds and music	V0.6
US-42	✓	Add sound effects for the game UI	DONE	✓	Sounds and music	V0.6
US-26	✓	Create button to end the current turn	DONE	✓	Turn based actions	V0.2
US-87	✓	When the net income is capped, the decreasing change can get stuck in minimal increments/decrements	DONE	✓		V0.8
US-85	✓	Create button to restart the game in the pause menu	DONE	✓		V0.8
US-81	✓	The game soft-locks after running of cash if the game has already been won	DONE	✓		V0.8
US-80	✓	Mismatch between the monthly budget shown in the company details and the in-game HUD	DONE	✓		V0.7
US-79	✓	The custom mouse cursor image lags when low framerate	DONE	✓		V0.7
US-76	✓	Likelihood available funds may not be available for a stopped up expense type	DONE	✓		V0.6
US-75	✓	The overall monthly expenses is not displayed in the Company Details screen	DONE	✓		V0.6
US-72	✓	Improve the game balance	DONE	✓		V0.7
US-71	✓	The budget allocation for each expense type is not updated if the monthly budget is reduced	DONE	✓		V0.5
US-68	✓	Finalise the game UI assets to improve the appearance of the game	DONE	✓		V0.6
US-64	✓	The expenditures screen slider values do not snap to the step size	DONE	✓		V0.5
US-59	✓	Create a win game condition	DONE	✓		V0.7
US-58	✓	Create a loss end game condition	DONE	✓		V0.7
US-57	✓	Moving the mouse cursor moves the camera	DONE	✓		V0.3
US-36	✓	The monthly budget is bigger than the overall cash available	DONE	✓		V0.3
US-55	✓	Create custom mouse cursor icon	DONE	✓		V0.6
US-45	✓	Create demo video of the game	DONE	✓		V0.8

Requirements Met

Key	T	Summary	Status	Requirement Priority	Fix versions
US-20	✓	Gathering Financial Data	DONE	Essential	V0.4
US-19	✓	The Neural Network model is stored as an ONNX file	DONE	Essential	V0.1
US-18	✓	Neural Network is used for simulations	DONE	Essential	V0.7
US-17	✓	Optimal performance	DONE	Essential	V0.6
US-16	✓	Intuitive to use	DONE	Essential	V0.8
US-15	✓	Accessible financial data	DONE	Essential	V0.5
US-9	✓	Financial data tooltips and help section	DONE	Desirable	V0.8
US-8	✓	Sounds and music	DONE	Desirable	V0.6
US-6	✓	Pause Menu	DONE	Desirable	V0.5
US-4	✓	Animated background	DONE	Desirable	V0.6
US-3	✓	Turn-based actions	DONE	Essential	V0.4
US-2	✓	Manage budget	DONE	Essential	V0.4
US-1	✓	Net Income simulation based on player's actions	DONE	Essential	V0.5

Appendix 8 – Python required packages

The creation and training of the neural network was done through different Python scripts, which used different packages to implement this functionality, most notably PyTorch, for creating the neural network.

The list of packages used for the Python component (Python 3.11.7) of this project are:

- pandas 2.2.1
- PyTorch 2.2.1
- scikit-learn 1.4.1
- matplotlib 3.8.3
- netron 7.5.4
- onnx 1.15.0