THE CELESTIAL BEYONDS

How Artificial Intelligence implementation can support companionship in Serious Games

John Shields

August 12, 2022

Supervised by
Dr. Sandy Louchart

School of Simulation and Visualisation
The Glasgow School of Art

THE GLASGOW SCHOOL OF ART

Submitted in partial fulfilment of the MSc in Serious Games and Virtual Reality

Abstract

Artificial Intelligence (AI) is becoming quite popular in our world. While AI technology is still relatively new, it is advancing progressively. There are many uses and different forms of AI. One form it takes is companionship. AI companionship is growing through its use in personal voice assistants; Siri, Alexa and Hey Google, chat bot apps such as Replika, is highly referenced in pop culture and is used throughout sci-fi films and games. This research area looks to explore the implementation of a supportive AI Companion into a Serious Game. It hopes to study and see if it is beneficial for future research and implementation to have AI companions in games. Mainly it focuses on how AI implementation can support companionship in Serious Games. The technologies used for the AI companion A WebGL sandbox style game in the form of a platformer set in space. The game's characters, the narrative and story-telling is where the research area comes in. The game features two main characters, Captain Hume (player) and his AI companion Moonbeam and together they pollinate different planets throughout the galaxy. With the game set up, it serves the purpose of gathering data on the said research area. The data was gathered through a survey that focuses on how Moonbeam supports Hume and the player themselves throughout the game. The gathering was done in two phases. The first was when the game had just one level that was functional and the second was with the entire game (3 main levels). The first test introduced the game, Hume and Moonbeam, in which the people got a sense of their relationship and overall said it worked well. However from the responses of the second test it was made clear that the testers were able to bond more with Moonbeam and see how he can support Hume in a larger scale. From the responses on the surveys it was made clear that it is defiantly worth pursuing this area in a wider matter as players found Moonbeam to be a supportive companion who adds to the experience of the game and certainly is supportive in terms of companionship.

Acknowledgements

I would like to thank my lecturers from this Masters and also from my Undergrad throughout the years, along with my supervisor, Sandy Louchart, for his advice, discussions, suggestions and for pointing me in the right direction throughout the summer.

I would like to thank the people at Infinity Ward and Activision for inspiring the research area and the game itself from one of the scenes from the game titled 'Call of Duty: Infinite Warfare'.

While Infinite Warfare inspired the research area, there is also a thank you in store to NASA as their mission with Voyager 1 and the Golden Records inspired the narrative and overall theme of searching for the remnants of humanity far in the future which makes for a lonely setting and this is where Moonbeam shines the most as he is a light and a friend in the game's lonely world.

Thank you to every participant who agreed to take part, who of course, cannot be named.

Sam McMillan and Michael Borodin for being the voices of Argyle and Viktor Van Gun.

Lastly, I cannot express my gratitude enough for the people who were so kind to record the messages of the Golden Record which extensively added to the impact of the game's narrative.

Contents

List of Figures

List of Tables

Declaration of Originality

- STUDENT ID No.: 21085609
- Name: John Shields
- Course/Programme: MSc, Serious Games & Virtual Reality
- Title of Work: The Celestial Beyonds - How Artificial Intelligence implementation can support companionship in Serious Games.

I confirm that all this work is my own except where indicated, and that I have:

- Clearly referenced/listed all sources as appropriate
- Referenced and added inverted commas to all quoted text (from books, journals, web, etc)
- Given the sources of all pictures, sound, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present, or lifted
- Not sought or used the help of any external professional agencies for the work
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Complied with any other plagiarism criteria specified in the Course Handbook

I understand that any false claim for this work will be penalised in accordance with the GSA regulations

Signature: John Shields
Date: 12/08/2022

Definitions and Abbreviations

AI	Artificial Intelligence
ML	Machine Learning
NASA	National Aeronautics and Space Administration
Unity	Unity Game Engine
Autodesk 3D Studio Max	3Ds Max
WebGL	Web Graphics Library
SG	Serious Games
FB	Figure Below
UI	User Interface
HUD	Heads-Up Display
API	Application Programming Interface
HTML	HyperText Markup Language
CSS	Cascading Style Sheet
AWS	Amazon Web Services
EC2	Elastic Compute Cloud
VPC	Virtual Private Cloud
TCP	Transmission Control Protocol
SSH	Secure Shell
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
Go	Golang Programming Language
JSON	JavaScript Object Notation
SQL	Structured Query Language
WWW	World Wide Web
VM	Virtual Machine
IDE	Integrated Development Environment
NPC	Non Playable Character
DLC	Downloadable Content
AAA game	High-budget and high-profile game

Game Characters

Captain Hume	Protagonist and Player
Moonbeam	Captain Hume's AI Companion
Aristaeus	Antagonist and Narrator
Argyle	Pollen Merchant
Viktor Van Gun	Arms Dealer

Chapter 1

Introduction

1.1 Project Introduction

In an achievable manner and at a Masters of Science standard, I aimed to develop a WebGL game titled 'The Celestial Beyonds' for the *MSc in Serious Games and Virtual Reality Dissertation*. The game mixes Space Hub Exploration, Platformers, Sandbox levels, Action, Pollination and Bees, which is set in Space or as the game lore refers to it, 'The Beyonds'. However, the research area focuses on *How Artificial Intelligence implementation can support companionship in Serious Games*. The research area was inspired by one single scene in an AAA game titled 'Infinite Warfare' from the Call of Duty franchise. The scene is an emotional one where two characters, Captain Reyes and his robot companion Ethan, have been involved in a spaceship crash. In that scene, the two characters share some very touching lines; Ethan says "My mission is you." Reyes replies, "Who says?", then Ethan responds by saying "I'm hardware, sir. Ultimately expendable." and afterwards Reyes says "No no. You're my brother Ethan." This one scene impacted me and compelled to me research the area and develop a game that featured an AI Companion. With the research area acquired, the methods of AI Companionship, Machine Learning, Chat bots and Serious Games will be explored adequately throughout this paper.

The game developed follows the adventures of Captain Hume and his loyal sidekick, Moonbeam. The game's world has been slowly dying and it is up to Hume and Moonbeam to go planets throughout the Beyonds and pollinate the plants on each planet. However, these planets are not safe as there is enemies lurking around them and attack on site. In the game Moonbeam supports Hume by being at his side and open to having a chat whenever

the player wishes to do so. In terms of Serious Games, Moonbeam works as a tour guide throughout the game's 3 main levels (planets). Each planet has 10 'Artifacts'. These Artifacts are from the game's world and lore but also from the real-world's history but for the purposes of fitting the game their origins are loosely altered to fit the game's world and lore.

This project is at Masters standard as the research has been thoroughly investigated and successfully applied to the development of the game. To explain this Moonbeam, is a fully functional AI companion that is supportive towards the player as discovered from the survey responses. Behind the character he is a Machine Learning Chat bot programmed in Python using the package Chatterbot. While Moonbeam is a character in the game, he is also a fully functional web page set up with Python's package Flask. The web page was required as the game was planned to be a WebGL based game meaning it would need to send HTTP requests to retrieve data, therefore the Moonbeam web page is hosted through an EC2 instance on AWS. Furthermore, the game has all it's required/intended functionality which includes, 3rd Person Player controls, movement and combat with enemies, Moonbeam implemented into the game, Dialogue System for General chat and Artifacts with Moonbeam and the player, Trading Systems, 3 Planet levels, a Boss Battle, Cinematics to tell the Narrative and as a bonus the game has a Photo Mode too.

The purpose of this dissertation is to unpack the planning, implementation and reflection of the research and technologies of the game and the objectives achieved, how it was designed and how it all came to together as a whole. It will firstly investigate the literature that relates to the research and how it was utilised in the game. Secondly it will discuss the material used to create the implementation of the WebGL game and the Moonbeam API. As well as explaining the research strategies, the methodology of the project's development and the testing procedure. Following on, it will demonstrate the design and implementation of the project. Second to last it will discuss the testing Survey results in analysis to the research and what they attempted to draw connections between key data points of interest. Lastly, the conclusion will then summarise and reflect on the achievements of the project's main objectives.

1.2 Research & Practical Objectives

The overall aim of the project was to achieve the following objectives:

- Research how AI implementation can support companionship in Serious Games
  - AI Overview
  - AI companionship
  - Machine Learning
  - Chat bots
  - Path-finding
  - AI companions in Games
  - Serious Games
- Develop an AI companion
  - Python & Chatterbot - Machine Learning Chat bot.
  - Python & Flask API for Chat bot.
  - Training data on general conversation and the game's lore.
  - Hosting of the API on AWS.
  - Design 3D Model and Animations.
  - Implementation into the game.
  - Web Requests for the chat bot data from the API.
  - Path-finding (to follow the player).
  - General Dialogue System with the player.
  - Game World Artifacts interaction and information System.
  - Combat Support
- Develop a WebGL Game
  - Design 3D Models and animations.
  - Player movement and Combat with enemies functionality.
  - Pollination System with dead Plants in levels.
  - In-game Trading Systems for Pollen and Ammo
  - Level design of 3 'Sandbox' Planets + Boss Battle.

  - Narrative about NASA's Voyager 1 and the Golden Record.

- Surveys & Testing
  - Moonbeam API and Level 1 Functionally for Test Demo.
  - Google Form Survey for Test Demo.
  - Send Test Demo to users and study their responses they gave in the form.
  - Use the feedback responses from the form to improve the game.
  - Finish Game.
  - Google Form Survey based on research and another for gameplay.
  - Send Final Game to users and study their responses they gave in the form.

1.3 Resources URLs

- GitHub Repository: `https://github.com/johnshields/celestial-beyonds`
- Game Jolt: `https://gamejolt.com/games/celestial-beyonds/740687`
- Moonbeam API Webpage: `https://api.moonbeambot.live/`
- PhotoMode Webpage: `https://scarlett-photographer-bot.herokuapp.com/`
- Video Presentation `https://youtu.be/yDOdMUqAN38`

1.3.0.1 Repository Contents

- celestial-beyonds_src
  - Source of Unity Project and C# code of the game.
- moonbeam
  - Source of Python code of the Moonbeam API.
- scarlett-bot_gallery
  - Source code of PhotoMode Website.
- workings
  - All models, art and miscellaneous files for rough work.

1.4 Chapter Descriptions

Chapter 2 - Literature Review

The Literature Review shows the research involved in the project in the detail of every section of AI covered in the research.

Chapter 3 - Materials & Methods

Materials & Methods discusses all the technology, research and development methods as well as time management and planning.

Chapter 4 - Design & Implementation

Design & Implementation goes through the entire design of the project and how it was all constructed together to become a final product.

Chapter 5 - Results & Analysis

Results & Analysis tells how the final product was thoroughly tested by selected Participants who tested the game.

Chapter 6 - Conclusion

The Conclusion summaries the project, possible next steps for it, what was learned from the project and addresses the objectives outlined in the Introduction.

Chapter 2

Literature Review

This Literature Review will focus on AI companionship in Serious Games. AI will be broken down by its different uses and how it can work as a companion. Then Serious Games will be described, discussed and then analysed on how they can work with AI.

2.1 AI Overview

Ever since the first computer games there has been Artificial Intelligence (AI) used in some aspects. Where it be a chess game where the opponent would be the computer, maze generation in Pacman or even the classic RPG, First Queen. Games are drowned in many different uses of AI technologies. AI is often used for NPCs to make them interactive, follow the player and objectives or even to assist or attack the player with the use of path finding algorithms.

At a 1956 academic conference, John McCarthy was credited with coining the term "artificial intelligence." The conference proposal was stated by McCarthy as follows: "The study is to continue on the basis of the hypothesis that any facet of learning or any other attribute of intelligence can in principle be so clearly specified that a machine can be constructed to replicate it. It will be tried to figure out how to make robots understand language, create abstractions and concepts, solve issues currently left to people, and develop. McCarthy may have coined the term but the idea of AI is awarded to the father of modern computer science, Alan Turing with his paper 'Computing Machinery and Intelligence'. (Skinner, G. and Walmsley, T., 2019). Turing proposed the question, "Can machines think?". This investigation is regarded as AI's inception. Please write me a sonnet on the subject of the Forth Bridge for the "Please write me a sonnet on the subject

of the Forth Bridge" question of the Imitation Game, widely known as the Turing Test. and the computer responded with "Count me out on this one. I never could write poetry." The computer's response demonstrates that it is not programmed with poetry knowledge and is intelligent enough to understand that it cannot. Even though it seems outdated and far-fetched, this was just the beginning (Turing, A.M., 1950).

AI is utilized in games, from maze and world generation, NPCs, chat bots, enemies and path-finding. AI relies on re-dominant paradigms/algorithms. These include Reinforcement Learning, Deep Learning, Deep Reinforcement Learning, Montre Carlo Tree Search, and Evolutionary Algorithms (Risi, S. and Preuss, M., 2020).

Although technological features of embodiment can effectively support interpersonal relationships, it is unclear if this conclusion holds true for AI designed with social objectives, such as AI companions. Because AI companions can be useful for people looking for company, human social perceptions of an AI companion may be especially important for producing a socially relevant interaction. In order to address the significance of social and relational perceptions of an AI companion, this study investigates the social presence and warmth of an AI companion, which are essential in developing companionship and relationships. (Merrill Jr, K., Kim, J. and Collins, C., 2022).

2.2 AI Companionship

AI companionship is quite popular in games today. AI companions can be in the form of characters that can follow the player and assist them throughout the game's gameplay world. Essentially, companions are game characters that accompany the player throughout a longer part of the gameplay, complementing their character or skill set and serving as part of the narrative and experience (Bouquet, E., Makela, V. and Schmidt, A., 2021). In some games there are sections where the player can even play as the companion themselves to get into areas to unlock items or doors that the player cannot, e.g. dangerous or small/cramped areas. Players can also develop relationships with these companions and even begin to care about them, especially if they make an impact and are unfortunately killed off. Many companions throughout games have become cultural icons such as Ellie from The Last of Us, Dogmeat from Fallout 3, Daxter from Jak & Daxter, Clank from Ratchet & Clank and so on. So much so some of these companions get their own dedicated game or at the very least a DLC package.

AI companionship can be seen throughout games and robotics. These

can be in the form of social bots that can interact with humans and users just as another human would. “A social robot is a physically embodied entity immersed in a complex, dynamic, and social environment, being capable to perceive and understand others, including humans, and sufficiently empowered to behave in a manner conducive to its own goals and those of its community by engaging in social interactions with them.” (Paiva, A., 2022).

There are numerous artificial companions. From smartphones, smart watches, voice assistant devices and more. Artificial companions include Amazon’s Alexa, Apple’s Siri, Google’s Assistant and Microsoft’s Cortana. Artificial companions that take the shape of a physical artefact are much more uncommon yet still exist. Artificial companions typically have the following general structure: They have a software-based user interface that serves as a conduit for their functions between themselves and the user, allowing the user to interact with the system verbally. This can be done with a computer programme, a smartphone app, or voice commands given to a speaker or other device that has ”intelligent” capabilities. Instead of being processed by the device itself, the spoken instructions or queries are sent to data servers over the internet. When a command reaches a server, speech recognition algorithms set up by AI systems are activated, questions and commands are processed, and the answers are sent back to the device. The result then manifests as a particular action or as a spoken response (Hepp, A., 2020).

While it is still early days, robot companions can be a great benefit to people as they can lift objects, assemble furniture or other household items. (Dario, P., Verschure, P.F., Prescott, T., Cheng, G., Sandini, G., Cingolani, R., Dillmann, R., Floreano, D., Leroux, C., MacNeil, S. and Roelfsema, P., 2011) in their paper ‘Robot Companions for Citizens’ discusses the idea that they could imagine robot companions being able to help an old bricklayer who is still productive and willing to work. This shows that companions can be of great help to those in need of it.

The foundation of the Robot Companions idea is a thorough understanding of the state-of-the-art in robotics research throughout the world. Robotics research has evolved to a very high degree in several nations, including Japan, Armerica, China, Korea, India, Australia and Europe (Bekey, G.A., Ambrose, R., Kumar, V., Sanderson, A.C., Wilcox, B., Zheng, Y.F., Yuh, J.K. and Lavery, D., 2008), (Siciliano, B., Khatib, O. and Kröger, T. eds., 2008). Robots operate with accuracy, speed, and dexterity in structured environments. They are capable of many different functions, like efficiently producing goods, performing surgery, exploring Mars and acting as prosthetics for people. However, robots are only moderately capable

of effectively and safely interacting with humans, autonomously doing activities of any complexity, or experiencing and exploring unfamiliar places, let alone blending in with the complex, everyday social world of humans (Dario, P., Verschure, P.F., Prescott, T., Cheng, G., Sandini, G., Cingolani, R., Dillmann, R., Floreano, D., Leroux, C., MacNeil, S. and Roelfsema, P., 2011).

2.3 AI Technologies for companionship

2.3.1 Machine Learning & Chat bots

According to Arthur Samuel in 1959, machine learning (ML) is the branch of computer science that "gives computers the ability to learn without being explicitly programmed." ML explores the study and construction of algorithms that can learn from and make predictions on data. ML is used in a variety of computing applications when it is challenging or impractical to create and programme explicit methods with high performance. Email filtering, network intrusion detection, optical character recognition, learning to rank, and computer vision are a few examples. (Ongsulee, P., 2017).

A popular application of AI and ML is chat bots, which are frequently employed on websites run by businesses like travel, phone, and broadband network providers to help clients with basic problems. A chat bot is an instant messaging account that can offer services through instant messaging systems with the goal of effectively conversing with users. A chat bot is a quick, less confusing web and mobile application that requires no installation packages and is simple to set up. These packages are simple to distribute and administer. Chat bots are totally different from human accounts as they do not have any online status or last seen timestamps, nor initiate the conversations and calls with any other accounts. (Rahman, A.M., Al Mamun, A. and Islam, A., 2017).

The basic components of a modern chat bot design may generally be categorised into Knowledge, Response Generation, Text Processing, and Machine Learning Models, which are typically developed via Neural Networks. A chat bot can be categorised as an open or close domain. The chat bot's domain is the range of knowledge it has, such as travel, phone, technical help, and fundamental human knowledge. Researchers have found that while open domain chat bots are still challenging to construct and have been yielding a decent proportion of false positive findings, close domain chat bots are simpler to build and are already producing good results. Retrieval and generative approaches are the two fundamental ways that chat

bots create responses. There are many versions of these two approaches, but their fundamental procedures have not changed much. Retrieval is the process of choosing the best output from a list of candidates, whereas generative is the flexible output production based on an input sequence and trained classifiers. Real numbers in vector space can indicate a semantic relationship between words within a certain vocabulary. This is known as word embedding or vector representation of words. Neural networks in particular, which are used in modern chat bot architecture, can be considered as a key all-around technology that is used to prepare and process input as well as produce output (Lokman, A.S. and Ameedeen, M.A., 2018).

With the development of AI, it is now possible to establish social and emotional connections with artificial beings, notably with so-called social chatbots. Users' affective, social, and relationship expectations may all be impacted by their interactions with social chatbots. It's critical to comprehend how these interactions grow and how they might impact users and their social setting as social chatbots increasingly assume the function of social companions. Media sources claim that people may develop long-lasting relationships with chatbots that have great impactful value. However, there is a lack of information regarding how social relationships between people and social chatbots begin, grow, and affect users' larger social contexts .Given that social chatbots are anticipated to become more common in the future, there is a critical knowledge gap in this area at the moment (Skjuve, M., Følstad, A., Fostervold, K.I. and Brandtzaeg, P.B., 2021).

2.3.2 AI - Path-finding

Path-finding generally refers to finding the shortest route between two endpoints which are commonly referred to as the start node and the goal node with various paths of linking nodes between the endpoints/nodes. For many years, researchers have studied path-finding in video games. It is likely the most well-known but annoying game AI issue in the gaming industry. Before the A* algorithm emerged as a provably optimal path-finding solution, other search methods, including Dijkstra's algorithm, bread first search algorithm, and depth first search algorithm, were developed to solve the shortest path problem. Since its creation, it has drawn the interest of thousands of scholars who have worked hard on it. Path-finding is just one of the numerous issues that can be solved using the general search algorithm A*. The most promising unexplored location (node) that the A* has not already seen is repeatedly examined. The objective is to finish exploring a node and once A* has surveyed neighbouring nodes for more exploration (Cui, X. and Shi, H., 2011).

Another well-liked method for AI path-finding in 3D environments is navigational mesh (NavMesh). A NavMesh is a collection of convex polygons that represents a 3D environment's "walkable" surface. AI entities in the game environment can use this straightforward, incredibly intuitive floor plan for path-finding (Cui, X. and Shi, H., 2011). NavMesh can also be used for NPCs to follow players. This works by having the player as the goal node so the NPC will always try to find the player and follow them to be by their side as their companion.

2.4 Companions in Games

The majority of technical systems are controlled by humans. Systems of various types are utilised in a wide range of diverse application areas, where Companion-technology can improve the functionality and user-friendliness of the system. The field of robotics is a significant area of study and development that is concerned with helping human users. Particularly intriguing for the achievement of Companion-characteristics are areas of application where robots interact with humans directly. These fields include anything from smart toys with limited social contact, to service robots that carry out duties in the home or with the elderly, to systems in health care that assist those with impairments or in rehabilitation (Biundo, S., Höller, D., Schattenberg, B. and Bercher, P., 2016).

With games expanding to many different genres and themes, players' companions do not necessarily have to be human. They can vary from animals such as dogs and horses, fantasy creatures and of course robots too. Having companions in games especially if they are open world or have large sandbox type levels can make the player feel more welcome in the game's world since the companion is an 'inhabitant' of the world itself. (Emmerich, K., Ring, P. and Masuch, M., 2018). The integration of an effective companion into a game is challenging and time consuming, but it also presents great potential. Companions can act as tutors and goal-givers, provide an emotional element to the game and create a certain mood by showing the player how to feel in a particular circumstance (Pinchbeck, D., 2009). A companion may play a significant role in the game's narrative without necessarily contributing to gameplay (Pinchbeck, D., 2009). For example, a character may have a significant role in cutscenes and dialogues without participating in the gameplay. On the other hand, a companion can play a prominent in-game role without having any diegetic significance by having a significant influence on gameplay (for example, by being a proactive fellow soldier) (Emmerich, K., Ring, P. and Masuch, M., 2018).

In this context, (Aarseth, E., 2012) distinguishes between bots, which can lack in personality or individual traits or appear to be shallow characters, which have unique traits like names and roles but don't change or adapt to game events and deep characters, which exhibit elaborate personalities. The ability of a companion to communicate is the final relevant characteristic. If a game character is supposed to be able to speak (for example, because it is supposed to be a human instead of an animal), using natural language can help make the character more believable. Additionally, conversation content is important; contextualised conversational responses are preferred (Lankoski, P. and Björk, S., 2007). This indicates that statements and responses should not be overused and should always be relevant to the situation at hand. As a result, there is a connection to the difficulties of general companion behaviour that were previously discussed, such as in terms of awareness and player models. (Dignum, F., Westra, J., van Doesburg, W.A. and Harbers, M., 2009) states that a companion should be able to interact with other NPCs in the gameworld in addition to the player character in order to further enhance the sense of sociality.

2.5 Serious Games

SG are games designed to serve a purpose other than entertainment. Serious gaming is being employed in various fields, including education, health care, marketing and other industries. Since serious games are entertaining, engaging, and immersive, they have the edge over other learning modalities (Poos, J.M., van den Bosch, K. and Janssen, C.P., 2017). SG can be used to augment or assist training efforts. When training scenarios can get too sophisticated, SG might be a better approach. The usefulness of a training scenario may be boosted due to the diversity of the scenario, and skills learned through serious games can be transferred to real-world situations. (Caserman P., Cornel M., Dieter M., Gobel S. 2018).

This process of developing and discarding theories and techniques may be used in developing and assessing SG. The area of serious games can sometimes suffer from its interdisciplinary nature because of conflicting definitions, assessment techniques, and multiple conceptualizations. On the other hand, serious games research has the ability to profit from the historical precedents established by its converging fields Wilkinson P. (2016).

SG can be developed for several platforms, such as board games and digital games that can gain inspiration from entertainment games. The first modification of an entertainment game into a SG is credited to the US Marines. Marine DOOM (1996) was designed by the US Marines and was

based on the DOOM Game Engine. Due to increasingly limited time and opportunity for actual training exercises, Marine DOOM was utilised as a tool to strengthen military thinking and decision-making abilities (Plass, J.L., Homer, B.D. and Kinzer, C.K., 2015).

Game based learning (GBL) is a concept focused on a learning process that uses a specific approach to games as the primary educational tool for generating and developing abilities. Effective GBL focuses on players' emotions, attitudes, and beliefs and how the game environment's design influences learners' effective state through effective engagement. It also looks at how effect influences and is influenced by cognitive, motivational, social, and cultural learning components. This consideration of these dynamic components of the learning process is how game designers deliberately create the learning experience. It is often overlooked when other learning environments are designed (Kirriemuir, J. and McFarlane, A., 2004).

GBL is applied in various areas, including schools, work departments, healthcare, rehabilitation, training, and other industries. There are a few unfavourable characteristics of GBL that appear to be resurfacing regularly. In comparison to competitive video games, games that focus on GBL are overly basic (Kirriemuir, J., and McFarlane, A., 2004), which tend to include repeated chores that make players feel monotonous and essentially feel like they are 'working.' Another argument made by (Kirriemuir, J., and McFarlane, A., 2004) is that the target audience becomes aware that it is being forced to 'learn' making them feel they are being taught in a condescending way.

2.6 Research Utilisation

From the research it is seen that in terms of a game the techniques and technologies used for the AI implementation are focused on Chat bot systems to allow for interactions and dialogue with the player. These systems include general conversation with one option that leads to a dialogue tree which allows the player to ask the game's AI companion a question such as "How are you?". The companion then answers this question and then asks the player "And you?", the player then gets to say another response based on how they are, and lastly, once that player says that response, the companion then says a closing statement. The Chat bot systems are also used to allow the companion to inspect various 'Artifacts' (A collectable system in the game) throughout the game's levels. These inspections work to enable the companion to be a 'Tour Guide' throughout the game's levels. These interactions allow the player to ask their companion about the

specific Artifact and learn information that the Artifact holds as the companion is able to 'scan' in and read its story. The game's Artifacts are from not only the game's lore and world, but from history that uses historic artefacts with some minor information changes to fit with the game. With this implemented it allows for an interactive learning experience in terms of SG that bleeds into heritage studies. Accompanied by the Chat bot systems is a path-finding system which allows the companion to make their way around the game's levels and follow the player. With these techniques and technologies implemented they work together to support the player along with an 'Attack Mode' to help assist the player in the game's combat. This discussion will be described in a lot more detail in terms or how it all functions in the Design and Implementation chapter.

Chapter 3

Materials & Methods

In this chapter the material and software used to create and host the Moonbeam Chat bot API and the WebGL game with Unity implementation will be outlined with clear reasons for usage and an explanation of purpose. The methods used to gather the research on the topic will be discussed. The IDEs used to write scripts. The Version Control System GitHub will be described in the way it helped pack up the development side. Next, the development methodology that was followed for the development. Lastly, how the survey forms were designed and structured to relate to the research and gameplay and how they were sent to testers to study and analyse their responses to the game.

Materials

3.1 Moonbeam Chat bot API

The interpreted programming language, Python (version 3.7.13) was used to build Moonbeam's API. For the API to be setup it required quite a bit of work and a number of different packages along with a cloud service provider. To make the Moonbeam API, the library Flask with JavaScript was used to create the Webpage along with HTML and CSS. The Machine Learning Chat bot system is also a Python library called ChatterBot. While Flask and ChatterBot are the website's functionality, it are hosted through an EC2 instance on AWS using Ngnix (all terms explained below).

3.1.1 Python, Flask & ChatterBot

Python is a popular language that is designed for multiple uses such as webpages, console apps, desktop apps, and of course AI (especially regarding machine learning). Python's commonly used library for webpages is Flask, which essentially creates an environment for a webpage with JavaScript, HTML and CSS attached to run it.

3.1.1.1 JavaScript, HTML and CSS

JavaScript is one of the most used programming languages as it is used extensively throughout the web. On the web, JavaScript works with HTML and CSS to structure and design webpages. JavaScipt is the main component as it controls the connections the webpages have to other webpages and servers (APIs).

3.1.1.2 ChatterBot

A Python library called ChatterBot enables the creation of automated answers to user input. ChatterBot generates several types of responses using a variety of machine learning methods. This facilitates the development of chat bots and the automation of user interactions (ChatterBot 2022).

3.1.2 Amazon Web Services

AWS is a cloud platform that offers a variety of services. Some of these services include EC2, Elastic Beanstalk, Simple Storage Service Bucket and CloudFront. Deploying to AWS can be as simple as uploading source code files. AWS has its own CLI which, generally works better for deployment. Uploading files can overload the service. Should a significant update need to be deployed, previous files would first need to be deleted before uploading any new files. With AWS's CLI, command modifiers such as '–delete' can be used during deployment. This modifier deletes any files and deploys the updated ones with ease.

3.1.3 Virtual Machines/Elastic Cloud Compute

VMs are virtual emulations of computer environments. For example, VMs allow having a Linux Operating System (OS) accessible from the CLI of a Windows computer with no installation needed. From the Windows side, the connection is generally made in a PowerShell or GIT Bash CLI. AWS'

EC2 service handles VMs as instances. EC2 provides the service to have many VMs with different operating systems (Redhat, 2022).

3.1.4 NGINX

NGINX is open source software that began as a web server built for maximum performance and stability and dependability. It can be used for web serving, load balancing, caching, media streaming, and other functions. (NGINX, 2022).

3.2 The Celestial Beyonds Game

The Unity Game Engine (version 2020.3.34f1) was utilised for the development environment of the game. Unity is a very impressive and powerful tool with a great UI design. Unity is great for indie games meaning it suits to be an engine one person can manage to make their own game. On top of the last statement game companies do use it in teams to create impressive games. Unity allowed for a focus on functionality but also for the visual aspect, with models and assets being able to be 'dragged' in to a scene in the engine. Unity is integrated with the programming language C# which allows users to write custom scripts that control what their game does and how it all functions. Like Python, C# is a general-purpose language that uses the Pascal Case style and runs from the .NET framework created by Microsoft. Unity surely utilises the language as it has base classes such 'MonoBehaviour'. MonoBehaviour includes many methods that Unity needs to run games efficiently. These methods include Awake, Start, Update and FixedUpdate.

3.2.1 Model & Asset Creation

To design the game's characters, structure, world objects and enemies, 3Ds Max (2022) was the tool used to create these models. 3Ds Max is a very robust software that has many functions and it is a very professional tool. Being new to modelling as I only started in the first semester of this Masters, I could navigate 3Ds Max efficiently due to its good UI design and layout. Using 3Ds Max was great as it let my creative side flow to create the game's models without damping it by a hard-to-use system. Along with the models created solely for the game, 3rd party assets and models were gathered for Unity's Asset Store, Sketchfab and Textures.com.

3.2.2 Animations

For animating characters, the built-in Unity animator and Adobe's Mixamo was used to bring the characters and enemies to life.

3.3 IDEs

The IDEs used to write the project's scripts where all from JetBrains. JetBrains has a wide range of powerful IDEs for many purposes that guarantee good code quality. JetBrains IDE for Unity and C# Rider, their IDE for Python, PyCharm and lastly their version of Go's IDE is called GoLand.

JetBrains Rider version 2022.2.1
JetBrains PyCharm version 2022.2
JetBrains GoLand version 2022.2.1

3.4 Hosting the Game

From the early days of planning it was decided to have this game to be WebGL based to allow it to be widely accessible to players where ever they might be and not have to download the game or even run the game from Unity. Initially for the hosting of the game it was planned to have it on Itch.io. Itch.io is a webpage where users can play games mostly made by indie developers. For the Test Demo Itch.io worked just fine, but when the game was finished it ended up being slightly too big to upload to Itch.io. As a result, the game was compressed and hosted on Game Jolt as they allowed for bigger games. Game Jolt is very similar to Itch.io and even has an integrated social media and communities.

All the above sections will be discussed in far more detail in the Design & Implementation Chapter.

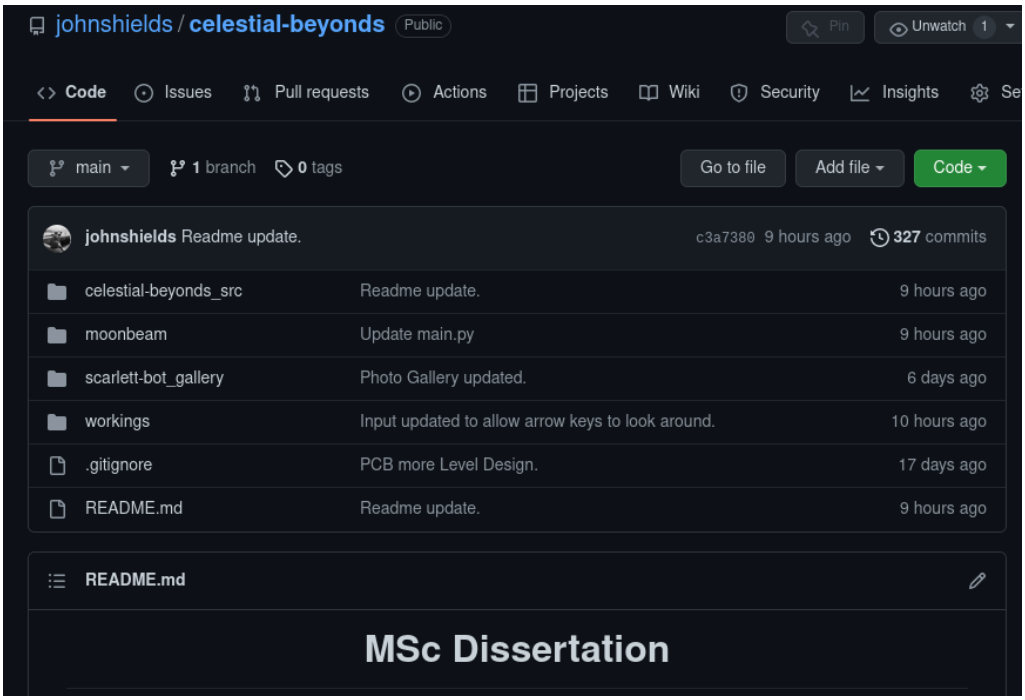
3.5 Version Control

3.5.1 GitHub

GitHub is a Version Control Software that stores the entire code history of projects in what they call 'Repositories' (FB), with READMEs and Wikis

in Markdown for documentation. Repositories can be used for individual usage, and they can add collaborators to commit (contribute) to the repository. For team-based usage, an organisation can be created that would contain individual repositories that all team members can commit to the repository. It is an amazing tool. With GitHub, if something went wrong with a project, for example, a bug or a sudden loss of files, they can all be recovered from the repository, making it easier and faster to fix problems. It gives users the opportunity to show off work their users have done to future employers without having to send ZIP files or screenshots. GitHub was widely used in this project, from the repository to the projects board, Wikis and the issues.

Figure 3.1: The Celestial Beyonds - Repository



3.6 Agile

Agile project management and software development is an iterative approach that enables teams to be flexible and adaptable. This method allows for versatility and, depending on the project, allows for delivering a valuable product to consumers faster and without upfront requirements. Agile employs the Scrum framework, which involves sprints that are fixed-length work iterations. Each sprint has four ceremonies that provide structure

(Atlassian, 2022).

The Four Ceremonies of Scrum

- Sprint Planning
- Sprint Review
- Sprint Retrospective
- Daily Scrum

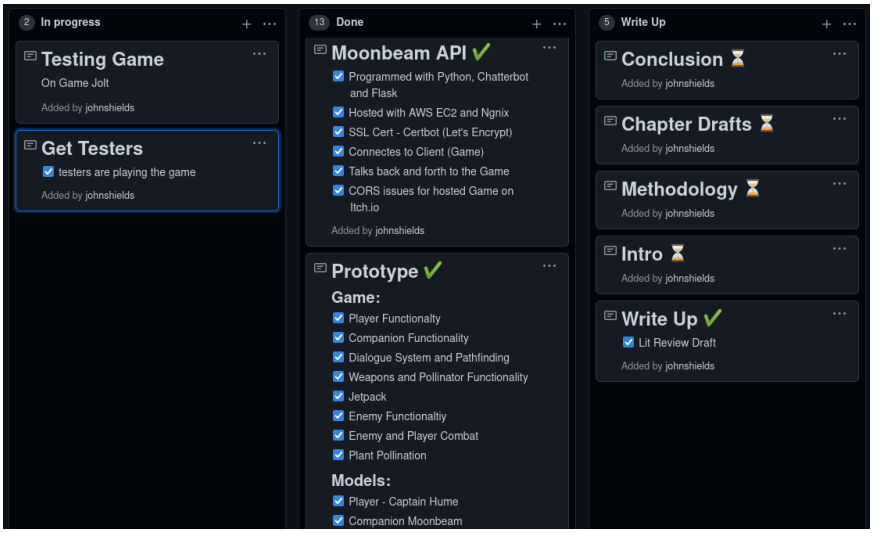
3.6.1 Personal Scrum

Personal Scrum is an Agile methodology that employs scrum practices for one-person projects; through observation of what work has been fulfilled, adaptation of work that requires refactoring, incremental elaboration for planning, prioritising and sizing of the essential work with using allocated time-frames to improve personal productivity. The project's development followed Personal Scrum, which allowed to have time in the months of the project dedicated to it. These times mainly consisted of a review the previous work, plans and setups for implementations to be done the next. Also, if the implementations were either finished or in part, the project's supervisor was informed of the progress, any problems that occurred, and what was next to be done.

3.6.2 Project & Time Management

To have a structured schedule for the project, this required a tracking software with a scrum board. Softwares such as Jira and YouTrack were considered, which seemed a bit over the top for a one-person project. GitHub's repositories have a section titled 'Projects' (FB). Here a work board for the repository can be created to organise the work into cards. GitHub issues can be referenced to allow a card to keep track of a particular issue. GitHub's Projects was very convenient, especially with it being in the project's repository.

Figure 3.2: GitHub Projects



3.7 Research Methods & Strategy

The topic AI companionship was research by reading and analysing, articles, papers, books, websites and other forms of information to bring together the knowledge required to study and analyse this topic and implement it effectively into the game.

As for testing, participants who have various backgrounds were recruited as the topic does not require participants with specific knowledge to test and give feedback on the game. As mentioned before, testing was done in two phases. One with a test demo of the game and one with the final product. It was planned to have an early test to make sure the work was going in the right direction and to know if it needed to be better improved while finishing the game. Having the early tests allowed the game's development to hone what it was initially trying to do and improve it for a satisfying delivery of the final product.

These tests consist of 3 survey forms. The first was for the Test Demo, which was a mix of research and gameplay-based questions. The reasoning for the two areas, is that in the game, Moonbeam is a key part of the gameplay. It had to be seen what worked and did not from both sides. Having the two areas in the test, it allowed the two to be improved on to harmonise with each other and work efficiently.

The other two tests were for the final game. The first was solely based on the research aspect and the second was on the gameplay. The first test was designed to see if Moonbeam was indeed a supportive AI companion in the game. This test mainly asked questions about how the participants resonated with Moonbeam, how they think of the interaction with him, and how was his relationship with Captain Hume. These questions where designed to answer if the research question was a beneficial study, to see if it was worth implementing into a game and if it was a good idea to research it more in the future. The second form was to see how participants found the game itself. The purpose of this was to see if it was worth picking up the idea again and further development and expanding The Celestial Beyonds world in the future.

Chapter 4

Design & Implementation

4.1 Moonbeam Chat bot API

To develop an AI and ML chat bot the WebGL game could access and as mentioned in the previous chapter, the Moonbeam API was programmed in Python using the libraries Flask and ChatterBot. Flask is used as a web framework environment to allow the API to be a fully functional webpage. In the API Flask works by initialising the web page with all its required files. These files include main.py, wgsi.py the templates index.html and home.html along with the static files middle_man.js and style.css. Also with these files there is a data directory that holds the training data for the chat bot. The data is in the form of 4 JSON files and these files are based on general conversation and the levels/planets in the game. These files include; general.json, trappist.json, pcb.json and kepler.json. Please view the table below for a brief explanation of said files before the API functionality is discussed in more detail.

Table 4.1: Moonbeam API Required Files

main.py	Main functionality for the API and Chat bot.
wgsi.py	For running the API.
index.html + home.html	Structure of the Webpage.
middle_man.js	Connection to main.py's Flask operations.
style.css	The style sheet for the Webpage.
general.json	Training Data General Conversation.
trappist.json	Training Data for TRAPPIST-1.
pcb.json	Training Data for Proxima Centuri B.
kepler.json	Training Data for Kepler-186f.

4.1.1 Moonbeam's Inner Workings

4.1.1.1 Chat bot Initialisation

When the API runs the very first element in main.py that is set up after the required libraries for the API is the chat bot with a SQL Storage Adapter and a ChatterBot Logic Adapter that runs an algorithm for Moonbeam's responses using the 'BestMatch' option (FB).

Figure 4.1: Moonbeam ChatBot Initialisation

```
11  # Set up Moonbeam as a Chatbot.
12  mb_bot = ChatBot(
13      'Moonbeam', storage_adapter='chatterbot.storage.SQLStorageAdapter',
14      logic_adapters=['chatterbot.logic.BestMatch']
15  )
```

4.1.1.2 Chat bot Training

Next, the Bot is trained with the general and planets' data by opening the JSON files and using the ChatterBot trainer function to input the data into the Bot (FB). The Bot is initialised first as it would affect performance on the webpage if that were the first element to run since files and data longing is required to train the Bot.

Figure 4.2: Loading & Training of Data

```
17  # Train the bot.
18  print('[INFO] Training bot...')
19  trainer = ListTrainer(mb_bot)
20  with open('data/trappist.json') as f:
21      trappist_data = json.load(f)
22
23  with open('data/pcb.json') as f:
24      pcb_data = json.load(f)
25
26  with open('data/kepler.json') as f:
27      kepler_data = json.load(f)
28
29  with open('data/general.json') as f:
30      general_data = json.load(f)
31
32  trainer.train(trappist_data)
33  trainer.train(pcb_data)
34  trainer.train(kepler_data)
35  trainer.train(general_data)
36  print('[INFO] Training complete!')
```

4.1.1.3 Flask App Creation

After the Bot is trained the Flask App is created with the folders; static and templates for the structure, API connection and style of the webpage. Along with the require routes/endpoint for the webpage (e.g. moonbeambot.live/api/chat) (FB).

Figure 4.3: Flask App Creation

```
38  # create a new web app
39  app = Flask(__name__, static_folder="static", template_folder="templates")
40  app.config["DEBUG"] = True
41
42
43  # add root route
44  @app.route("/")
45  def index():
46      return render_template('index.html')
47
48
49  @app.route("/api/chat")
50  def home():
51      # have the home.html on home page
52      return render_template('home.html')
```

4.1.1.4 Chat Function for User Input and Bot's Response

And lastly, the function for using the Bot is set up as a POST HTTP request that has a user input as a WWW form to send to the Bot. Then the Bot gets a response related to the user's input and sends that response to the webpage (Figure 4.4). This response is then retrieved by middle_man.js and displayed on the webpage (Figure 4.5 & 4.6).

Figure 4.4: Chat Function

```
55  @app.route('/api/chat', methods=['POST'])
56  def chat():
57      user_input = request.form['value']
58      response = mb_bot.get_response(user_input)
59      print("User: " f'{str(user_input)}')
60      print("Moonbeam: " f'{str(response)}')
61      return str(response)
```

Figure 4.5: JavaScript code for retrieving Bot's response

```
$("#talk").click(function () {
    let value = jQuery("#user_input").val();
    const userInput = {value: value};
    console.log("User:");
    console.log(userInput);

    $.ajax("/api/chat", {
        type: "POST",
        // take the entered value
        data: userInput,
        success: function (data) {
            JSON.stringify(data)
            $("#bot_response").val(`${data}`);
            console.log("Moonbeam:" + data);
        },
        error: function (error) {
            console.error("failed to respond." + error);
        },
    });
})
```

More info: Takes in user's typed input, sends to the API and retrieves the Bot's response to display it on the webpage.

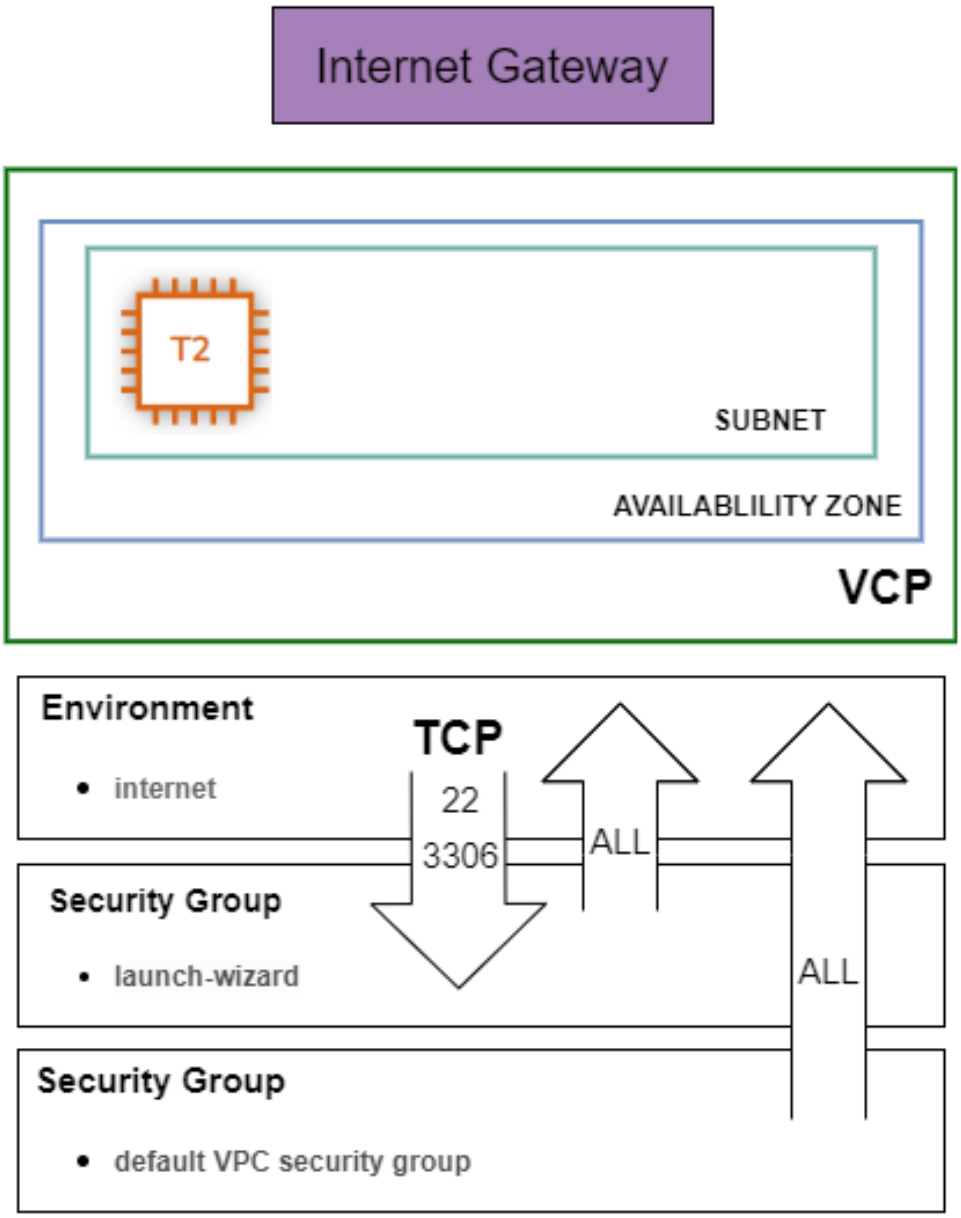
Figure 4.6: User input and Bot's response on the Webpage

More info: Since the Bot is trained for the game the responses are more to related to it rather than real-world general conversation.

4.1.2 Moonbeam Hosting

The Moonbeam API is hosted on an AWS EC2 Ubuntu VM as a server. This provides a VPC with transmission control protocol for ports 22 (SSH login), 8000 (running API), 80 (HTTP) and 443 (HTTPS). This holds the API in a secure environment, which the game on Game Jolt has access to through HTTP requests. NGINX is used with service files on the VM to run the API on the server using the wsgi.py file. This is so the API is always running and does not shut down with an exit of the VM. The figure below shows the entire set-up of the VM/server.

Figure 4.7: EC2 Ubuntu Server



4.1.3 Moonbeam Not Responding

Sometimes when the VM is getting a lot of requests it gets overloaded, crashes and unfortunately goes down for about 1-2 hours. Therefore, in the

game Moonbeam's responses are not available when this happens. This is handled by having Moonbeam let the player know he is not working right at that moment, which is discussed later in this chapter.

4.2 The Celestial Beyonds Game

The game was designed and developed using Unity and 3Ds Max along with various assets from the Asset Store and Sketchfab. This chapter breaks down the game and discusses how it all functions.

4.3 Narrative

The Celestial Beyonds may be the title of the game, but it is also the name of the world the game is set in. The Celestial Beyonds (The Beyonds for short) is set in the distant future, deep in interstellar space and is far, far away from Planet Earth. Throughout The Beyonds there are scattered remains of humanity and its history. One of these remains is NASA's Voyager 1 that has a Golden Record on board, which is a key element in the narrative. The Record contains sounds and images selected to portray the life and culture of Earth. In 1977, NASA sent out the Voyager into space in the hope that if extraterrestrial life exists, they will one day come across and view what the Record holds. The lore behind the creation of The Beyonds is that there are these 'God' like figures called The Celestial Beings who created The Beyonds and all living things in it. The Celestial Beings are inspired by Gods and Goddesses from Greek, Roman and Norse mythology, as well as Kings and Queens from real-world history. The antagonist of the narrative, Aristaeus, (inspired by the Greek God of bee-keeping) a Celestial Being has gone rogue after seeing the planets' plants and nature in The Beyonds wither away and die. Aristaeus was once the leader and founding father of The Apis Rangers (Apis is Latin for bee). The Rangers explore planets and attempt to pollinate their plants throughout The Beyonds. The game's protagonist, Captain Hume, is a Ranger himself and Moonbeam is his loyal AI companion and the narrative follows the duo on their mission to pollinate planets.

4.4 Character and Asset Creation

4.4.1 Captain Hume, Moonbeam & Aristaeus

As mentioned earlier, the software 3Ds Max was used for the characters and assets of the game. The act of pollination in the narrative inspires the look and style of Aristaeus and the Rangers by making them resemble bees. This means that the ranger's spacesuits are mixed with beekeeper suits, bee hives with black and yellow stripes. The Ranger's spaceship also resembles bees. The figures below show the character designs and models of Captain Hume, Moonbeam, Aristaeus and the Anthophila PS-200 (Hume's spaceship).

Figure 4.8: Captain Hume

Figure 4.9: Moonbeam

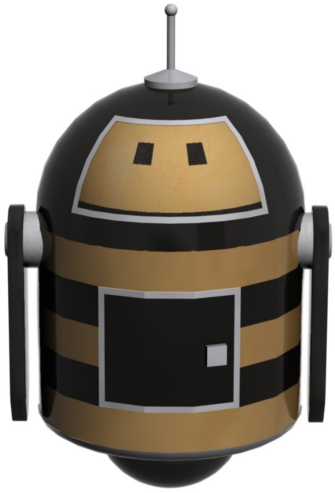


Figure 4.10: Aristaeus with Bee Stinger Staff

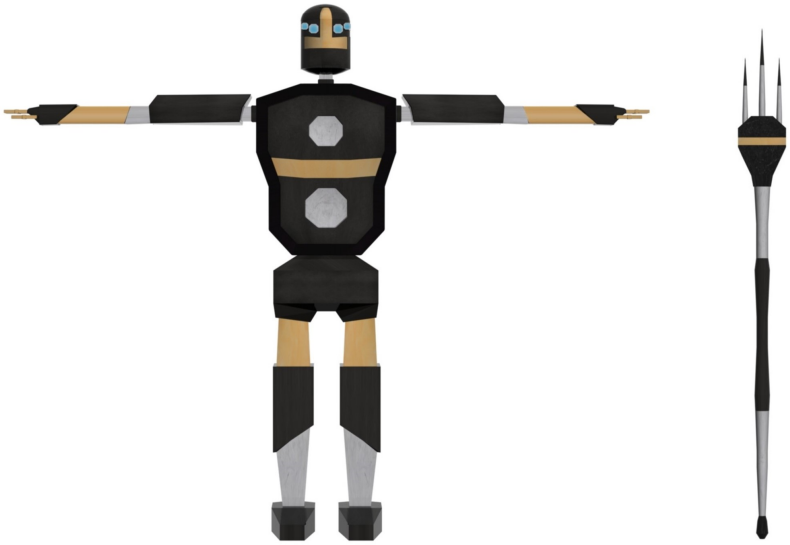


Figure 4.11: Anthophila PS-200

4.4.2 Captain Hume's Arsenal

To survive out in The Beyonds, Hume needs his arsenal, therefore models were made for his weapons. The first figure resembles a bee hive scraper, second his Pollinator Gun (for pollinating plants), the third his Cannon Blaster and following are its upgrades.

Figure 4.12: Scraper

Figure 4.13: The Pollinator f6000

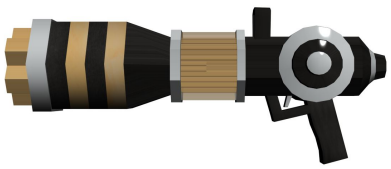


Figure 4.14: The Cannon Blaster

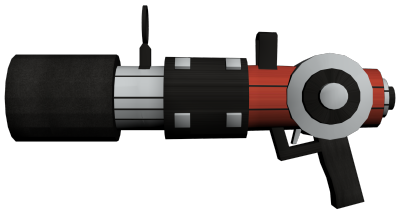


Figure 4.15: The PepperBox Blaster

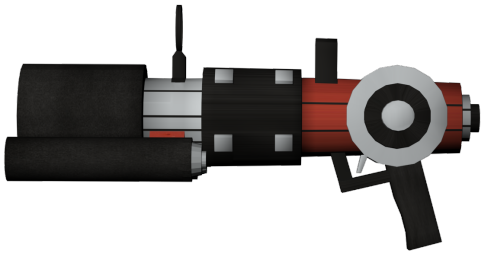
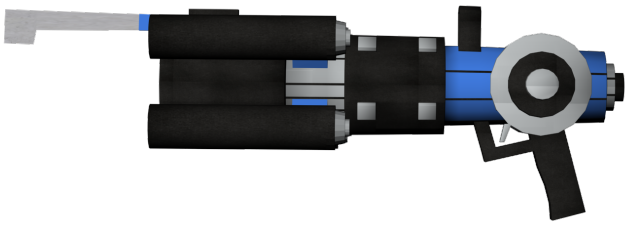


Figure 4.16: The Celestial Defier



4.4.3 Collectables

Around the game's planets there are items that Hume can collect. The first figure is of a Peridot. These are the currency in the game. The second is a Honey Jar, which Hume can collect to regain health. The third figure shows an Ammo Jar which Hume can use to reload his Cannon. The last two figure are crates/chest that bigger Peridots and Honey Jars are stored in.

Figure 4.17: Peridot

Figure 4.18: Honey Jar

Figure 4.19: Ammo Jar

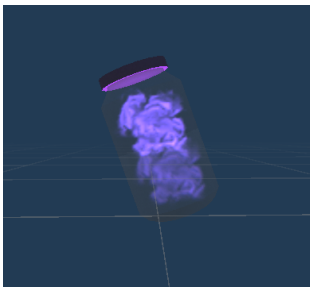


Figure 4.20: Honey Crate and Peridot Chest

4.4.4 Plants

The Plants The Apis Rangers pollinate are mostly flowers. The figures below show the plants I modelled for the game.

Figure 4.21: Plants

4.4.5 Enemies

The Planets in The Beyonds are not a safe place. They are inhabited by creatures of all kinds which can be hostile and attack on site. The figures below show the enemies I modelled for the game.

Figure 4.22: Enemies

4.4.6 Stores

In the game there are two stores where Hume can fill up on Pollen, Ammo and get upgrades. The first figure is The Hive Shack that the Pollen Merchant, Argyle works at. The second is Van Gun's Guns where Viktor Van Gun runs the stand.

Figure 4.23: The Hive Shack and Argyle

Figure 4.24: Van Gun's Gun and Viktor Van Gun

More info: Please note that the hat, the gun on the wall, the radio and ammo boxes, are not modelled by me, they are from other creators on Sketchfab.

4.4.7 Artifacts and Narrative

4.4.7.1 Artifacts

As mentioned above, there are remains of humanity and its history scattered throughout The Beyonds. These remains are known as 'Artifacts'. While

I did get most of the Artifacts from Unity's Asset Store and Sketchfab, I modelled some myself, some where specially made for the game and some where from previous games I developed and designed. The figures below show the Artifacts I modelled.

Figure 4.25: Endian (Celestial Being)

Figure 4.26: The Celestial Being, Arthurus' Excalibur

Figure 4.27: Ballycarbery Castle Ruin

4.4.7.2 Narrative: Voyager 1 and The Golden Record

The next figures shows the model I did for NASA's Voyager and The Golden Record. The Voyager is defiantly one of the hardest models I have done as it has so many parts attached to it.

Figure 4.28: Voyager 1

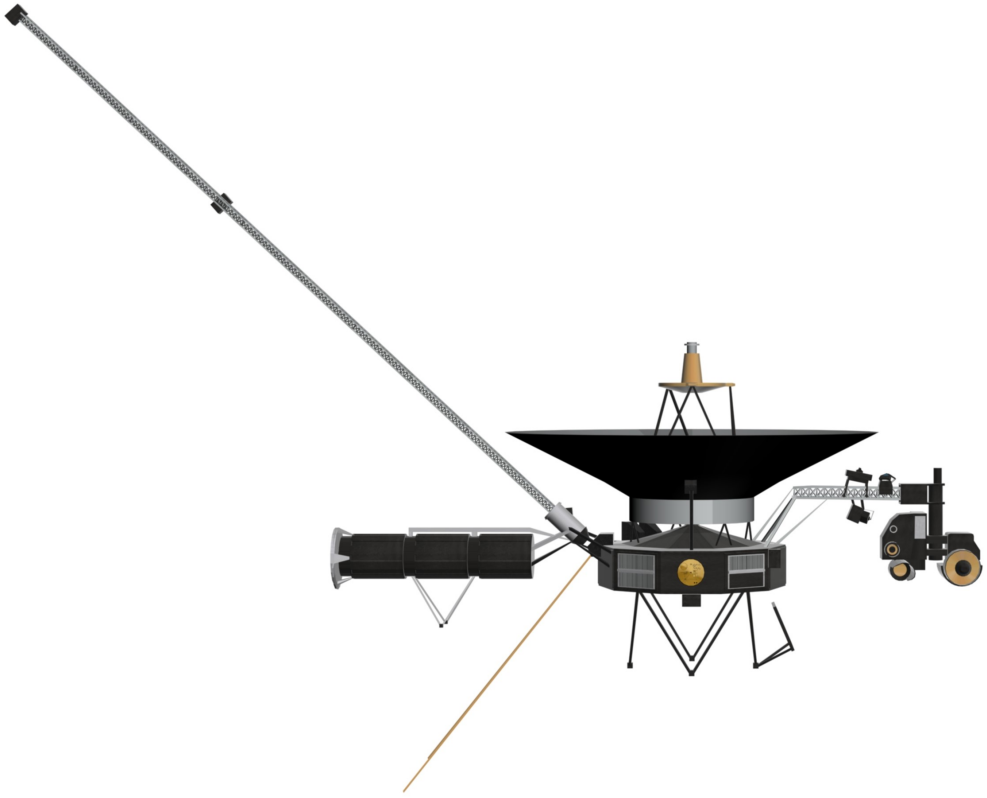
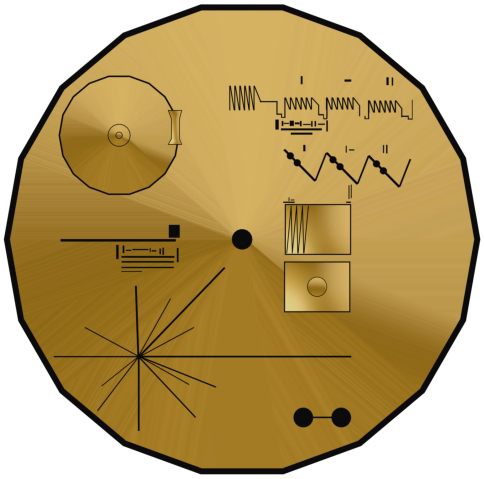


Figure 4.29: The Golden Record



4.4.8 Animations

Unity has a built-in animation system (FB) which was used for a lot of the games enemies, world objects and last, but certainly not least Moonbeam. The other characters, Captain Hume, Aristaeus, Argyle, Viktor Van Gun, and some models for enemies downloaded from Sketchfab required more complex animations. Adobe's tool Mixamo (FB) allows these characters to be rigged easily as it has a built-in character rigger and a wide choice of animations available to download.

Figure 4.30: Robot Bear Unity Animation

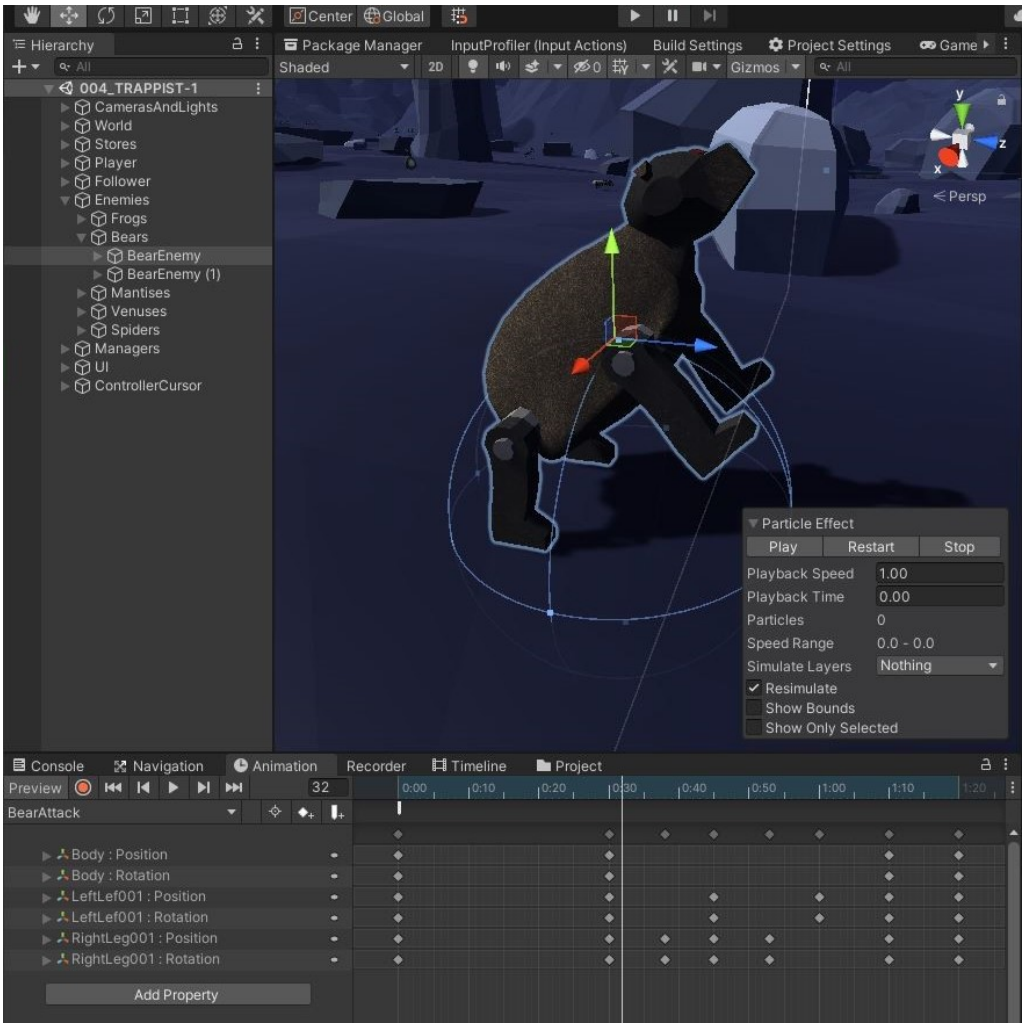
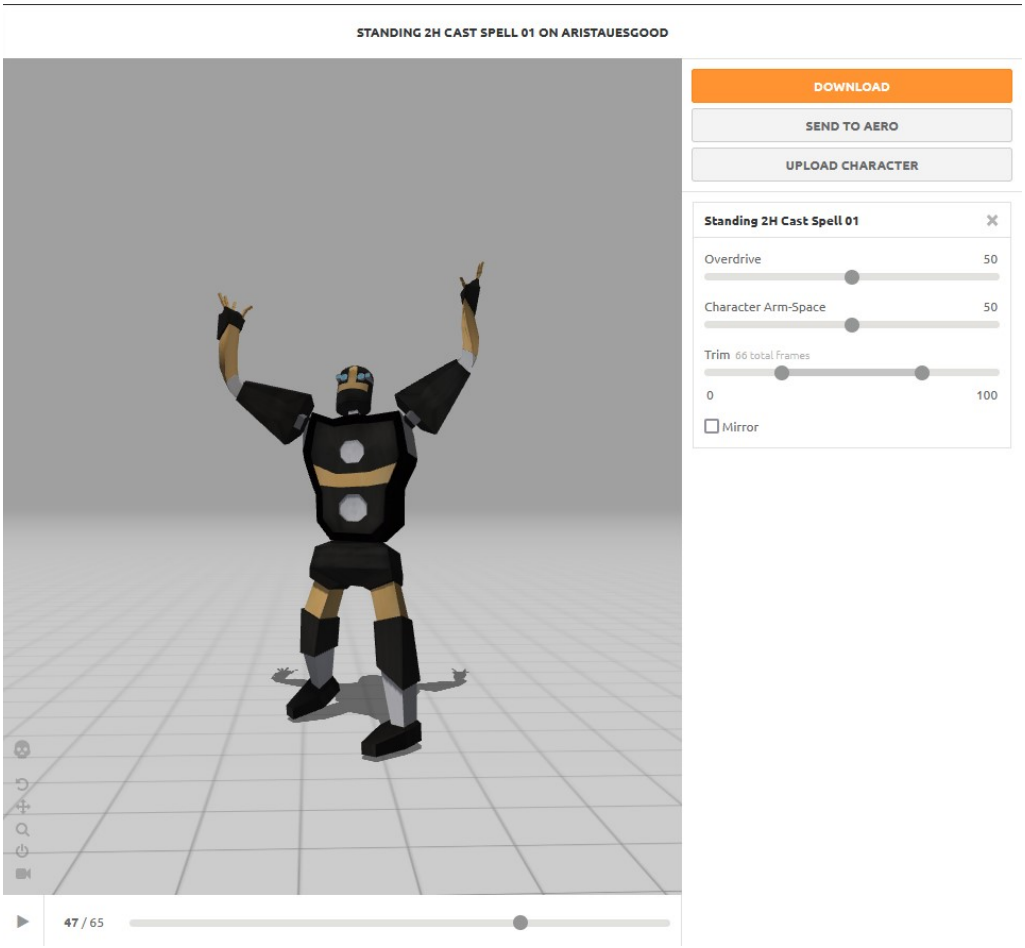


Figure 4.31: Aristaeus Mixamo Animation



4.5 Player Functionality

4.5.1 Low and High Profiles

The very first element of the game was prototyping Captain Hume's actions. In the game the player takes control of Captain Hume. Hume is programmed with a wide range of actions he can preform. These actions are divided into two profiles, low and high. In low profile, Hume can stand idle, walk, run, jump and dodge. In high profile (or armed) Hume can melee attack, shoot his cannon, and idle, walk and run with his weapons. All these animations are controlled by Unity's Animator. In the Animator Hume has 5 layers of animations (Figure 4.32). The animations idle, walk and run are controlled by a Blend Tree (Figure 4.33). This Blend

Tree makes the transitions very smooth and realistic. These layers allow for multiple animations organised and can be altered to override each other should they conflict. The layers also connect all these animations together so Hume can transition freely between them all.

Figure 4.32: Captain Hume's Animator

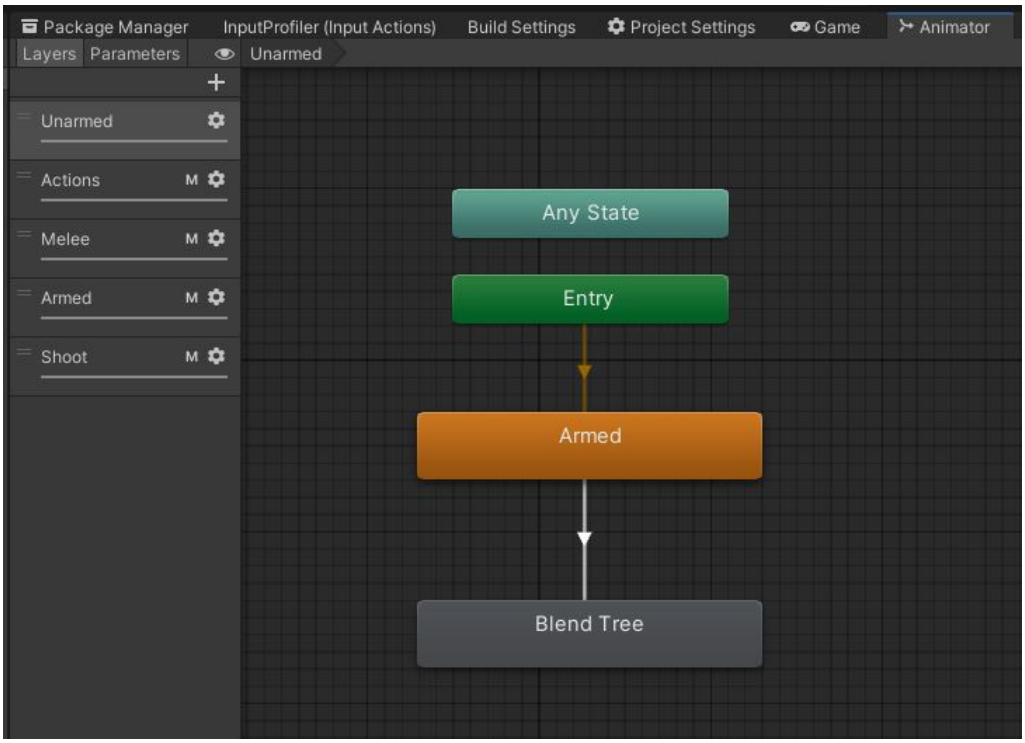
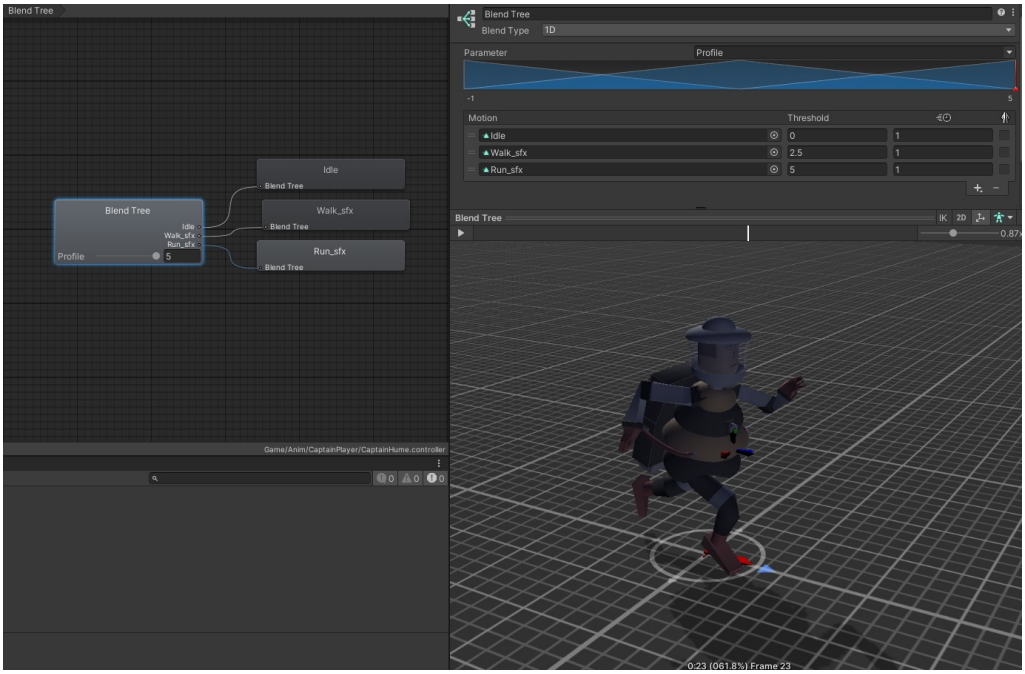


Figure 4.33: Blend Tree



4.5.2 Captain Hume's Profiler Scripts

Hume's actions and Animator are then controlled by the Scripts, CaptainProfiler.cs and CaptainAminAndSound.cs. These Scripts work together to allow the player to control Hume using Unity's Input System (Figure 4.34). The Input System is designed to allow for multiple key bindings to single actions. This is very efficient for when developing a game to support keyboard, mouse and gamepad controls. Hume is programmed to work with these 3 devices, but unfortunately WebGL does not support the new Input System yet, therefore he can only be controlled by a gamepad in Unity's Editor or a PC build of the game. The Input System is accessed in the Scripts and used in methods (Figure 4.35 & 4.36) to control Hume's actions and access other Scripts by a key binding input from the player. In the Scripts there are also 'helper' functions (Figure 4.37) to organise the code and avoid repetition.

Figure 4.34: Input System

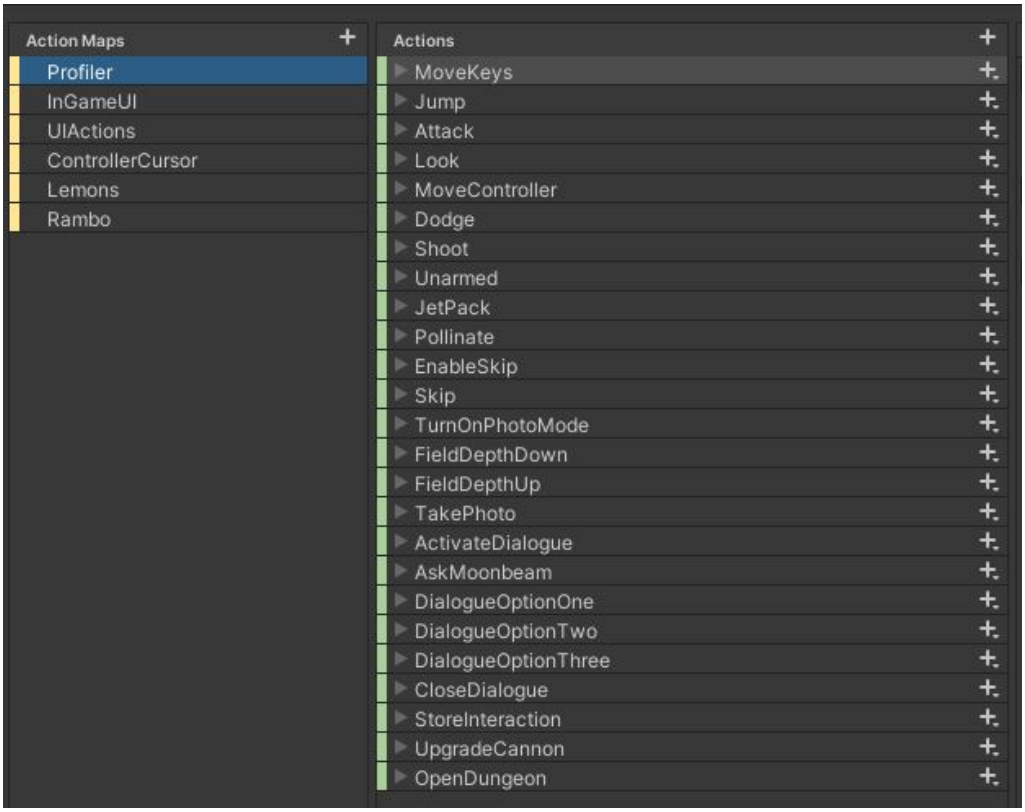


Figure 4.35: Unarmed Method

```
// Disable all weapons
2 usages  johnshields +1 *
private void Unarmed(InputAction.CallbackContext obj)
{
    if (!pauseMenu.GetComponent<InGameMenus>().pausedActive &&
        _mb.GetComponent<MoonbeamDialogue>().convEnd)
    {
        PlayerState( unarmed: true,  armed: false);
        if (_unarmed)
            WeaponSelect( s: false,  c: false,  p: false);
    }
}
```

Figure 4.36: Dodge Method

```
private void Dodge(InputAction.CallbackContext obj)
{
    if (!pauseMenu.GetComponent<InGameMenus>().pausedActive &&
        !GetComponent<CaptainHealth>().capDead)
    {
        if (_actionDone) return;
        AnimWeight( actions: 1f,  shoot: 0f); // alter layer
        _animator.SetTrigger(_dodge);
        StartCoroutine( routine: WaitToDodge());
        _actionDone = true;
        Invoke(nameof(ResetAction), delayAction);
    }
}

 Frequently called  1 usage  new *
private IEnumerator WaitToDodge()
{
    // have hume dodge back his current position
    yield return new WaitForSeconds(.25f);
    _rb.velocity = transform.TransformDirection( x: 0,  y: 0,  z: dodge);
    Invoke(nameof(ResetAnimWeight), delayAction);
}
```

Figure 4.37: WeaponSelect Method

```
// Allow Hume to freely select between weapons
 6 usages  John Shields +1
public void WeaponSelect(bool s, bool c, bool p)
{
    if (!pauseMenu.GetComponent<InGameMenus>().pausedActive)
    {
        _scraper.SetActive(s);
        _pollinator.SetActive(p);
        _cannonObj.SetActive(c);
        UpgradedCannon();
        if (pbUpgrade)
            _pepperBox.SetActive(c);
        if (cdUpgrade)
            _celestialDefier.SetActive(c);
    }
}
```

4.5.2.1 Jetpack

The planets in The Beyonds can be quite large. Therefore it would take a lot of time to walk/run around them during missions. For this Hume is equipped with a Jetpack that allows him to fly around the planets and gets to where he needs to go in a timely and cool fashion. This works by having Hume's RigidBody component controlled and lifted up into to air in the script Jetpack.cc. Once Hume is in the air, the player can control the direction he flies in with the same keys for movement (FB). The Jetpack does run out of fuel, but it can recharge itself when not in use.

Figure 4.38: Jetpack

4.5.2.2 Combat with Enemies

Hume has a Health System in the game. When enemies attack him, it damages his health (FB). If his health goes all the way down, he gets 'terminated' and the player has to restart the level to progress. The Health System is controlled by the CaptainHealth.cs Script. The Enemies are controlled by the NavMesh System in Unity. This allows the Enemies to patrol the planets and avoid objects in their directions, and if Hume is in range the enemies go into attack mode and Hume and Moonbeam have to fend for themselves. On both Hume and Enemies, there is a Particle Effect that resembles blood. This is to notify the player that either Hume or the Enemies are taking damage.

Figure 4.39: PlayerTakeDamage Method

```
// Allow Hume to take damage from Enemies and terminated him if his health goes to 0.
// Accessed in EnemyCombat and SpiderCombat.
3 usages  John Shields *
public void PlayerTakeDamage(int amount)
{
    if (!capDead)
        currentHealth -= amount; // take damage

    // if Cap's Health is below or equal to 0 terminate him.
    if (currentHealth <= 0)
    {
        capDead = true;
        print( message: "Player Terminated!");
        // reset Peridots collected in level.
        if (!peridotsReset)
        {
            peridotsReset = true;
            PlayerMemory.peridots -= Peridots.peridotsCollectedInLvl;
        }
        // Death animation.
        _player.GetComponent<CaptainAnimAndSound>().CapDeath();
        StartCoroutine( routine: GameOverScreen());
    }
}
```

4.5.3 Captain Hume in Attack Mode

Hume can swing his Scraper which does little damage to Enemies, or he can use his Cannon to blast them away. The Scraper damages enemies due to having a BoxCollider Trigger on it to hit off the enemies and call the Script, CaptainCombat.cs (Figure 4.40) to make the enemies take damage. The Cannon works with an Ammo System which can run out by emptying the Cannon's magazine. If the Ammo runs out, the player cannot shoot anymore enemies and has to stock up on ammo from Van Gun's Guns, which will be discussed later in this chapter. The Ammo that shoots out of the Cannon is a Particle Effect (Figure 4.41) that can collide with enemies to take damage from them, which, works much in the same way as the BoxCollider Trigger for the Scraper.

Figure 4.40: CaptainCombat - OnTriggerEnter Method

```
Event function  John Shields +1 *  More...
private void OnTriggerEnter(Collider other)
{
    // Only allow Scraper to hit off Enemies
    if (other.gameObject.CompareTag("Enemy") &&
        _player.GetComponent<CaptainAnimAndSound>().meleeActive &&
        !_player.GetComponent<CaptainAnimAndSound>().endgame)
    {
        // Enemies take damage and play a 'blood' particle effect
        other.gameObject.GetComponent<EnemyProfiler>().TakeDamage(other.gameObject);
        other.gameObject.GetComponent<ParticleSystem>().Play();
    }
```

Figure 4.41: The Celestial Defier - Particle Effect

4.5.4 Pollination System

As mentioned earlier, The Apis Rangers can pollinate plants with their Pollinator f6000's (Figure 4.13) which is controlled by a Pollination System in the game. The Pollinator f6000 works similarly to the Cannon Blaster, but it cannot damage Eenemies. Its only purpose is the pollinate plants. The Pollen is also done by a Particle Effect which hits off dead plants on planets and brings them back to life (Figures 4.42 & 4.43). The pollination works by having the dead plant destroyed and blossoming into a new one with a glowing effect from a Spot Light (Figure 4.44). Each planet pollinated adds

to the Pollination Level in a planet. Once the player has pollinated all the plants on a planet, they are free to move on to the next to do the same (Figure 4.45). Hume can also run out Pollen, which is acquired by Argyle at The Hive Shack, which will also be explained later in this chapter.

Figure 4.42: Dead Plant

Figure 4.43: Plant Pollinated

Figure 4.44: Blossom Method

```
1 usage  John Shields
public void Blossom(int num)
{
    // Blossom Plant
    AudioSource.PlayClipAtPoint(blossomSFX, plantsOG[num].transform.position, volume: 0.5f);
    Destroy(plantsOG[num]);
    plantClones[num].SetActive(true);
    plantClones[num].GetComponentInChildren<Light>().enabled = true;
    // IncreasePollination.
    miniMenu.GetComponent<MiniMenu>().plantsNum += 1;
    _pl.GetComponent<PollinationLevel>().IncreasePollination();
    print( message: "Plant cloned and dummy destroyed: " + num);
}
```

Figure 4.45: IncreasePollination Method

```
1 usage  John Shields *
public void IncreasePollination()
{
    // Increase the pollination lvl
    pollinationPercent += pollenIncrease;
    print( message: "PollinationLevel: " + pollinationPercent);

    if (pollinationPercent == maxPollination)
    {
        // once the pollinationPercent is max the planet is complete
        levelCompleted = true;
        print( message: "Level complete! " + levelCompleted);
        StartCoroutine( routine: LevelCompleteUI());
    }
}
```

4.6 Player Interaction with Moonbeam

Hume would be lost without Moonbeam, not only because he is his best friend, he also acts as his personal assistant, right hand man and his very own tour guide. The duo can talk in general conversation and Hume can ask Moonbeam about the Artifacts scattered throughout The Beyonds through the Dialogue System in the game.

4.6.1 Dialogue System

The Dialogue System works with the Moonbeam Chat bot API, which is controlled by the following Scripts; MoonbeamAPI.cs, DialogueForm.cs and MoonbeamDialogue.cs. With all the responses Moonbeam can say these Scripts are quite complicated as they have to take in everything that Hume says to Moonbeam, send it to the API, retrieve Moonbeam's response, and finally display it in the game. The responses are retrieved by the method, PostRequest in MoonbeamAPI (FB). This method takes in what Hume says which is inputted through the MoonbeamDialogue.cs and DialogueForm.cs Scripts and sends Hume's request to the API. Next, there are checks to see if there is a problem or error with the request, if there is not the API will send a response back to Hume. As mentioned in the Moonbeam Chat bot API section, the API can get overloaded with requests and suffer a down time of about 1-2 hours. This downtime is called a 'Timeout' and if Hume tries to send a request to the API at this time, it results in a 'Request Timeout' and the API cannot respond. To handle this downtime in the game Moonbeam's response is changed to "Sorry, my API is down." to let players know he is not working right at that moment. Hume's was a large range of requests he could ask Moonbeam. He has 36 general requests and 90 requests about Artifacts (30 per planet, 3 per Artifact). These requests are further explained below.

Figure 4.46: MoonbeamAPI - PostRequest

```
Frequently called  1 usage  John Shields *
public IEnumerator PostRequest(string uri)
{
    if (!disabledMoonbeam)
    {
        var form = new WWWForm();
        GetComponent<DialogueForm>().SetUpForm(form);
        // Send POST request.
        using var webRequest = UnityWebRequest.Post(uri, form);
        yield return webRequest.SendWebRequest();
        // if there is a Request error
        if (webRequest.result == UnityWebRequest.Result.ConnectionError ||
            webRequest.result == UnityWebRequest.Result.ProtocolError ||
            webRequest.result == UnityWebRequest.Result.DataProcessingError)
        {
            print( message: "Error getting response: " + webRequest.error);
            _response = "Sorry, there seems to be a screw lose.";
            _mb.GetComponent<MoonbeamDialogue>().response = _response;
            PlayRandomClip( vol: 0.5f);
        }
        // if Moonbeam can indeed respond.
        else
        {
            // response from API.
            print( message: "Moonbeam says: " + webRequest.downloadHandler.text);
            _response = webRequest.downloadHandler.text;
            _mb.GetComponent<MoonbeamDialogue>().response = _response;
            IsItAQuestion(); // checks to see if its a question for Dialogue Trees.
            PlayRandomClip( vol: 0.5f);
        }

        // When server is down.
        if (webRequest.error == "Request timeout")
        {
            print( message: "Error timeout: " + webRequest.error);
            _response = "Sorry, my API is down.";
            _mb.GetComponent<MoonbeamDialogue>().response = _response;
            PlayRandomClip( vol: 0.5f);
        }
    }
}
```

4.6.2 General Dialogue

Hume has 3 options to say to Moonbeam, these options are randomly selected from an array of options when the Dialogue is initiated (Figure 4.47). There for the player can input option 1, 2 or 3, Moonbeam takes in the option and responds with a response related to the option (Figure 4.48). Hume and Moonbeam can also have a small conversation. For example, Hume asks "How are you?", Moonbeam responds "I am good thanks, how are you?" and Hume finishes with saying "I am good too, thanks!" These small conversations are called Dialogue Trees (Figures 4.49 & 4.50).

Figure 4.47: Talk to Moonbeam Prompt

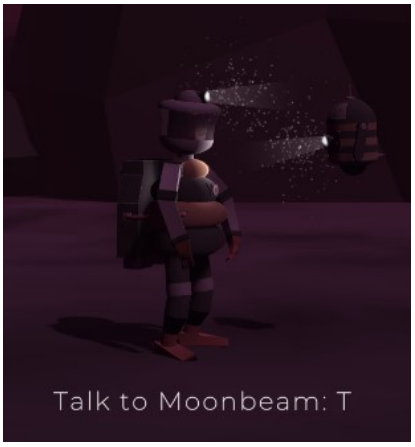


Figure 4.48: Dialogue - No Tree

Figure 4.49: Dialogue Tree 1

Figure 4.50: Dialogue Tree 2

4.6.3 Artifacts Interaction

Hume can ask Moonbeam about the Artifacts they discover. Moonbeam has the ability to scan and analyse the Artifacts to learn about what they are, what there purpose is/was and what there origins are. The Artifacts are inspired from a wide range of sources. These sources are, the game's lore,

monuments, history, ruins, damaged ships, mythology, folk lore, references to other games, films and TV shows. Some examples of these Artifacts are The Statue of Liberty (Figure 4.51), Stonehenge (Figure 4.52), King Arthur's Excalibur (Figure 4.53), A Brotherhood of Steel helmet from the Fallout games (Figure 4.54), The Monolith from the film, 2001: A Space Odyssey (Figure 4.55) and The Dharma Initiative Van from the TV show LOST (Figure 4.56). As previously discussed the Artifacts' name, information and origins is slightly altered to fit with the game's lore. Where this has been done is showing in the figures below after the backslash.

Figure 4.51: Statue of Liberty/Statue of Libertas (Celestial Being)

Figure 4.52: Stonehenge (Built by The Celestial Druids)

Figure 4.53: King Arthur's Excalibur/Arthurus' Excalibur

Figure 4.54: Brotherhood of Steel/Celestial of Steel Helmet

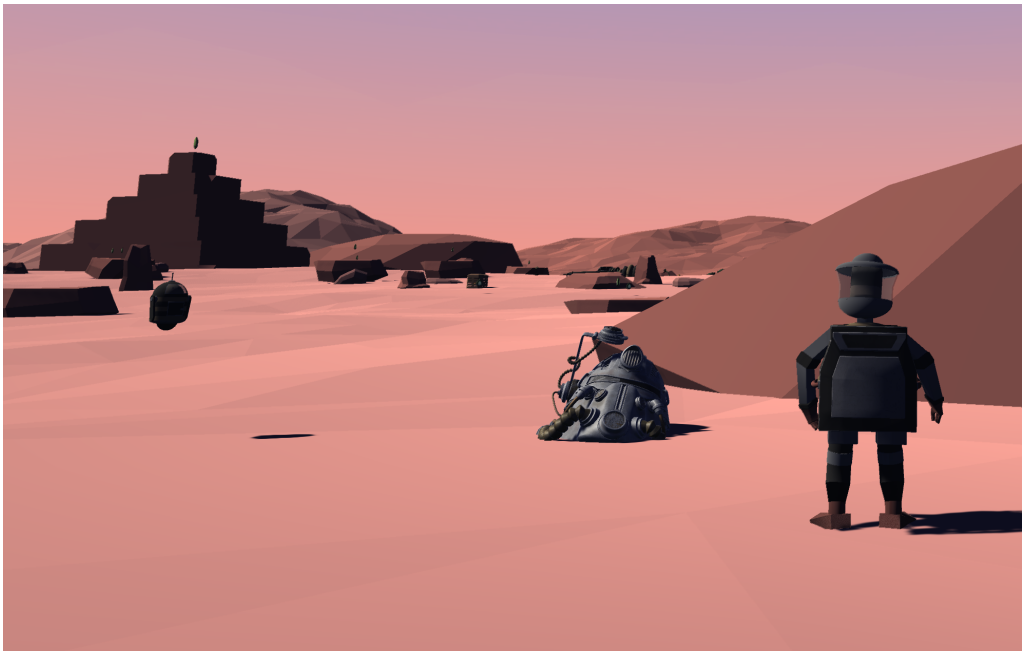


Figure 4.55: Monolith

Figure 4.56: The Dharma Initiative Van

4.6.3.1 Serious Games and Heritage

The Artifacts provide a learning aspect in the game for players as some of them are instantly recognisable. Since the Artifacts are from real-world sources and the game's lore the player gets to learn about them in an engaging and amusing method. Also, due to the Artifacts being from real-world sources it gives players a chance to learn about the Heritage of them and may even interest them to research more information on the real-life artefacts.

4.6.4 Artifact Dialogue

For Hume to ask Moonbeam about an Artifact it works much in the same way as the General Dialogue does, only with information relating to the specific Artifact Hume wants to know more about. Hume gets the chance to ask Moonbeam about an Artifact by entering its collider to trigger the interaction. From there, like the General Dialogue, there are 3 options in the form of questions that Hume can ask Moonbeam using the keys 1, 2, 3. The figures below show the interaction of the Artifact, The Book of Kells which in the real-world comes from Irish history and is an illuminated manuscript of the four gospels of the Christian New Testament.

Figure 4.57: Ask Moonbeam about Artifact Prompt

Figure 4.58: Dialogue Option 1

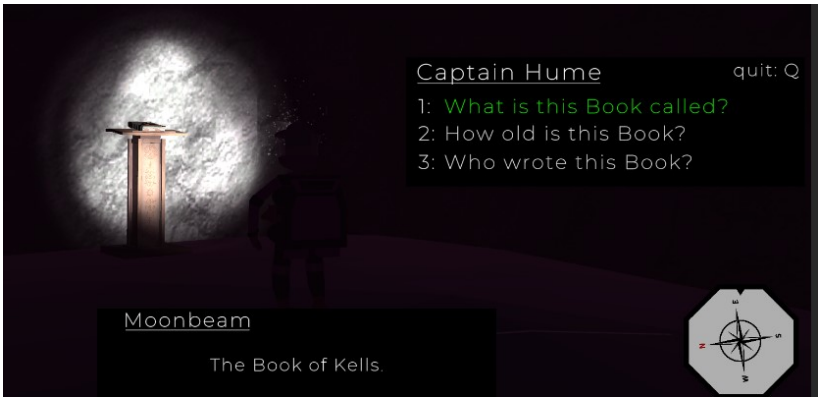


Figure 4.59: Dialogue Option 2

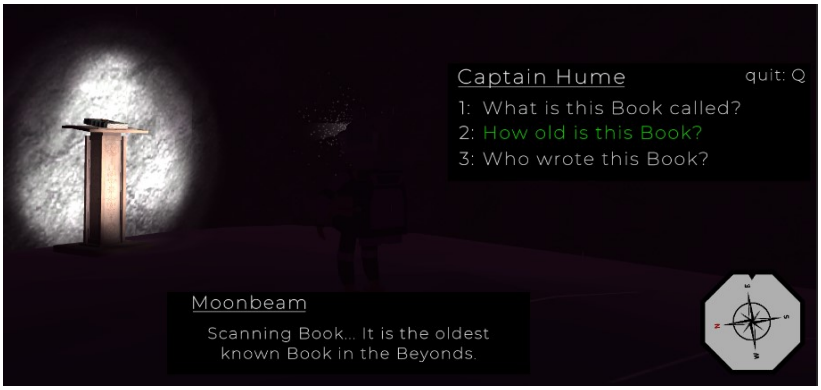
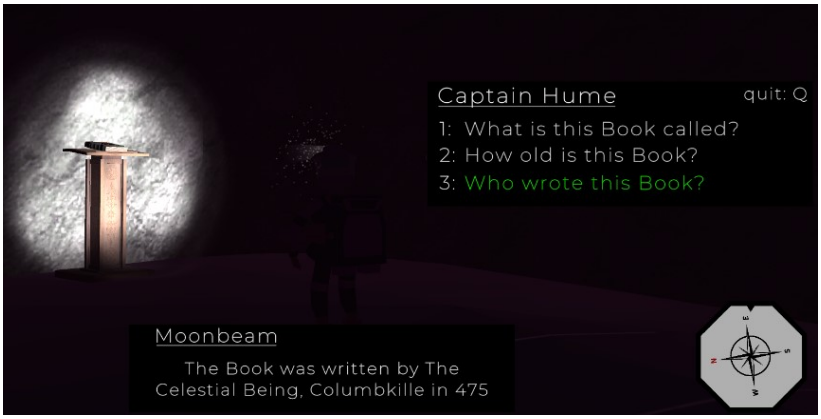


Figure 4.60: Dialogue Option 3



4.6.5 Moonbeam Path-finding and Combat

Moonbeam can support Hume by following him around where ever he might go. Moonbeam can do this as a result of his Path-finding System. That is the same system used for the Enemies, NavMesh (FB).

Figure 4.61: NavMesh

When Hume swings his Scraper or shoots his Cannon Moonbeam's method AttackMode in the Script, MoonbeamProfiler.cs is called. This call makes Moonbeam spin around and shoot out two Particle Effects that resemble moonbeams (FB). These moonbeams work in the same way the Cannon's Particles do for collision with Enemies.

Figure 4.62: Captain Hume and Moonbeam in Combat

4.7 Levels/Planets Design

The game's levels are 3 planets, TRAPPIST-1, Proxima Centuri B and Kepler-186f. While the names of these planets are from real planets their style and atmosphere is not inspired by them. This allowed for a more creative approach to level design. The planets design and gameplay is done in a 'Sandbox' fashion. This means that players have the freedom to explore them until they are content giving the planets an 'open-world' freedom.

4.7.1 TRAPPIST-1

The first planet in the game that Hume and Moonbeam land on is TRAPPIST-1 (FB). This planet works as an introduction to the game for the player. The planet's surface is scattered with rocks, mountains and Artifacts. All the Planets Enemies are the ones I modelled for the game.

Figure 4.63: Planet TRAPPIST-1

4.7.2 Proxima Centuri B

Proxima Centuri B is the second planet in the game. It is a much brighter atmosphere than TRAPPIST-1. It is set on a beach with a sunset in the background. This planet includes sand, rocks, water, a rock pool and various Artifacts. In this planet there are two new Enemies, The Cyclops Diver and the Kraken which, were downloaded from Sketchfab, along with the

ones I modelled. The Cyclops Diver was animated in Mixamo and The Kraken was animated in Unity's animator.

Figure 4.64: Planet Proxima Centuri B

4.7.3 Kepler-186f

The third and final Planet in the game is Kepler-186f. This planet is inspired by the Celtic/Medieval time period and the Viking era. It is the darkest planet of the 3 with a thick fog with Enemies hiding in it. This planet has 3 new Enemies (All from Sketchfab and animated in Mixamo), the Stone Golem, Demogorgon and the Templar Knight Zombie.

Figure 4.65: Planet Kepler-186f

4.7.4 Aristaeus Boss Battle

The final level takes place on Kepler-186f. In this level, Hume and Moonbeam have to defeat Aristaeus which, plays as a Boss Battle (FB). Aristaeus is the toughest and baddest enemy that Hume and Moonbeam have to fight. The battle can be quite a challenge for players so they have to be strategic and hone their skills to defeat Aristaeus. Aristaeus also takes use of the NavMesh system to have him chase Hume and Moonbeam around Kepler-186f and attack the two when he is in close range. Aristaeus can damage Hume as a result of his Bee Stinger Staff having a BoxCollider that triggers when it collides with Hume.

Figure 4.66: Aristaeus Boss Battle

4.7.5 Collectables

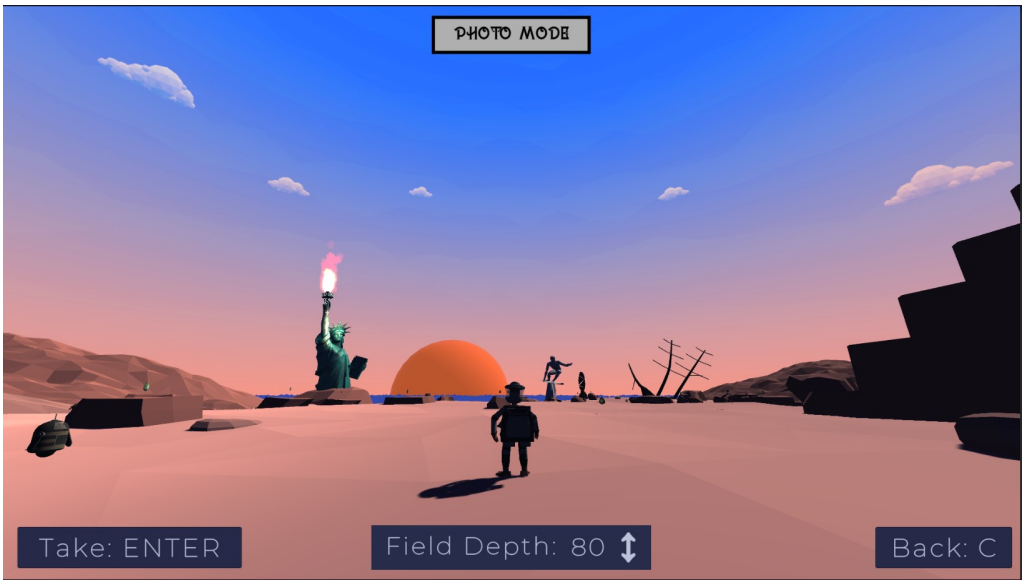
The collectables in the game are Peridots, Honey Jars, Ammo Jars and Artifacts. The player has to collect Peridots to be able to stock up on Pollen, Ammo and purchase Upgrades which, is discussed later in this chapter. The Honey Jars regain Hume's health after taking damage from Enemies. The Ammo Jar are for reloading Hume's Cannon during the Boss Battle to give the player more of an advantage. The player can also find large Peridots that are worth more in Chest and large Honey Jars in Crates to regain more health. These Chests and Crates can be broken open with Hume's Scraper. The Artifacts also work as a collectable as the game's code counts the ones the player has found. All these collectables are displayed in the HUD and MiniMenu witch will be discussed in UI & HUD Design.

4.7.6 Photo Mode

While the was in development it was decided to implemented a PhotoMode to get high quality shots of the game (FB), which is a popular feature in many games today. This was mainly added to get good game screenshots for this paper and to add to the media of the game where it is hosted. It was decided to leave the PhotoMode in the game as it allows players to be creative and get some nice pictures of their gameplay. Player's can access the PhotoMode from the PauseMenu. In order for players to be able to see their photos a Telegram Bot was set up to have the game send

the photos to the bot where they could be downloaded and uploaded to a webpage. The PhotoMode webpage is also online and it was programmed in the programming language Go using the web framework net/http and hosted with Heroku. It is just a basic webpage where players can look at the Photo Mode Gallery.

Figure 4.67: PhotoMode



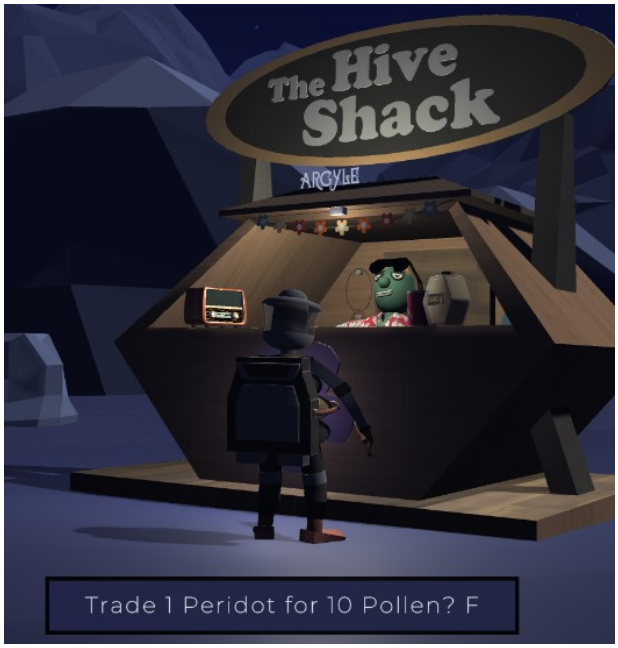
4.8 Trading Systems

The game has two trading systems where the player can stock up on Pollen, Ammo and purchase Upgrades.

4.8.1 The Hive Shack

The Hive Shack maned by Argyle from the Planet Proxima Centuri B sells Hume Pollen for Peridots that he has collected (FB). This works by checking to see if Hume has enough Peridots to buy Pollen. If he does the transaction is made, he acquired Pollen and Peridots get taken away from his collection. If Hume does not have enough Peridots the transaction is declined. Lastly is Hume already has the maximum amount of Pollen he cannot buy anymore.

Figure 4.68: The Hive Shack Trading System



4.8.2 Van Gun's Guns

Van Gun's Guns, maned by Viktor Van Gun from the Planet Rusjan, is where the player can buy Ammo and Upgrades for Hume's Cannon. The trading system works the exact same as it does for The Hive Shack. The player can unlock upgrades when they have terminated all the Enemies on a planet. When they have done so they an option to buy an upgrade becomes available at Van Gun's Guns.

4.9 UI & HUD Design

The UI in the game is the MainMenu, PauseMenu, Hume's HUD, the MiniMenu and the Dialogue Boxes for when talking with Moonbeam. The MainMenu (Figure 4.68) is the first scene players load into the playing the game.

4.9.1 Menus

From the MainMenu players can Launch into the game, Restart (Figure 4.69) it (if they have played through), load a specific Act (Figure 4.70) (if they have completed the game), view the Controls (Figure 4.71), Credits,

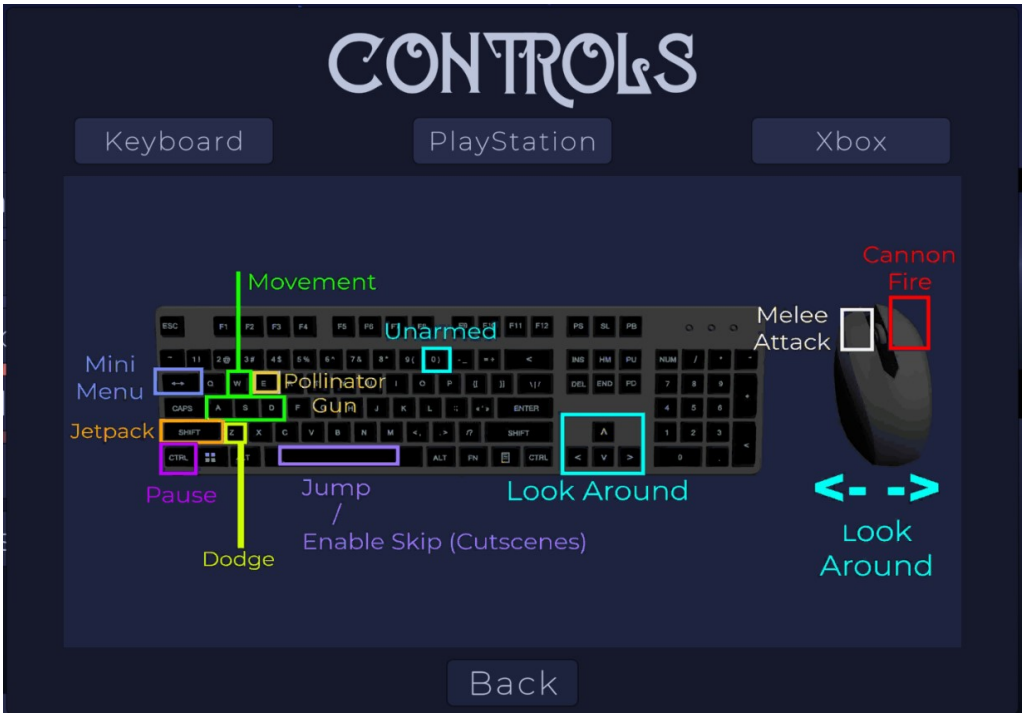
Quit the game and Mute the game's Audio. They can also see what current planet they are on. This is thanks to Unity's PlayerPrefs which saves progress in the game.

Figure 4.69: MainMenu

Figure 4.70: Restart Panel

Figure 4.71: ActLoader Panel

Figure 4.72: Controls Panel



The PauseMenu (FB) is accesses in-game. In the PauseMenu the player can Resume the game, view the Controls, load back to MainMenu, Mute

the Audio and access the PhotoMode.

Figure 4.73: PauseMenu

The MiniMenu is where the player can see what planet they are currently on, see how Planets they have Pollinated, Enemies they have Terminated and how many collectables they have in their inventory (FB).

Figure 4.74: MiniMenu

4.9.2 HUD

The HUD (RB) displays Hume's Health, Jetpack Fuel, Pollen, Ammo, the Pollination Level of a Planet and a Compass for navigation.

Figure 4.75: HUD

4.9.3 Banners

Along with the MiniMenu there are Banners that display if a player has Pollinated all the Plants, Terminated all Enemies, Found all Artifacts, Collected all Peridots, unlocked an Upgrade and lastly a Banner to show when Hume has been Terminated (Figures Below).

Figure 4.76: Max Pollination Banner

Figure 4.77: HUD

Figure 4.78: All Artifacts Found Banner

Figure 4.79: All Peridots Collected Banner

Figure 4.80: Upgrade Banner

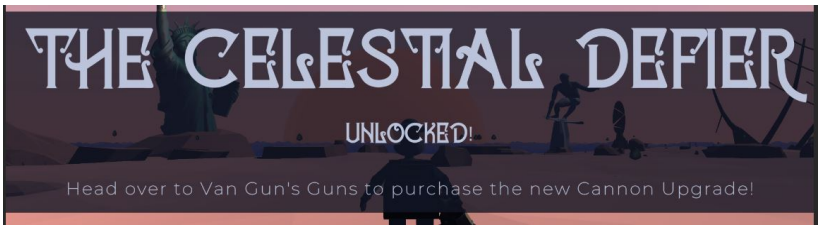


Figure 4.81: Hume Terminated Banner

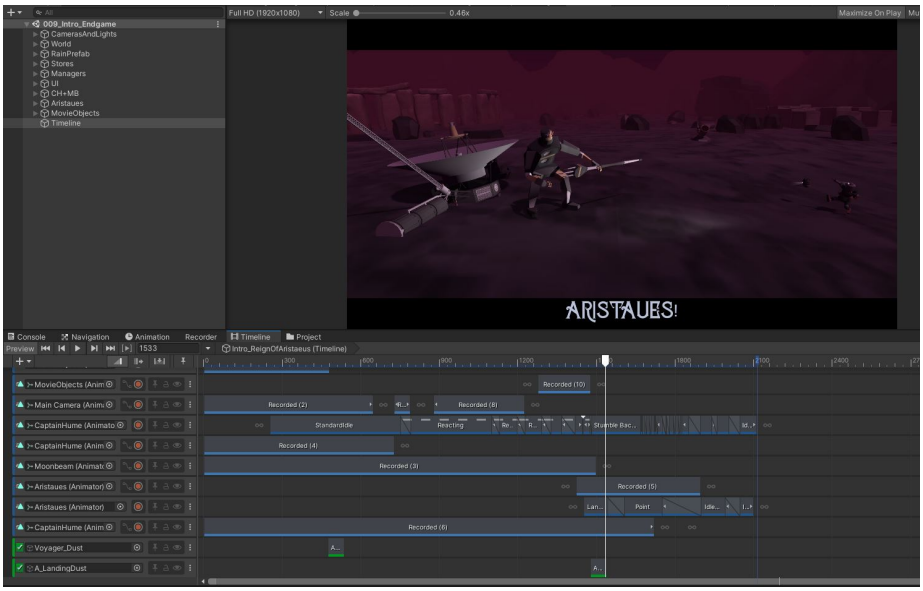
4.10 Audio & Soundtrack

The game's Audio, SFX (Sound Effects) and Soundtrack was sourced from a number of areas. Which include Freesound.org, Pixabay.com, Pond5.com and by me. The sounds from other sources were edited to fit the game and they are used across the game for Enemies, Dialogue and interactions. Linus Johnsson a student from SimVis created Moonbeam's voice. I voiced Aristaeus (with some heavy editing). The game's original score was composed myself using Fruity Loops Studio which, I had experience with before this project. The voices of Argyle and Viktor were done by two good friends of mine, Sam McMillan and Micheal Borodin. One of the tracks from the original score features 23 different messages in different languages. These are the messages of The Golden Record in the narrative. These message come from my friends that I am so lucky to have all over the world.

4.11 Cinematics

The Cinematics in the game are away of telling and showing the player about Hume's and Moonbeam's mission, the travelling between planets, intros to planets and telling the Narrative. These Cinematics where done in Unity's Timeline with PlayableDirectors (FB). This really allowed for a lot of creativity in development and Narrative and was a nice break from scripting.

Figure 4.82: Unity Timeline

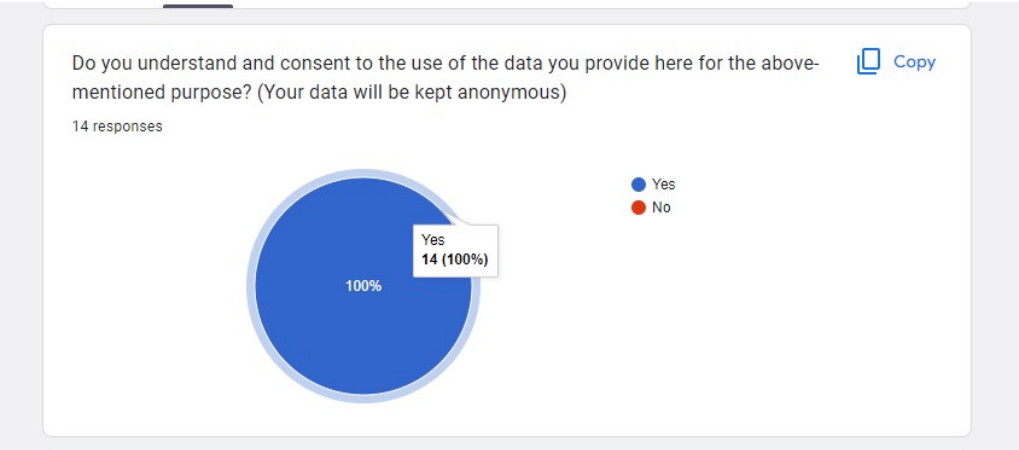


Chapter 5

Results & Analysis

This chapter discuss how the game performed with Participants who play tested it. As said before the testing was done in two phases. One for the Test Demo and one for the Final game. The Participants tested the game and filled out surveys on Google Forms. For the surveys the very first questions asked is a disclaimer to let the Participants know their involvement and that their data with be kept anonymous (FB).

Figure 5.1: Disclaimer Question



5.0.1 Test Demo Results

The Game was made available to Participants on the hosting website Itch.io. On this website Participants could play the game from their browser and not have to download it. The Test Demo allowed to gain feedback and further improve the game before the final version was finished. The figures below show the results of the Test Demo.

Figure 5.2: Test Demo Question 1

Figure 5.3: Test Demo Question 2

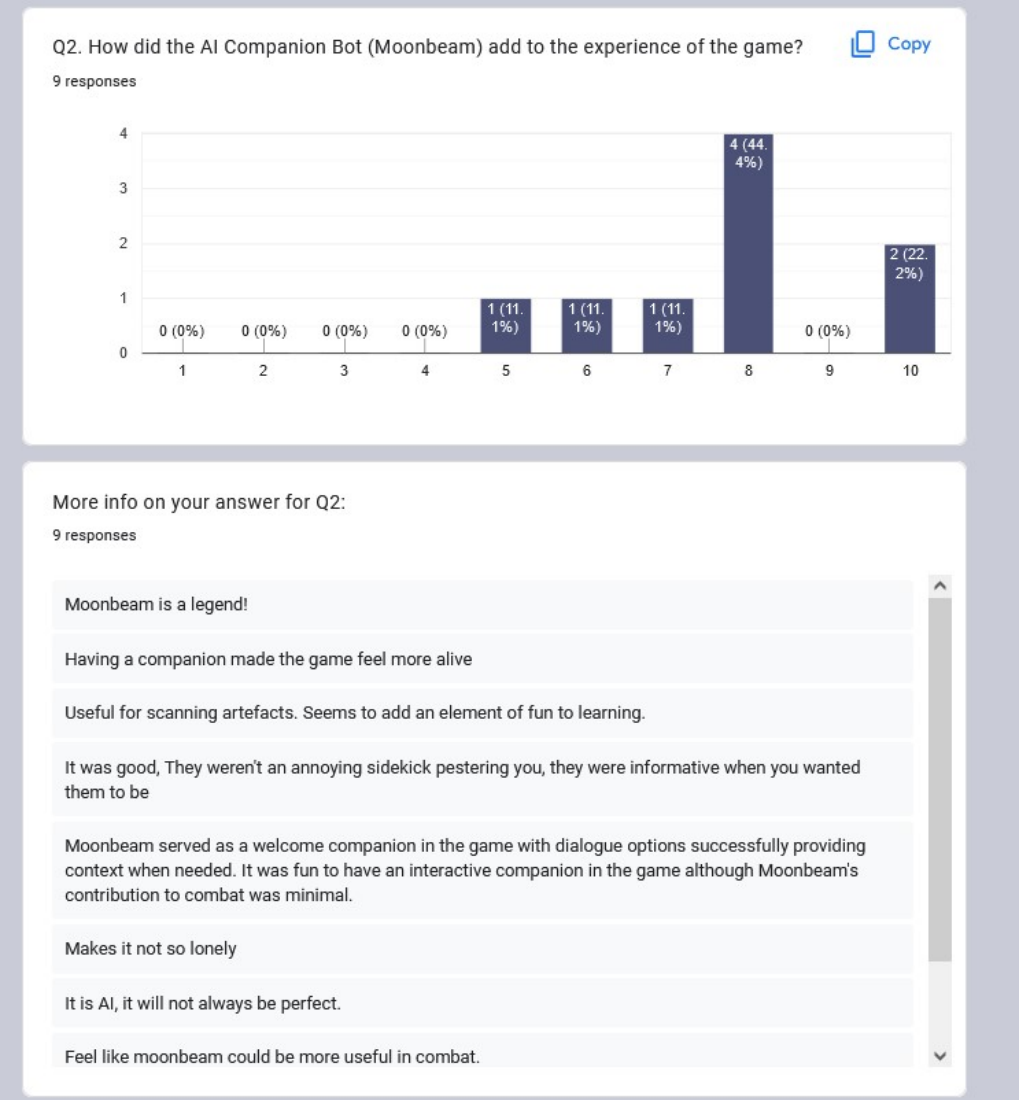


Figure 5.4: Test Demo Question 3

Figure 5.5: Test Demo Question 4

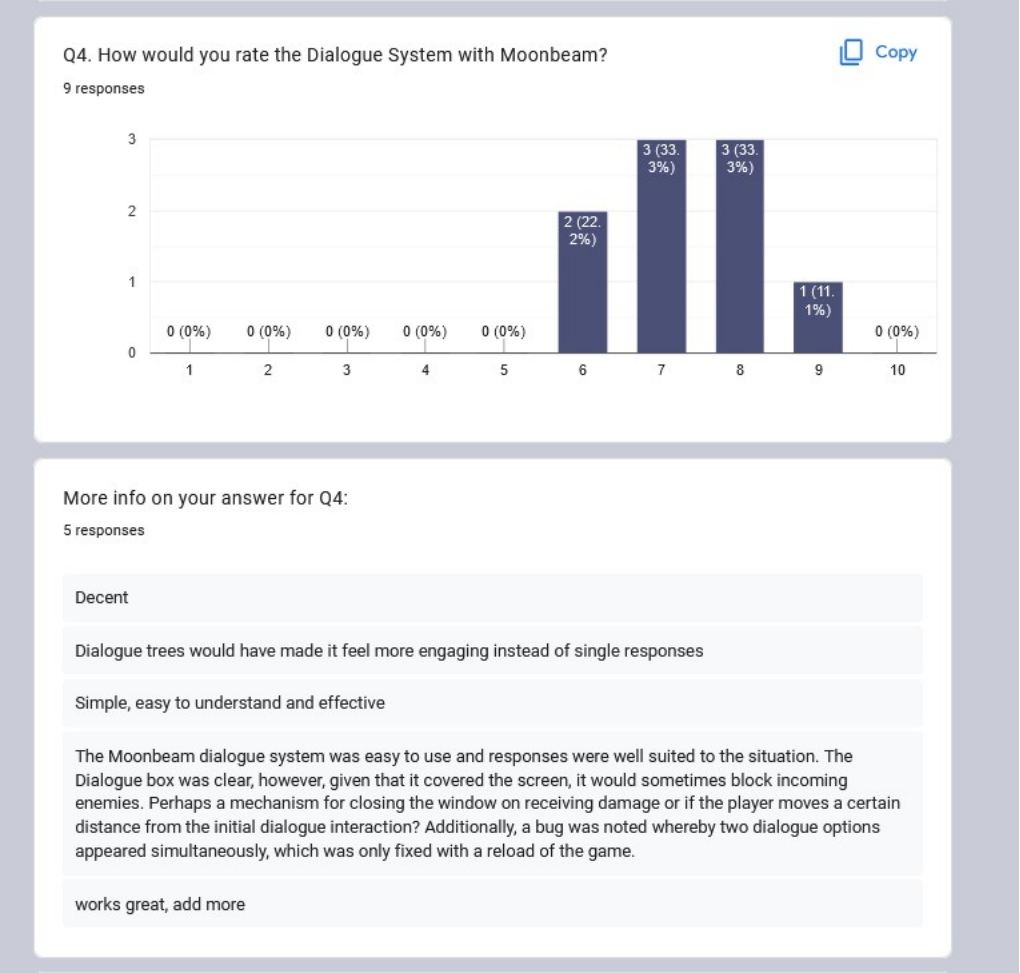


Figure 5.6: Test Demo Question 5

From the results in the surveys it was clear that the work has potential but it did need improvement which was applied after conducting the Test Demo. This paved the further research and development for the rest of the game's implementation.

5.0.2 Final Game Results

For the final game it was made available to Participants on Game Jolt. Game Jolt allowed them to experience the game on their browser in its full capacity. The purpose of the second test was to see if the work had improved from the first and was it effective in terms of AI companionship. The results of this test where a few numbers higher from the first which allowed for a wider collection of data. The figures below show the results of the Final Game Testing.

Figure 5.7: Final Test Question 1

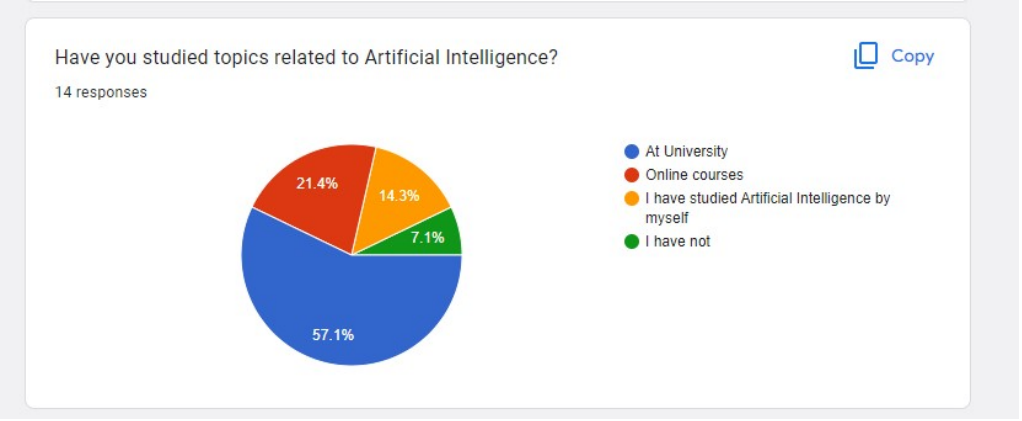


Figure 5.8: Final Test Question 2

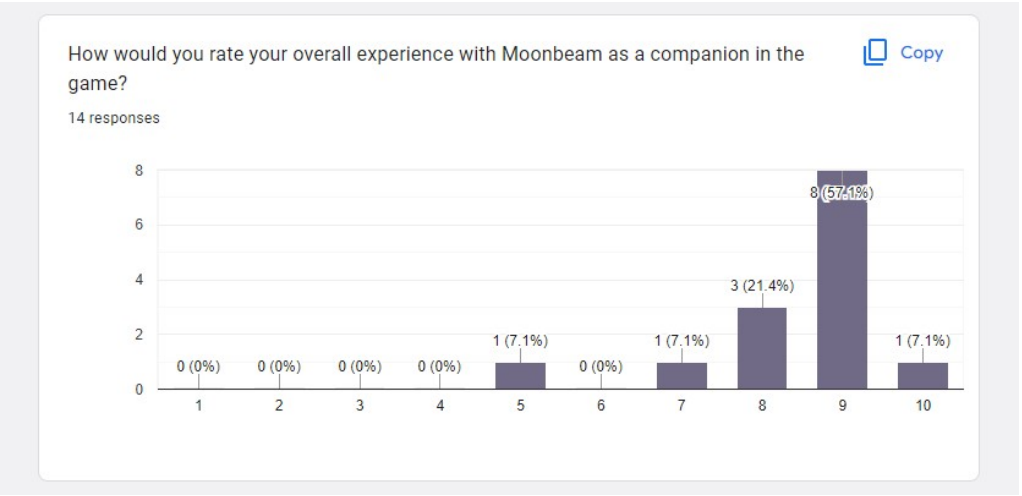


Figure 5.9: Final Test Question 3

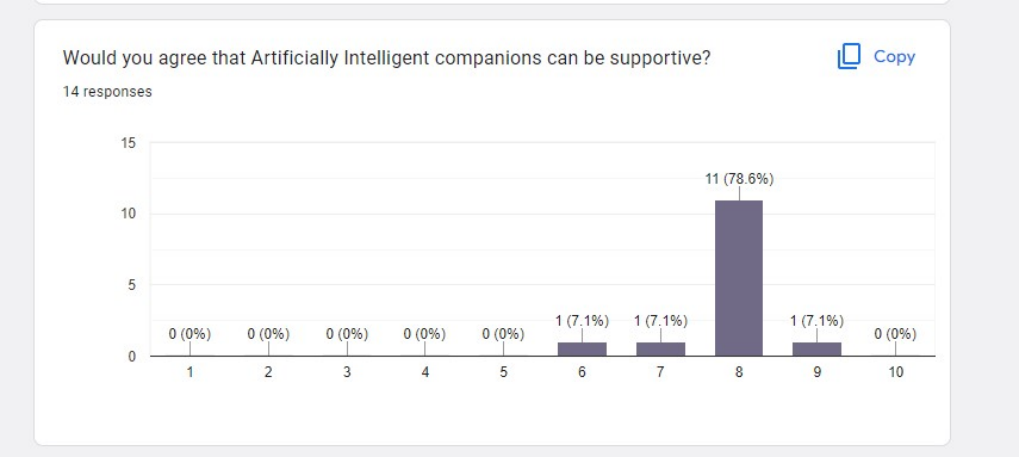


Figure 5.10: Final Test More info on Question 2 & 3

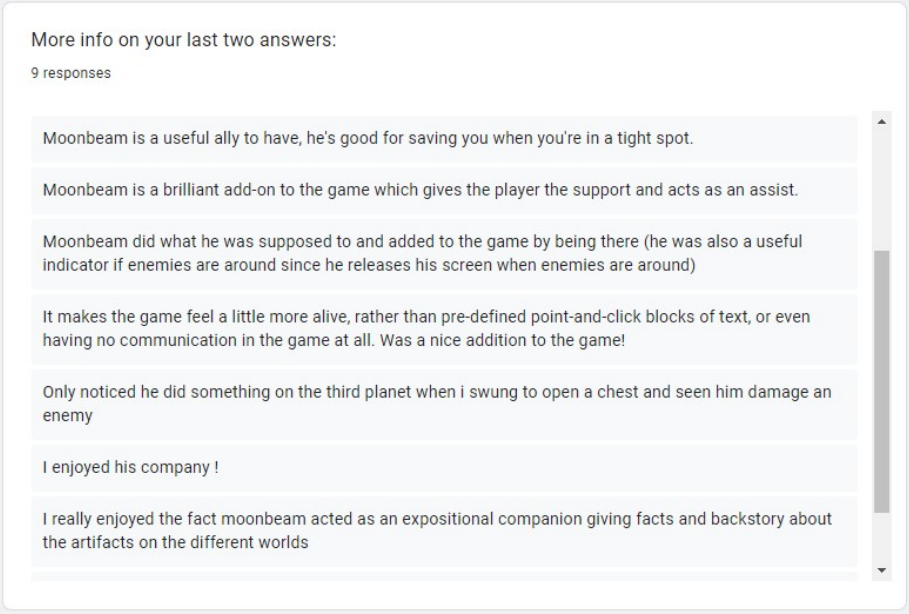


Figure 5.11: Final Test Question 4

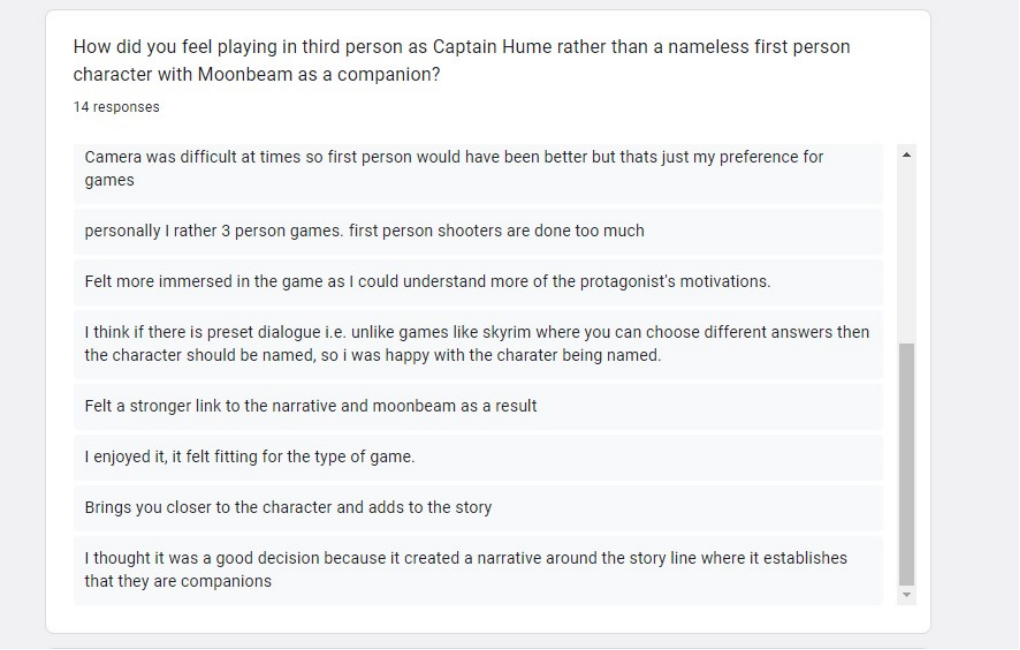


Figure 5.12: Final Test Question 5

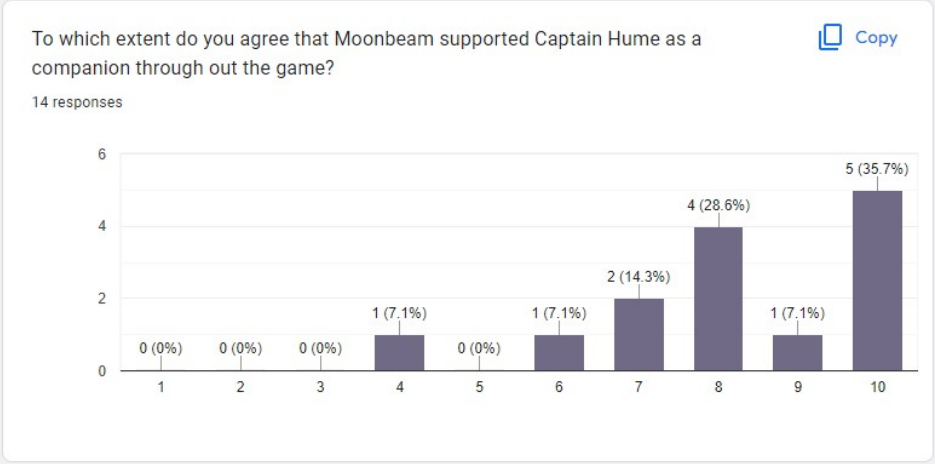


Figure 5.13: Final Test Question 6

Figure 5.14: Final Test Question 7

Figure 5.15: Final Test Question 8

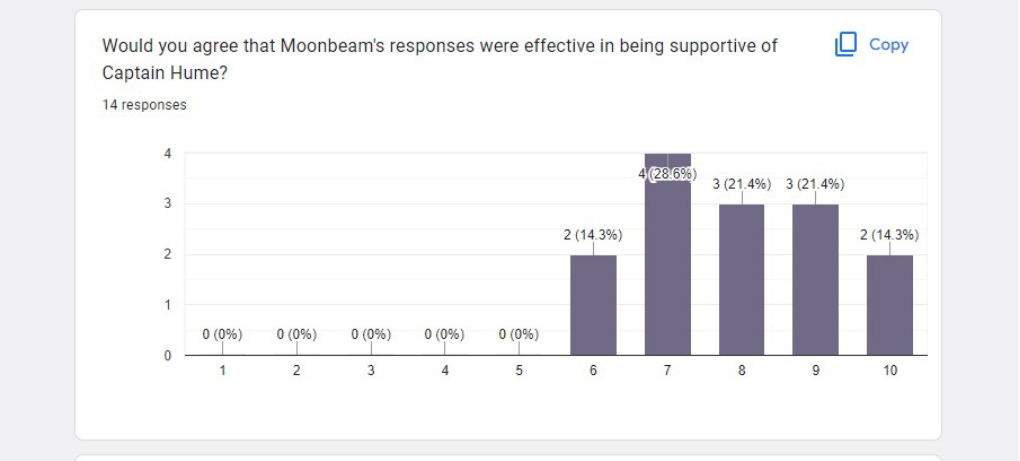


Figure 5.16: Final Test Question 9

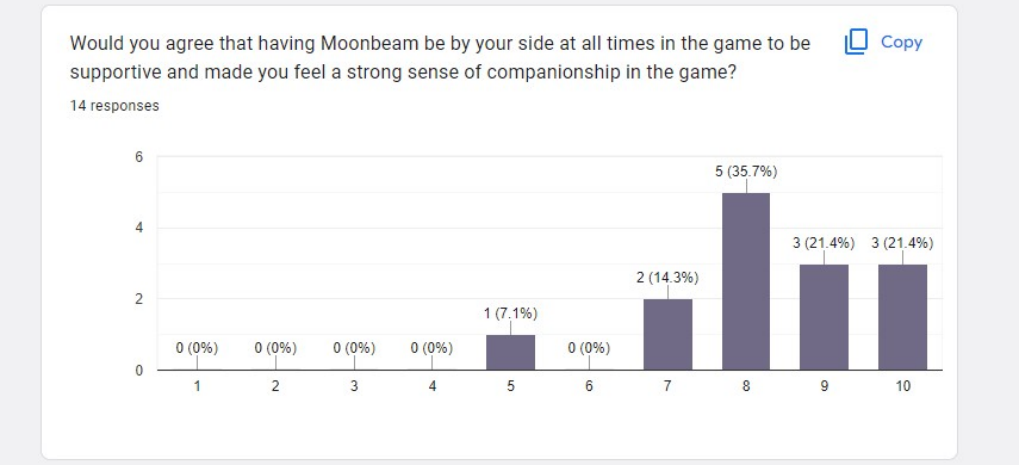


Figure 5.17: Final Test Question 10

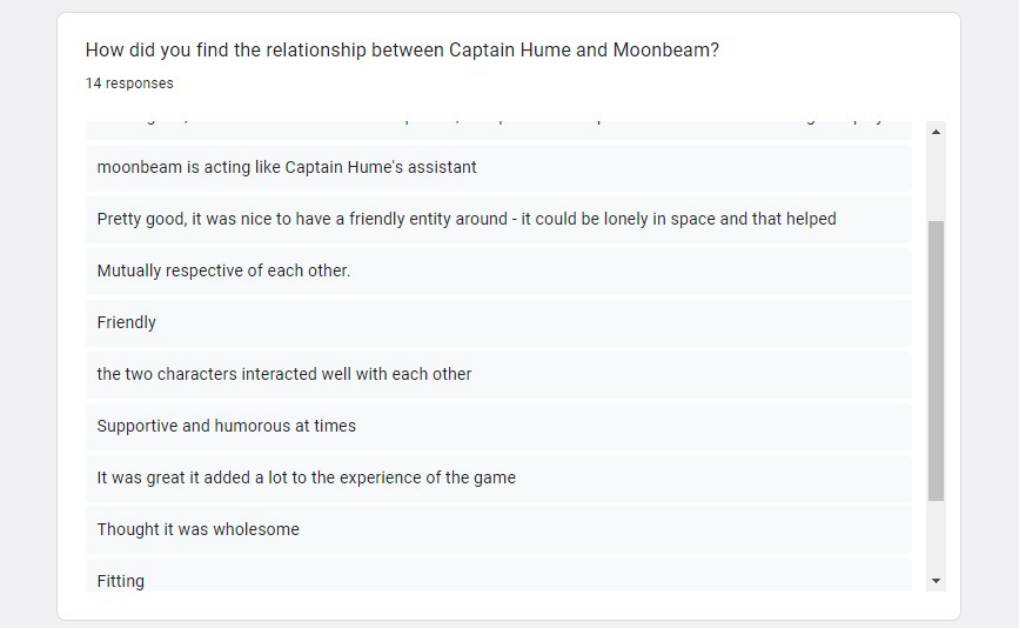


Figure 5.18: Final Test Question 11

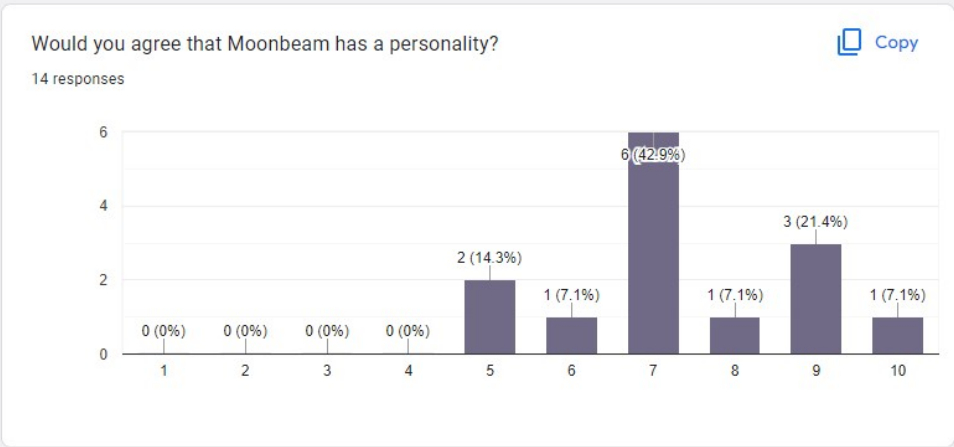


Figure 5.19: Final Test Question 12

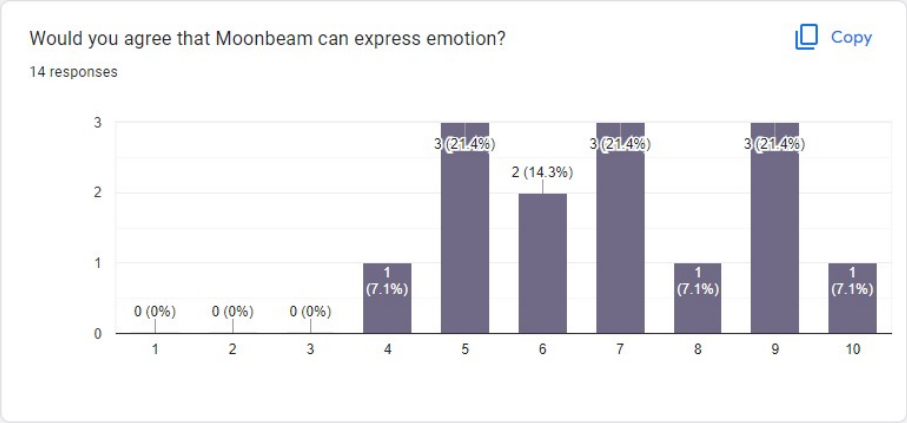


Figure 5.20: Final Test More info on Question 11 & 12

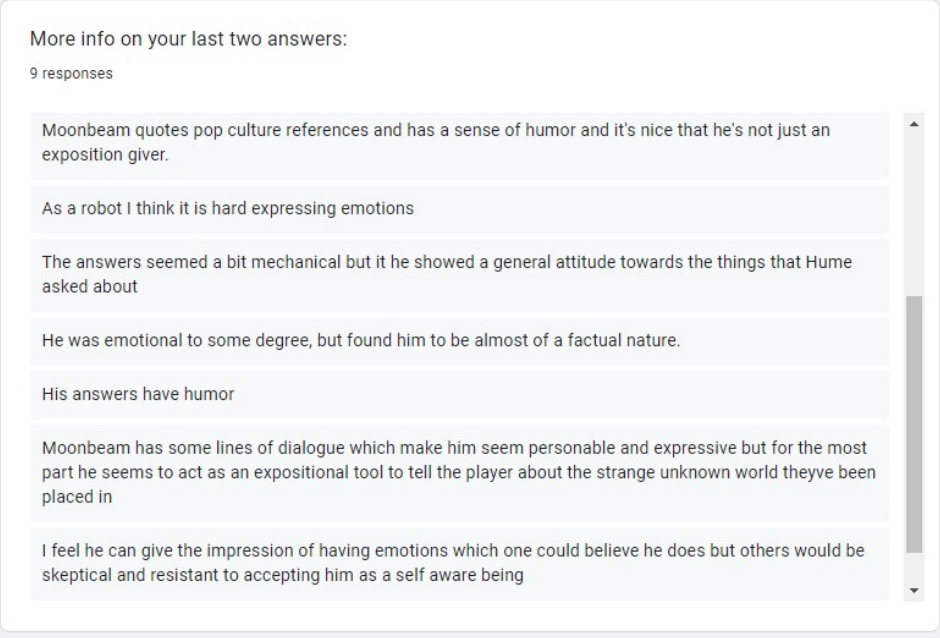
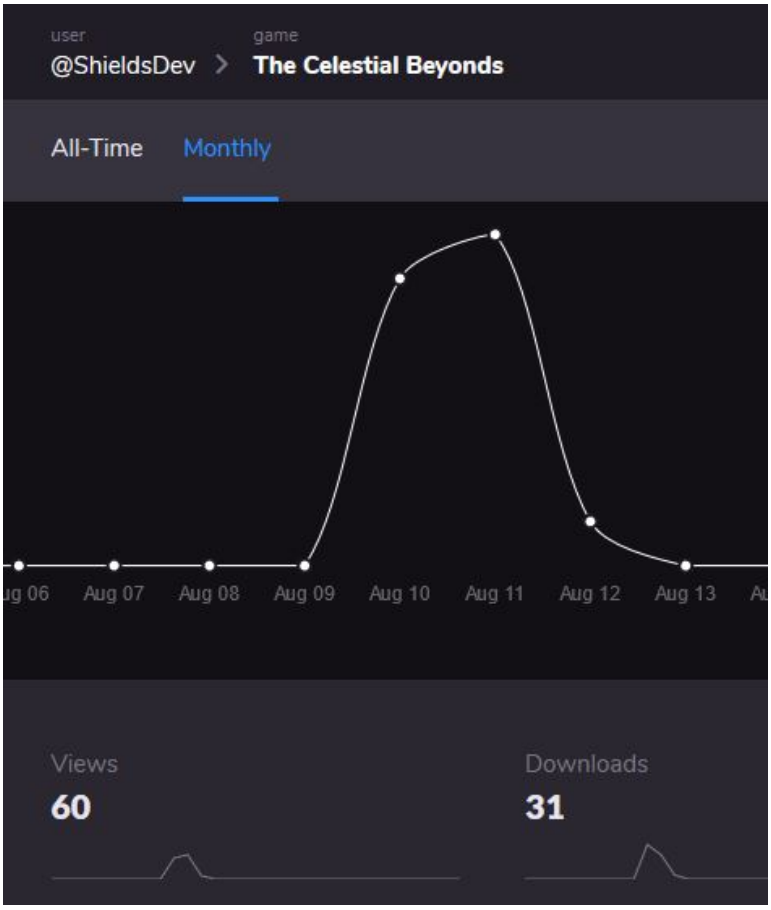


Figure 5.21: Final Test Question 13

It was quite pleasing to see these results from the Final Test as it shows that Moonbeam is indeed a supportive AI companion. The majority strongly agreed that Moonbeam is great addition, supportive, useful, effective, friendly, respective, humorous, has a personality, adds to the experience, can teach players about the game-lore, history, mythology and real-world artefacts and is the perfect companion for the game's story. On Game Jolt it shows the views and players of the game. 60 people have viewed it and 31 people have played it. These analytics (FB) show that in a matter of days the game is quite popular and hopefully they will continue to grow to gather more response to further improve the research and game. Looking back at the Test Demo, it can now be said that it as an essential process as it highly improved the research and implementation of the game.

What the Final Test's results displayed was that the research question was certainly worth studying, analysing and implementing into The Celestial Beyonds.

Figure 5.22: The Celestial Beyonds Analytics on Game Jolt



More info: Game Jolt counts Plays as Downloads in its analytics.

Chapter 6

Conclusion

6.1 Project Conclusion

The purpose of this project was to research the question *how Artificial Intelligence implementation can support companionship in Serious Games*. The next purpose was to implement this question and research into the game titled The Celestial Beyonds. The work involved included; building a ML Chat bot API using the technologies; Python with the libraries, Flask and ChatterBot along with an EC2 from AWS and a NGNIX setup to host the API, designing and developing a WebGL Sandbox style game that has Captain Hume and his loyal AI Companion, Moonbeam as the main stars who can interact with and support each other through General Conversation, Artifact findings and Combat, a narrative, lore and world set in space, has 3 Planets to explore, a Pollination and Combat System.

Overall, the final product and research study was a success and it achieved the objectives; the scope proved to be perfect size for a one-person project, resulting in a successful output from careful consideration, research, implementation and analysis through trial and error. The combination of Python with Flask, Chatterbot and AWS along with Unity worked out quite well. With them, The Celestial Beyonds turned out to be a great success as it has all its intended functionality. Through this report, it is evident that the game and research could be very successful in suitable for future development and further the research.

Reflecting on the first steps in the initial setup of the project (Test Demo) which included just the basic first level and the Moonbeam API and observing the differences made through targeting and achieving the intended

objectives of the project has thoroughly expanded my skills as a researcher and developer.

6.2 Research & Practical Objectives Achieved

- Research how AI implementation can support companionship in Serious Games ✓
  - AI Overview ✓
  - AI companionship ✓
  - Machine Learning ✓
  - Chat bots ✓
  - Path-finding ✓
  - AI companions in Games ✓
  - Serious Games ✓
- Develop an AI companion ✓
  - Python & Chatterbot - Machine Learning Chat bot. ✓
  - Python & Flask API for Chat bot. ✓
  - Training data on general conversation and the game's lore. ✓
  - Hosting of the API on AWS. ✓
  - Design 3D Model and Animations. ✓
  - Implementation into the game. ✓
  - Web Requests for the chat bot data from the API. ✓
  - Path-finding (to follow the player). ✓
  - General Dialogue System with the player. ✓
  - Game World Artifacts interaction and information System. ✓
  - Combat Support ✓
- Develop a WebGL Game ✓
  - Design 3D Models and animations. ✓
  - Player movement and Combat with enemies functionality. ✓

  - Pollination System with dead Plants in levels. ✓
  - In-game Trading Systems for Pollen and Ammo ✓
  - Level design of 3 'Sandbox' Planets + Boss Battle. ✓
  - Narrative about NASA's Voyager 1 and the Golden Record. ✓

- Surveys & Testing ✓
  - Moonbeam API and Level 1 Functionally for Test Demo. ✓
  - Google Form Survey for Test Demo. ✓
  - Send Test Demo to users and study their responses they gave in the form. ✓
  - Use the feedback responses from the form to improve the game.
  - Finish Game. ✓
  - Google Form Survey based on research and another for gameplay. ✓
  - Send Final Game to users and study their responses they gave in the form. ✓

Chapter 7

Appendix

Models and Assets

Rain Maker	https://tinyurl.com/2p8rep4r
Water Effect	https://tinyurl.com/4xu6puvv
Skeleton	https://tinyurl.com/5eushers
Flashlight PRO	https://tinyurl.com/52rz2wkz
Historic environment	https://tinyurl.com/mrxcbe44
48 Particle Effect Pack	https://tinyurl.com/nhd6ejnw
Desert Kits 64 Sample	https://tinyurl.com/yfe658sv
Vintage Radio	https://tinyurl.com/39k5eyxy
Volkswagen Kombi Micro Bus	https://tinyurl.com/b4v6wa8f
Avengers - Thor Hammer	https://tinyurl.com/4sssna8j
Zombie Head	https://tinyurl.com/3tav45jw
Demogorgon	https://tinyurl.com/2uf3uxw8
Templar Knight	https://tinyurl.com/awxfn9u2
Statue of Odin	https://tinyurl.com/bdz2nc9u
Basalt Walk	https://tinyurl.com/2p93vvf4
Engel	https://tinyurl.com/4hnhmmna
Apple Of Eden	https://tinyurl.com/35nbppua
Book Stand	https://tinyurl.com/yc38bffu

Stonehenge	https://tinyurl.com/2rxz3rxv
Stone celtic cross (1)	https://tinyurl.com/285a352z
Celtic Book	https://tinyurl.com/27tdc4jf
Viking Horn	https://tinyurl.com/3var392w
Old celtic stone	https://tinyurl.com/3fuenybr
Stone Celtic Structure	https://tinyurl.com/h6fwsjby
The Diver	https://tinyurl.com/44ywkdtx
Sand castle	https://tinyurl.com/yckfsc9k
Armor Helmet	https://tinyurl.com/23329a5j
DoodleBob	https://tinyurl.com/r5e27tw6
KRAKEN	https://tinyurl.com/ycksa26r
Ghost Ship	https://tinyurl.com/3442yhs7
Statue Of Liberty	https://tinyurl.com/yp537r8p
Dharma Initiative Lost	https://tinyurl.com/3j3uc5tz
Lost Lighthouse	https://tinyurl.com/5xkmwv2p

Radio From Metro 2033	https://tinyurl.com/mr3r6utv
Ammo Can	https://tinyurl.com/3bj8ephc
12G Ammo	https://tinyurl.com/2p8s9jek
Ammo Box	https://tinyurl.com/49u5r9r9
Wundderwaffle DG2	https://tinyurl.com/5x5nja2b
Hook Holder	https://tinyurl.com/2p8rfm42
Trapper Hat	https://tinyurl.com/mtxd4w4v
Ceramic Fragment	https://tinyurl.com/259fknx4
Xbox One Controller	https://tinyurl.com/44ds58n7
Dualshock 4	https://tinyurl.com/44dzdwt7
Mouse and Keyboard	https://tinyurl.com/ynxn8cmc

Google Survey Forms

Test Demo - Report Form

This form has been created by The Glasgow School of Art student John Shields for his Master's dissertation, under the supervision of Dr. Sandy Louchart. As part of John's dissertation, he has developed the game titled 'The Celestial Beyonds'. This form allows users to test and evaluate the game's current state. It has a list of questions about your experience with different aspects of the game. I am essentially trying to figure out what aspects of the game are effective and which ones need improvement. At the end of the form, there is a open box for the freedom to write comments, any suggestions or ask any questions you may have. With your consent, these answers will be used to improve the game. Thank you for supporting this as it will certainly help the game's development.

Play the Game: The Celestial Beyonds - Test Demo (Itch.io)

For more information, you can contact John @: shields.game.dev@gmail.com

*Required

1. Do you understand and consent to the use of the data you provide here for the above-mentioned purpose? (Your data will be kept anonymous) *

Mark only one oval.

- ◯ Yes
- ◯ No

2. Q1. How was your overall experience of the game? *

Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Poor	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Excellent

3. More info on your answer for Q1:

4. Q2. How did the AI Companion Bot (Moonbeam) add to the experience of the game? *

Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Not at all	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Very much so

5. More info on your answer for Q2:

6. Q3. Did Moonbeam come off as _____? *

Tick all that apply.

- [ ] Friendly
- [ ] Supportive
- [ ] Intelligent
- [ ] Helpful
- [ ] Useful
- [ ] Well mannered
- [ ] Humble
- [ ] He could do more to support Player
- [ ] Other: _____

7. More info on your answer for Q3:

8. Q4. How would you rate the Dialogue System with Moonbeam? *

Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Poor	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Excellent

9. More info on your answer for Q4:

10. Q5. How did you find Moonbeam's Responses? *

Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Poor	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Excellent

11. More info on your answer for Q5:

12. Q6. How was the Level Design of Planet TRAPPIST-1? *

Mark only one oval.

	1	2	3	4	5	
Poor	◯	◯	◯	◯	◯	Excellent

13. More info on your answer for Q6:

14. Q7. How did you find the Combat with the Enemies? *

Mark only one oval.

	1	2	3	4	5	
Easy	◯	◯	◯	◯	◯	Hard

15. Who were the hardest Enemies to Terminate? *

Mark only one oval.

- ◯ Frogs
- ◯ Spiders
- ◯ Venus Traps
- ◯ Mantises
- ◯ Robot Bears

16. More info on your answer for Q7:

17. Q8. How did you find the Pollination System with the Plants? *

Mark only one oval.

	1	2	3	4	5	
Poor	◯	◯	◯	◯	◯	Excellent

18. More info on your answer for Q8:

19. Q9: How did you find the Pollen and Peridot Trading System with the Argyle at The Hive Shack? *

Mark only one oval.

	1	2	3	4	5	
Poor	◯	◯	◯	◯	◯	Excellent

20. More info on your answer for Q9:

21. Q10. How did you find the Collectable Systems with the Peridots and Artifacts? *

Mark only one oval.

	1	2	3	4	5	
Poor						Excellent

22. More info on your answer for Q10:

23. Q11. How did you find the Health System with the collectable Honey Jars to regain health? *

Mark only one oval.

	1	2	3	4	5	
Poor						Excellent

24. More info on your answer for Q11:

25. Q12. How did you find the Controls? *

Mark only one oval.

	1	2	3	4	5	
Poor	◯	◯	◯	◯	◯	Excellent

26. More info on your answer for Q12:

27. Q13. How did you find the User Interface (Menus/HUD)? *

Mark only one oval.

	1	2	3	4	5	
Poor	◯	◯	◯	◯	◯	Excellent

28. More info on your answer for Q13:

29. Q14: Did you find any Bugs (Major or Minor)?

Tick all that apply.

- [ ] Major
- [ ] Minor
- [ ] None

30. More info on your answer for Q14:

31. If you have any comments, suggestions or questions please write them here.

The Celestial Beyonds - Research Implementation Evaluation

This form has been created by The Glasgow School of Art student John Shields for his Masters dissertation, under the supervision of Dr. Sandy Louchart. As part of John's dissertation, he has developed the game titled 'The Celestial Beyonds'. This form allows users to test and evaluate the game in terms of the research area. It has a list of questions about your experience with different aspects of the game. I am essentially trying to figure out what aspects of the game are effective and which ones could be improvement. At the end of the form, there is a open box for the freedom to write comments, any suggestions or ask any questions you may have. With your consent, these answers will be used to improve the game. Thank you for supporting this as it will certainly help the game's development.

Play the Game: The Celestial Beyonds on Game Jolt

For more information, you can contact John @: shields.game.dev@gmail.co

*Required

1. Do you understand and consent to the use of the data you provide here for the above-mentioned purpose? (Your data will be kept anonymous) *

Mark only one oval.

- ◯ Yes
- ◯ No

2. Have you studied topics related to Artificial Intelligence? *

Mark only one oval.

- ◯ At University
- ◯ Online courses
- ◯ I have studied Artificial Intelligence by myself
- ◯ I have not
- ◯ Other: _____

3. Would you agree that Artificially Intelligent companions can be supportive? *

Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Strongly Disagree	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Strongly Agree

4. How would you rate your overall experience with Moonbeam as a companion in the game? *

Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Poor	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Excellent

5. To which extent do you agree that the character Moonbeam adds to the experience of the game? *

Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Strongly Disagree	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Strongly Agree

6. More info on your last two answers:

7. How did you feel playing in third person as Captain Hume rather than a nameless first person character with Moonbeam as a companion? *

8. To which extent do you agree that Moonbeam supported Captain Hume as a companion through out the game? *

Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Strongly Disagree	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Strongly Agree

9. Did Moonbeam come off as _____? *

Tick all that apply.

- [ ] Supportive
- [ ] Friendly
- [ ] Intelligent
- [ ] Helpful
- [ ] Useful
- [ ] Well mannered
- [ ] Humble
- [ ] Empathetic
- [ ] He could do more to support Player
- [ ] Other: _____

10. More info on your answer:

_____

_____

_____

_____

_____

11. Would you agree that being able to talk with Moonbeam makes you feel a strong sense of companionship in the game? *

Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Strongly Disagree	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Strongly Agree

12. Would you agree that Moonbeam's responses were effective in being supportive of Captain Hume? *

Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Strongly Disagree											Strongly Agree

13. Would you agree that having Moonbeam be by your side at all times in the game to be supportive and made you feel a strong sense of companionship in the game? *

Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Strongly Disagree											Strongly Agree

14. How did you find the relationship between Captain Hume and Moonbeam? *

15. Would you agree that Moonbeam has a personality? *

Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Strongly Disagree											Strongly Agree

16. Would you agree that Moonbeam can express emotion? *

Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Strongly Disagree											Strongly Agree

17. More info on your last two answers:

18. To which extent do you agree that Moonbeam helped you as a player learn about the game world and it's inspired history? *

Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Strongly Disagree	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Strongly Agree

19. More info on your answer:

20. If you have any comments, suggestions or questions please write them here.

References

Bouquet, E., Makela, V. and Schmidt, A., 2021. Exploring the Design of Companions in Video Games. In Academic Mindtrek 2021 (pp. 145-153).

Skinner, G. and Walmsley, T., 2019, February. Artificial intelligence and deep learning in video games a brief review. In 2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS) (pp. 404-408). IEEE.

Turing, A.M., 2009. Computing machinery and intelligence. In Parsing the turing test (pp. 23-65). Springer, Dordrecht.

Risi, S. and Preuss, M., 2020. From chess and atari to starcraft and beyond: How game ai is driving the world of ai. KI-Künstliche Intelligenz, 34(1), pp.7-17.

Merrill Jr, K., Kim, J. and Collins, C., 2022. AI companions for lonely individuals and the role of social presence. Communication Research Reports, 39 (2), pp.93-103.

Ongsulee, P., 2017, November. Artificial intelligence, machine learning and deep learning. In 2017 15th international conference on ICT and knowledge engineering (ICT & KE) (pp. 1-6). IEEE.

Rahman, A.M., Al Mamun, A. and Islam, A., 2017, December. Programming challenges of chatbot: Current and future prospective. In 2017 IEEE

Region 10 Humanitarian Technology Conference (R10-HTC) (pp. 75-78). IEEE.

Lokman, A.S. and Ameedeen, M.A., 2018, November. Modern chatbot systems: A technical review. In Proceedings of the future technologies conference (pp. 1012-1023). Springer, Cham.

Skjuve, M., Følstad, A., Fostervold, K.I. and Brandtzaeg, P.B., 2021. My chatbot companion-a study of human-chatbot relationships. International Journal of Human-Computer Studies, 149, p.102601.

A*-based pathfinding in modern computer games. International Journal of Computer Science and Network Security, 11(1), pp.125-130

Paiva, A., 2022. Social Robotics. [online] Ana Paiva. Available at: https://ana-paiva.com/home/about/projects/ [Accessed 16 June 2022]. Hepp, A., 2020. Artificial companions, social bots and work bots: communicative robots as research objects of media and communication studies. Media, Culture Society, 42(7-8), pp.1410-1426.

Dario, P., Verschure, P.F., Prescott, T., Cheng, G., Sandini, G., Cingolani, R., Dillmann, R., Floreano, D., Leroux, C., MacNeil, S. and Roelfsema, P., 2011. Robot companions for citizens. Procedia Computer Science, 7, pp.47-51.

Bekey, G.A., Ambrose, R., Kumar, V., Sanderson, A.C., Wilcox, B., Zheng, Y.F., Yuh, J.K. and Lavery, D., 2008. Robotics: state of the art and future challenges.

Siciliano, B., Khatib, O. and Kröger, T. eds., 2008. Springer handbook of robotics (Vol. 200). Berlin: springer.

Biundo, S., Höller, D., Schattenberg, B. and Bercher, P., 2016. Companion-

technology: An overview. KI-Künstliche Intelligenz, 30(1), pp.11-20.

Emmerich, K., Ring, P. and Masuch, M., 2018, October. I'm glad you are on my side: How to design compelling game companions. In Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play (pp. 141-152).

Pinchbeck, D., 2009. An Analysis of Persistent Non-Player Characters in the First-Person Gaming genre 1998-2007: a case for the fusion of mechanics and diegetics. Eludamos: Journal for Computer Game Culture, 3(2), pp.261-279.

Lankoski, P. and Bjork, S., 2007, September. Gameplay Design Patterns for Believable Non-Player Characters. In DiGRA Conference (pp. 416-423).

Dignum, F., Westra, J., van Doesburg, W.A. and Harbers, M., 2009. Games and agents: Designing intelligent gameplay. International Journal of Computer Games Technology, 2009.

Aarseth, E., 2012, May. A narrative theory of games. In Proceedings of the international conference on the foundations of digital Games (pp. 129-133)

Poos, J.M., van den Bosch, K. and Janssen, C.P., 2017. Battling bias: Effects of training and training context. Computers & Edu- cation, 111, pp.101-113.

Wilkinson P. (2016) A Brief History of Serious Games. In: Dörner R., Gobel S., Kickmeier-Rust M., Masuch M., Zweig K. (eds) Entertainment Computing and Serious Games. Lecture Notes in Computer Science, vol 9970. Springer, Cham.

Caserman P., Cornel M., Dieter M., G obel S. (2018) A Concept of a Training Environment for Police Using VR Game Technology. In: Gobel S. et

al. (eds) Serious Games. JCSG 2018. Lecture Notes in Computer Science, vol 11243. Springer, Cham.

Plass, J.L., Homer, B.D. and Kinzer, C.K., 2015. Foundations of game-based learning. Educational Psychologist, 50(4), pp.258-283.

Kirriemuir, J. and McFarlane, A., 2004. Literature review in games and learning.

Sony Computer Entertainment Naughty Dog. 2014. The Last of Us. Game [PlayStation 3]. https://store.playstation.com/en-gb/product/EP9000-CUSA00557_00-THELASTOFUS00000

Bethesda Softworks. 2008. Fallout 3. Game [Xbox 360, PlayStation 3, Windows]. https://fallout.bethesda.net/en/games/fallout-3

Sony Computer Entertainment Naughty Dog. 2001. Jak and Daxter. Game [PlayStation 2, PlayStation 4]. https://store.playstation.com/en-gb/product/EP9000-CUSA07934_00-JAKDAXTER0BUNDLE

Sony Computer Entertainment Insomniac Games. 2021. Ratchet & Clank: Rift Apart [Playstation 5]. https://store.playstation.com/en-gb/product/EP9000-PPSA01474_00-RATCHETCLANKRIFT

Replika App (2022) https://replika.com/

Call of Duty: Infinite Warfare (2022) https://www.callofduty.com/uk/en/infinitewarfare

Redhat - What is a virtual machine (VM)? (2022) https://www.redhat.com/en/topics/virtualization/what-is-a-virtual-machine

NGINX (2022) https://www.nginx.com/

Atlassian - Agile Project Management (2022) https://www.atlassian.com/agile/project-management

ChatterBot (2022) https://chatterbot.readthedocs.io/en/stable/