Repercussions – Story Based Game

Final Report for CS39440

Author: Jamie Hall (jah79@aber.ac.uk)

Supervisor: Dr./Prof. Bernie Tiddeman (bpt@aber.ac.uk)

3rd May 2018

Version: 1.0 (Release)

This report is submitted as partial fulfilment of a BSc degree in Computer Graphics, Vision and Games (G450)

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

Wales, UK

Declaration of Originality

I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Quality and Records Office (AQRO) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work I understand and agree to abide by the University's regulations governing these issues.

Name: Jamie Hall
Date: 28/04/2018

Consent to Share This Work

By including my name below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name: Jamie Hall
Date: 28/04/2018

Acknowledgements

I would like to thank my project supervisor, Bernie Tiddeman for their invaluable help, support and guidance through this project.

I would also like to thank my family, most notably my father Kevin, my mother Stella and my Grandparents, Raymond, Christina, Raymond, Anette.

Abstract

Gaming is a growing industry with various genres. It's become a great way to show/express a story. There are different ways of doing this such as interactive cutscenes or a role-playing game etc. This projects aim is to produce a game made with the story of the game as the centre of it. However, gameplay elements are equally important as this contains the fun of the game.

I wanted to learn to use a game engine to produce a game as it is one of the reasons I chose to join this degree course. I also plan on recording animation through motion capture and how to use these in game as I believe this is something that will not only improve the game but will be useful to know if I continue down this career path.

Table of Contents

1 Background

1.1 Introduction

The gaming industry is constantly growing and has become one of the major forms of entertainment. In a society evolving more and more around technology, gaming has become popular in modern day society. For many it can provide an escape from reality.

There are various genres under gaming, some are strong on story and some may be more focused on fun gameplay elements. More recently, games like 'Life is Strange', 'A Way Out' and the Telltale franchise have had a focus on making an enjoyable interactive story-telling experience. These games were a big motivation in the decision of the aim for this project. Some games are used to convey important, powerful or emotional messages. With technology ever-evolving, developers are finding new and innovative ways to develop more immersive games.

The Kinect is a motion capture device, originally used as an Xbox accessory but with an adapter can be used as a motion capture device when connected to a PC. Motion capture can make it easier to record more accurate human-like animations to make movements and scenes more immersive. Kinect is a cheap method to record animation.

There are various game engines available to download for free. Some require high experience but some have been made to accommodate for beginners or developers who prefer simplicity over advanced technology.

1.2 Aim

The aim of this project is to produce a story based game that implements a dialogue system and player choice to further progress or change the story, allowing for more replay ability and variation. The game should also include the use of animation and motion captured animation to add a more immersive and professional look. Gameplay elements to be added, allowing player to walk or explore an area to ensure the game is more than interactive movie.

1.3 Objectives

- Convey a story
- Implement a dialogue system
- Use of motion capture animations
- Gameplay elements (including walking and interacting with other objects)
- Player choice implemented with noticeable differences

1.4 Deliverables

Project deliverables include:

- Final Technical – Source code of program. Playable game. Assets used.
- Final Report – Project report which includes the discussion of the design, methodology, testing and evaluation based on the project.

1.5 Methodology

A work plan was implemented at the beginning of the project to set goals and deadlines for separate tasks. Breaking the project into smaller goals made it easier to keep track on the progress of the project. After realising the project was behind schedule half way to the deadline, a basic Scrum framework was implemented in second half of the work period. These sprints ensured more work was completed and at the end of each sprint, the progress made from that sprint would be evaluated and this could change the work needed to be done during the next sprint. The game was

made to be easily playable by anyone, not needing experience with games. With every important addition to the game, some testing would be executed to ensure no problems would be encountered during a playthrough of the game.

1.5.1 Version Control & Back-up

The project folder was frequently backed-up to a USB, with each new back-up of the project being a separate folder so that if any problems would be encountered with one version of the project, a previous version of the project would be available to continue work on.

1.6 Degree Relevance

This project has used many of the skills and techniques learnt form the Computer Graphics, Vision and Games course. The CS32420 (Computer Graphics and Games) and CS24320 (Applied Graphics) modules both provided the most useful and necessary skills for this project. This included the use of a games engine, coding objects and creating animations. CS34110 (Computer Vision) helped in the understanding of how the Kinect recorded the motion capture and what some of the settings for the motion capture mean. CS26520 (Program Design, Data Structures and Algorithms) assisted in the understanding of data structures and helped in deciding which data structures to use. CS22120 (Software Engineering) taught the use of version control and the importance of backing up a project.

2 Research

This section will contain a brief description of the problem for the project, the research carried out in preparation for the project, and a look at similar products that influenced the problem to this project. It is important to first know exactly what the problem is for the project before researching into possible solutions.

2.1 Problem Overview

Stories in some games are as important if not more important than in films, capturing the player and immersing them in a world they want to be in. However, gameplay elements need to be balanced out to ensure there is a factor of player involvement and interaction otherwise it would just be a film or a show.

Replay-ability is also important in the modern gaming industry with developers wanting as many players to buy and play their game. Developers need to add something to their game that will keep attracting players back to play their game. In some major story-heavy games, this is done by adding choices which effect the game to allow for variation in dialogue, gameplay and story. This means every time the player chooses something different they will have a different experience, to some extent.

Another important element is the quality of game mechanics and animations. If a game has bad animations or broken gameplay, the players experience may be disrupted. Most games these days use motion capture to implement life-like/human-like animations.

The main solution for this project requires a good balance between gameplay and story-telling, an aspect of replay-ability and animation at some level of good quality.

2.2 Similar Products

It was important to look at some famous story-telling games to get an idea of what gameplay mechanics, dialogue systems, quality of animation, and any other factors that made these games well received.

A Way Out: This game manages to find a great balance between player interaction, gameplay and story. Various gameplay elements with some scenes involving driving a car, escaping a cell in prison, player using a gun, running and hiding. The dialogue system is automatically activated during cutscenes or activated by walking up to an NPC and pressing a button. The dialogue text appears as white text with no background. The replay-ability of the game comes from the fact it can only be played in co-op, the 2 players can choose which character they wish to be and each character will have a different experience. Choices must also be made by both players throughout the games that shape the story.

Telltale Games: Telltale games were some of the main influences in the decision to do this project. Telltale have made various games from various franchises including Batman, Guardians of the Galaxy and The Wolf Among Us. The animation in Telltale games aren't the best and haven't improved much over the few years they've been releasing games but the stories in the games are written so well that the animation doesn't take away from the experience too much. The dialogue system is well built, in most cases the dialogue choices come with a timer so the player must choose an option quick or their character will just remain silent. The choices mean that the plot of the stories in a Telltale game can lead down different paths and experiences.

Life is Strange: Although I haven't played this game, I have researched into the game a bit as it is well received by many. The format is very similar to that of a Telltale game. The games story is

apparently more realistic than a Telltale game with more realistic and relatable characters. The fact that players could relate to the characters meant many players enjoyed it.

Mass Effect: The original Mass Effect Trilogy is different to the games listed above as it is more of a 3rd person action/RPG. The gameplay contains some walking and exploration, more gunplay and a lot of dialogue. The story in the first Mass Effect game engulfed players enough to return for the next instalment, the second game done the same. The characters in this game are very well written and add to this. Choices made in the game can upset your relationships with characters and there are a number of other factors that can influence the story of the games. In Mass Effect 3 some missions were timed so that if you didn't complete a side mission before a major plot event, the mission would become unavailable but the story can form around it (e.g. If the side mission is to save students from the academy and the player doesn't go in time, the academy will be destroyed). Choices made in the first game can have effect in the second game, and choices from the first two games can influence the third game. This is done by importing saves and is one of the reasons players kept playing, to continue their story. Animations are average but facial animations are below average. There was an uproar about this with the newest Mass Effect instalment.

From these games, it is apparent that what makes a good story based game is a well implemented dialogue system, a good story, some fun and interactive gameplay elements. Animation is an added bonus but if the rest is executed well then this is not much of an issue although does affect the level of immersion.

2.3 Research of Game Development

As this project involves making a game, I looked into some guides and articles to get me started and point me in the right direction. The article from Kotaku states that a beginner should "start small and be ready to fail" and follow a mindset of trial and error. The article also states that as a beginner it is best to aim for "a smaller version of your idea with few working mechanics, a prototype that lets you see how your game plays, allows you to improve the design, and forms the foundation of the game". As I have found out from the degree scheme, with each new computing challenge comes a slight learning curve and this is possibly the best way to improve yourself and condition yourself to this type of challenge [1].

The Unity 3D website provides a short guide for game development, this was looked at in preparation for the project [2]. Some advice given in the video is to keep to your goals and warns that once the game development starts it is easy to keep wanting to add more features but this can lead to overloading the workload. It also states that it is best to start small and keep working on small project before trying to make anything big. With this project there is not enough time to go working on smaller projects before working on the main game. However, this could be implemented differently with this project by starting on small parts of the project that may feel familiar based on what has been learnt throughout the university course.

2.4 Research of Kinect & Motion Capture

Upon researching motion capture for this project, it quickly became clear that the best method for the lowest price was to use an Xbox Kinect with an adapter. There are various programs that allow the user to activate their Kinect, record an animation and save it. However, in the Unity game engine asset store there is a fairly cheap Kinect motion capture software names 'Cinema Mo-Cap' that allows the user to record the animation, see what the animation would look like in game, save the animation file and import straight to the game. This was going to be the most likely method to record motion capture for the game as I already had a Kinect 2.0 device.

2.5 Research of Game Engines

As this was the first time I would be making a game using a game engine, it was important to research into the most common game engines for beginners to see which would be best suited for this project and for a beginner.

Unity: Appears to be better for beginners, easier to learn how to use the software in comparison to other similar game engines. The inclusion of the asset store is especially useful for this project as it includes the Cinema Mo-Cap software mentioned in 2.5. The less assets used however the better. While researching which game engine to use, I installed both Unreal engine and Unity 3d to try them out. I installed a few free assets to practice using unity but realised that assets can install and add extra material to the game files that may not actually be used. For this project, the only asset that will be used is the Cinema Mo-Cap asset. Unity is free for personal use and students.

Unreal: The Unreal engine seems slightly more advanced than Unity especially in terms of graphics but it is still good for beginners. This engine may take longer to learn how to use it and will likely have a steeper learning curve. Although the Unreal engine has an asset store, I could not find an affordable Kinect motion capture asset. Due to project time, it is best to stick with unity. This would be the more preferable game engine and when creating any games in the future where time is not as pressing, will likely be the game engine I will use.

2.6 Relevant Technology

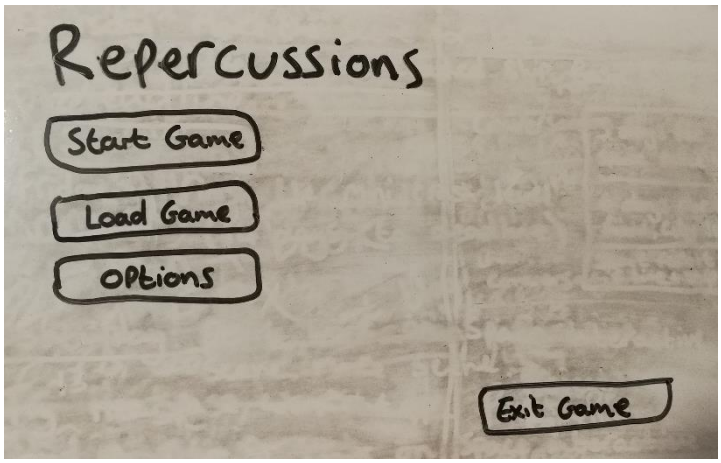
The development of this game will all be executed on a Windows laptop device. After the results of the research in the above sections, the Unity game engine will be the one used for this project. The asset stores' Cinema Mo-Cap asset will be utilised to record animations using an Xbox Kinect 2.0. A PC adapter will have to be used to allow the Kinect to work with the laptop. The Windows Kinect 2.0 Software Development Kit must be installed before use or the Kinect will not work correctly. Research was also carried out into character model creation. For the beginning of the development process, a placeholder model will be used for characters, likely a free model found online. When the character models are to be added to the game, a good character creation software to use is Adobe Fuse CC which allows for the user to choose head, arms, legs, torso and add clothing of choice. The Character model can then be exported to a website called Mixamo where a skeleton rig is automatically added to the model. Both Adobe Fuse CC and Mixamo are free to use. Visual Studio 2010 will be used as the IDE to write the code for the game.

3 Design

This section will look at the design process and discuss any decisions made in the designing of the game. This includes the decided programming language, game engine, IDE, data structures and the graphical design of the game.

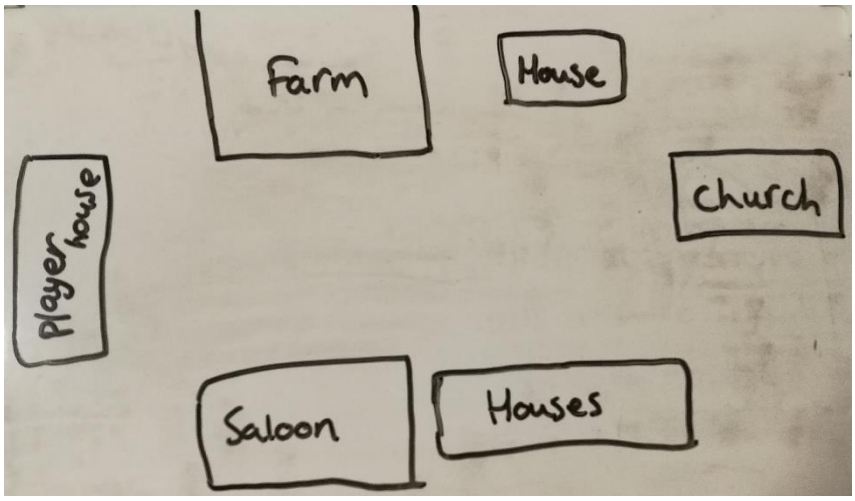
3.1 Architecture of Game

Main Menu – Basic menu. Image as background, slight visual effect to act as sand blowing across the screen. Buttons to start a new game and exit a game. If there is enough time towards the end of the project, implement a load game system.



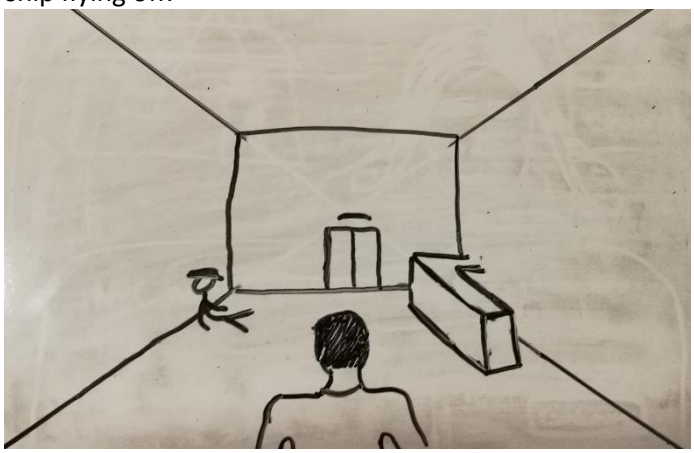
First Cutscene – Basic scene set in space in which various animations play to display a cutscene where a large delivery ship is attacked by three smaller fighter type ships. Script to switch to next scene either when animation ends or a timer to change scenes after a set time. No interaction.

Town Level – One of the bigger levels of the game. Scene is a western style town on a desert planet. Player can walk around and interact with the townsfolk, some may talk irrelevant dialogue but some may give some hints into the backstory or the player can simply walk straight to the objective. When a player gets so close to an NPC, a message should pop up to notify the player to press a specific button to interact with the NPC. Upon button press, the dialogue is displayed through text on a panel. After the player reaches the objective, a cutscene will play where the ship from the previous cutscene crashes into the land. Afterwards the player will have to make a decision on which part of the ship to check. This will be done by buttons displayed on a panel which when pressed will take the user to the corresponding scene (Debris Level or Crashed Ship Level). The camera for this level will follow the player as a third person point of view and rotate with the player model object.



Debris Level – If the Player chooses to check the debris first, they will be brought to this level. This is more of an extra level where the player can explore the small scene where they can look at various items. Some of the items can show what sort of items exist in that universe. When the player finds a 'ship jump-starter', they will be taken to the Crashed Ship Level. The jump-starter object found here will be used in a later level.

Crashed Ship Level – If the player chooses to check the crashed ship at the end of the Town Level, they will be brought straight to this level. Otherwise, after the debris level they will be brought here. This scene should be set in a large futuristic sci-fi looking corridor/hallway. Fallen debris and injured guards left around the scene to add to the atmosphere. Flickering lights along the hallway. The player will be able to walk down the hallway interacting with whatever objects they want to interact with. The camera will only move up the hallway, unlike the first level where the camera rotates and follows the player object. Player can pick up a laser pistol in this scene which if they do, they will the players character will use it in a later cutscene. Player can interact with each guard. The door at the end of the corridor will be closed and will not open until the player talks to the guard closest to the door. The guard closest to the door will explain to the player what happened to make the ship crash, developing the story and gives the player an important item. After speaking to the guard, the player will be able to walk through the door and climb onto a spaceship. This ends the level and displays a small cutscene of the ship flying off.



Inside Ship Level – This level takes place inside the ship the player flies off with in the Crashed Ship Level. Dialogue progresses the story here but gets interrupted by another ship. The ship introduces the enemies to the player, they board the players ship and the boss asks for the item given to the player in the previous level. The confrontation with the boss is a series of recorded animation, camera changes and dialogue. Maybe add some flashing warning lights and dim other lights. After the enemies leave, one of two things will happen. As the ship is left disabled, the player will either use the jump-starter found from the debris level, or set off a distress beacon to notify nearby police for help.

Police Level – This level would be similar to the town level, where the player has an area they can walk around and explore. The player will be able to interact with NPCs at the space police station. After interacting with three objects/NPCs, the camera will switch to another camera and show an agent calling the player over and asking for help. Through dialogue, the agent says they've tracked down the enemies based off the data on their ship and asks for the players assistance in dealing with them.

Final Level – This final level is an interactive cutscene. The Player finds the enemies on a futuristic planet making some super weapon. The player and the agent must fight to stop them. Using motion capture to record the animations, a fight sequence can be made where the player will have to press

the buttons that show on screen to progress the scene. Animation plays -> animation pauses -> player is notified to press button -> animation continues, this repeats until end of scene. Game ends with short scene of the characters overlooking the futuristic planet they've just saved.

3.2 Development

3.2.1 Game Engine

The game engine that will be used in the creation of this game will be Unity 3D. After the research made in section 2, it seemed as Unity would be the best option as this is my first time using a game engine and the Unity asset store contains the Cinema Mo-Cap software which will be very useful in the making of the animation for this project.

3.2.2 Development Environment & Programming Language

Unity scripts can be written in either C# or Java. C# will be used for this project as it is the default option for Unity and tutorials/practices done during the research process were done in C#. Visual Studio 2010 will be used as the IDE as I have some experience with Visual Studio. If any problems are encountered along the way Unity comes with a build in IDE which can be used instead.

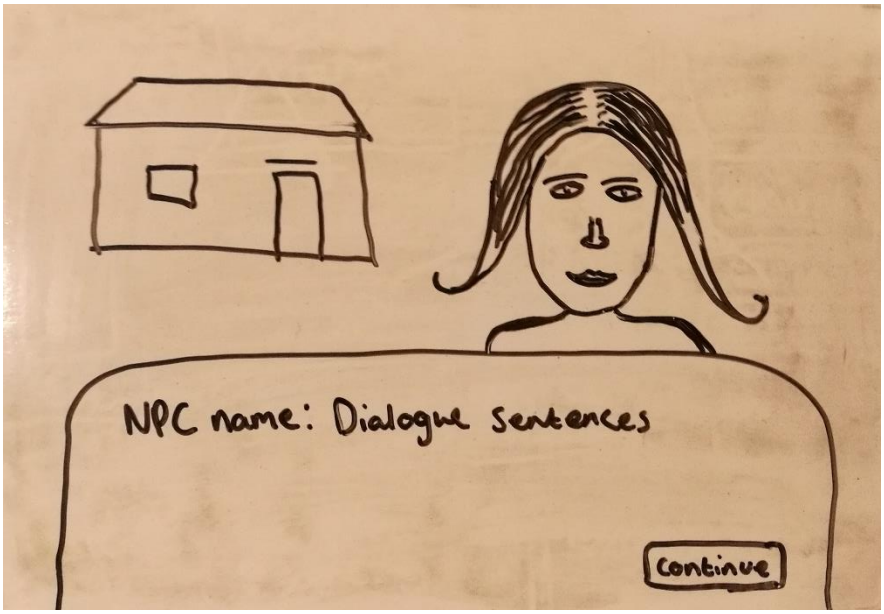
3.3 Game Design

3.3.1 Menu

The menu will have a western/desert image as the background as it will give the look of the planet the first level is set on. There will be a button for the user to start a new game and a button to exit the game. If there is enough time, an options button will be added to allow the user to change some basic settings. The title of the game will be at the top of the screen as large test.

3.3.2 Dialogue

An array of strings will be used to hold each conversation. The strings will be queued and then dequeued to display each sentence in the order it should be. The dialogue text will be displayed as text on top of a panel. The panel behind the text will make it easier to read the sentences. A continue button will be placed at the bottom right of the panel which when clicked will display the next sentence. There will also have to be a function in the script that when the queue is empty it will remove the dialogue panel. When dialogue is started with a character, the main camera should switch to another, the other camera will be in a better view to see both the character the player is talking to and the model of the player.



3.3.3 Scenes & Models

Some free online models will be downloaded to save time as making all models myself would take up time better spent working on other areas of the project. A placeholder character model will be used for al characters to begin with just until the game itself is working to some extent.

Cutscene: Space and stars sky-box will be used to set the scene in space. Two spheres will be used, one as a planet and the other as an asteroid. Two different ship models will be used here, one to be used as the delivery ship and the other to act as the model for the three fighter ships. If no model can be found online for free, a basic model will be made instead.

Town Level: Some building models will be downloaded and placed to form the town. A simple Plane will be used as the ground and a sand-like material will be made to go with it. The characters will later be created with Adobe Fuse CC. Character models will have collider components added to allow for dialogue to be triggered.

Debris Level: Plane used as sandy ground again. Basic shapes to be used to create a pile of rubble and broken parts of a ship. The Player model will be the same as that used in the Town Level.

Crashed Ship Level: A collection of cube objects to be used to form the corridor, perhaps slightly angled to show the scene is on the crashed ship. A few lights will be used here, one or two next to the door so the player is able to see, a red warning light to add to the atmosphere of the crashed ship and a directional light with low intensity so the scene is still visible in all areas. Some basic objects to act as fallen crates or ceiling. Shuttle model will be reused from first cutscene. Particle effect to create a smoke, like effect in some areas.

Inside Ship Level: This scene will just be an enlarged version of the shuttle used in the first cutscene so the seats inside are scaled to the player. Lights will be set up to best suit the scene.

Police Level: Items such as a reception desk, chairs or water holder will either be downloaded from online to save time or made from basic shapes. The quality of these items isn't as high a priority as that of the functionality of the game.

Final Level: If scaled correctly, cubes can be resized and used as far away buildings for the scene with lights used to act as windows on those buildings. The main scene for this level will be in an abandoned skyscraper, this can be made by using the 3D shapes Unity provides and must be made in such a way to work with the idea of an interactive cutscene, for example enemies or objects can't be placed so far apart that during the cutscene they have to move fast to compensate.

Adobe Fuse CC will be used to create all characters in the game as provides a simple and easy way to create numerous characters with a professional look.

3.3.4 Player Interaction

As mentioned previously, the story element must be balanced out with player interaction and gameplay elements. In four of the planned levels the player will be able to walk and explore the scene with the ability to interact with certain objects or NPCs. Some objects placed in a scene will be able to be picked up by the player which may influence the game at another point. The player will be able to talk to NPCs of their choice when exploring a scene. A multiple-choice dialogue would be a great addition but as mentioned in the research section it is best to start small to avoid the problem of adding so many features it becomes difficult to work with, especially when this is a first project on this scale. There will be choices at certain points where the player must decide between two paths to progress the game. These will be displayed as buttons and once clicked will run their specific

code. Colliders will have to be used to determine is the player is near enough to an object to interact with it and these colliders will act as triggers for the dialogue event.

The final level of the game will interact with the player in a slightly different way in that the player will have to press the buttons displayed onscreen to activate the animations during the fight sequence. This adds some variation to the games gameplay.

3.3.5 Level Variation

First playable scene (Town Level) and the Police Level will both be in third person, the camera will follow the players position and rotation.

In the Crashed Ship Level the camera will only follow the player on one axis but the player controls will remain the same as the Town Level. This is to make it look as if the camera is moving down the corridor and avoid the camera clipping through the walls and objects as this scene is a small space.

The final level will be unique in comparison to the other levels as it is more of an interactive cutscene where the player will press a sequence of buttons to activate animations that continue the cutscene. If there is time it would be good to make this scene so that when the player gets a button wrong they are sent back to a previous checkpoint, but implementing a checkpoint system may be time costing.

3.3.6 Style

It will take a lot of time to try to make the game look realistic. It may be a good idea to add an art style to the game to make it look more professional without the extra workload for realism to allow for work in other areas of the game. Some art styles that may suit the game could include cell-shading, where there is an outline around objects. This style has been used in various games including some Telltale games and the Borderlands series. Some other styles could include a Block like style similar to Minecraft or a cartoon effect.

3.4 Extra Features

Using a keyboard, music could be recorded once the game itself is complete. It is not a prioritised feature for this project but would be a good addition to have some background music to listen to while playing. Voice-overs for dialogue would also be a good feature to include in the game, it would add to the characters and the immersion of the game. Sound effects would be another way to improve immersion, for example, wind sound during the town level, footsteps when walking or a blast sound when taking of in the ship. A load game option on the main menu is not necessary but would be something I would like to add to the game so the player can continue their story from wherever they left off. This could be done by creating a text file when a new game is started and add text to the document whenever the player makes influential choices, picks up objects or changes scene. A script to read the document could read it line by line and if it was the same as the comparison text the game would be able to know the players story progress.

4 Implementation

This section runs through what work was accomplished each week, any problems encountered and an explanation of the main algorithms used.

4.1 Work Journal

Week 1 (8th February – 14th February)

Planning and Preparation. Project planning done. Rough timetable planned for project tasks to keep on top of work, each task has a deadline. Attended extra optional lecture on the Unity 3D game engine and how it works. Installed Unity, the Kinect Cinema Mo-Cap asset and the Kinect 2.0 SDK. Watched various Unity 3D tutorials and practiced using Unity to make some basic scenes, explored with physics and experimented with the various components that can be used on objects.

Week 2 (15th February – 21st February)

The adapter arrived this week. The Kinect was set up using the adapter and practiced using it with the Cinema Mo-Cap asset. Although I was able to understand how to use the software and record some animation, the animation only worked with the model that comes with the Mo-Cap asset and I was unable to map it to another model due to the skeleton difference. A workaround to this will have to be found at a later point. The Menu for the game was designed and implementation of it begun.

Week 3 (22nd February – 28th February)

Menu was completed and the first cutscene was started. The scene for the cutscene was made. Found a suitable free space image to use as the skybox. Two different free models added to the game found online to act as spaceships. Two Spheres added to scene to act as Planets. Not much work done this week as I worked on completing the assignments for other modules so I could concentrate fully on this project.

Week 4 (1st March – 7th March)

Learned about animation in Unity, luckily quite similar to how animation works in Blender from the graphics module on the course. First cutscene finished. Animations added to ships and the camera. The Town Level scene was started, plane added as ground with a custom material used to try to make it look like sand. An image of western-style land used as the skybox for this level.

Week 5 (8th March – 15th March)

Walking mechanics added to the game along with camera movement mechanics to follow the player as they move. Tried recording a walking animation and applying to the player model but the skeleton rig didn't match up. Started coding a dialogue system with help from a few YouTube tutorials.

Week 6 (15th March – 21st March)

Basic humanoid models used for other characters as a placeholder for now. Colliders added to the NPCs and when the player walks into them a button appears. When the button is clicked it will show the dialogue for that character. It would've been better if this was made so that a press of a keyboard button would activate the dialogue instead of an onscreen button but I found the onscreen button a slightly easier method and with being slightly behind schedule I decided to stick with what was done.

Week 7 (22nd March – 28th March)

Code added to the end of the town level so that when the Player finishes talking to the NPC named Wendy, they will be shown a small cutscene of a ship crashing in the distance. This is followed by some more dialogue and once that dialogue finishes another panel is displayed with two buttons for

the player to make their first story choice. The buttons then load the scene corresponding to the button chosen by the player. The Debris scene was made, this didn't take long. The Crashed Ship Level scene has started to be made.

Week 8 (29th March – 4th April)

The Crashed Ship Level scene has finished being made. NPC and interactable objects added. Similar to interaction in the Town Level, when the player triggers a collision with the object, a button will appear to allow the player to activate an interaction with it. The hallway door is made in such a way so that it is closed to begin with, when the player tries to open the door without talking to the guard closest to the door, the door will not open and a message will be displayed in the console log. If the player talks to the guard, a Boolean in the guards' script will switch and the script of the door has a Boolean that matches the value. When the Boolean in the door script changes the player will be able to enter through the door. When the player walks through the door, the camera changes to show another room. In this room the player can walk to the ship, climb up and end the level. A cutscene plays for the end of the level which shows the players ship flying away from the crashed ship.

Week 9 (5th April – 11th April)

The player controls are of the same script as the Town Level but the camera controls are different, although similar, to the camera control script of the Town Level. The camera moves forward as the player does but at a slightly slower speed so that the player gradually walks further ahead. This is just to add a cinematic-type effect to the level. A light has been added above the doorway and a red light in the middle of the hallway ceiling. The red light has a script attached to it to flashed after a set time. A spot light was later added with a random flickering script to look like a broken light. A problem occurred with the laptop the game was being developed on and had to be reset. Time was lost this week resetting the laptop, reinstalling the required applications to continue the project and catching up on the lost work. Luckily the project was backed up recently so the amount of work lost was low. With time getting on and being slightly behind schedule, the Debris level will be left for now and returned to at a later time.

Week 10 (12th April – 18th April)

Inside Ship Level has been made. A few scripts added with timed events to activate camera switches and dialogue initialisation creating a basic sort of interactive cutscene. This level is more about developing the story rather than gameplay. The decision was made here to remove the Police Level as reaching this stage took longer than expected and with only three weeks left it is important to reach the end of the game and tidy up what has been down so far. The story will change slightly to compensate.

Week 11 (19th April – 25th April)

Character models made using Adobe Fuse CC and used to replace the placeholder models in the game. Mixamo website was used to add skeleton rigs to the models. Mixamo also provides a small set of free animations, numerous basic animations were downloaded from Mixamo to use in game. This included the walking animation and idle animation for the Player model. The idle animation is the default animation in the animator for the player, whenever the player is moving forward the Boolean parameter in the animator becomes true and the animations transitions to the walking animation. When the player stops walking, the Boolean returns to false and the idle animation plays. Various animations have been added to the NPC's in the Town level, this instantly makes the town seem livelier and on the Crashed Ship Level some animations were added for the injured guards. No animations have been added yet for the Inside Ship Level as they will be recorded with the Kinect along with the Animations for the next level.

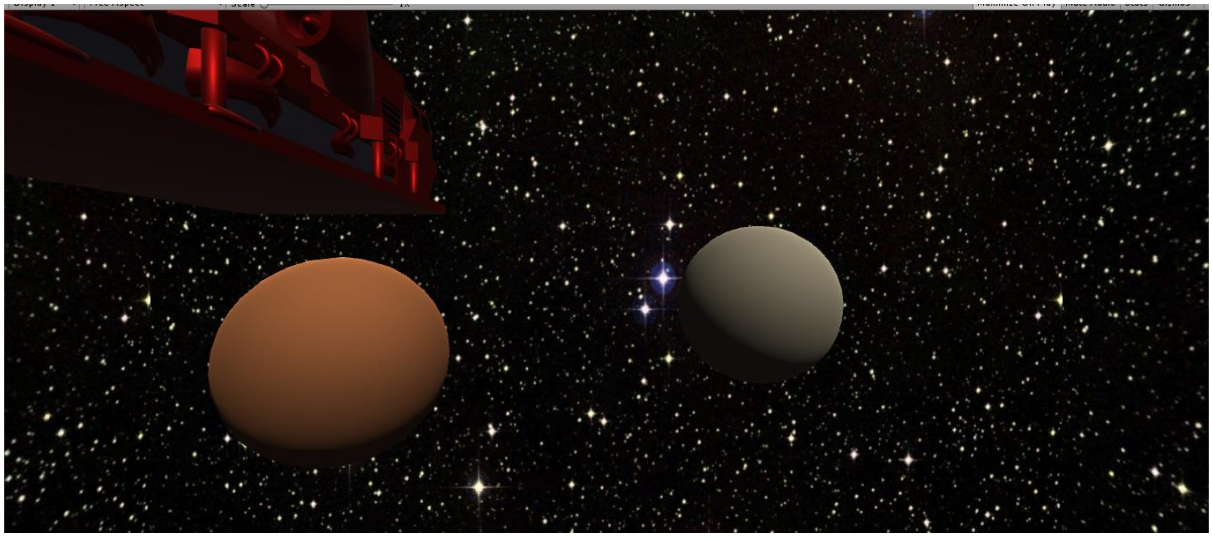
Week 12 (26th April – 4th May)

Final Level scene created. Prioritising report over end level of game. Final level work continued, recorded animations. Had trouble with applying motion capture to other models, as this is the final week I decided to use the humanoid model provided by the mo-cap asset just to make sure I could add motion capture to the game as this is something I have wanted to do for a long time. The story of the game has changed a little to compensate for this. Motion captured animation is used in the Inside Ship Level and the Final Level. Issues with the final level as I could not get the script to read for the users input with the amount of code within it but after working on this part for so long I decided to split the code into separate scripts, although this doesn't seem efficient.

4.2 Screenshots

Main Menu

First Cutscene



Town_Level

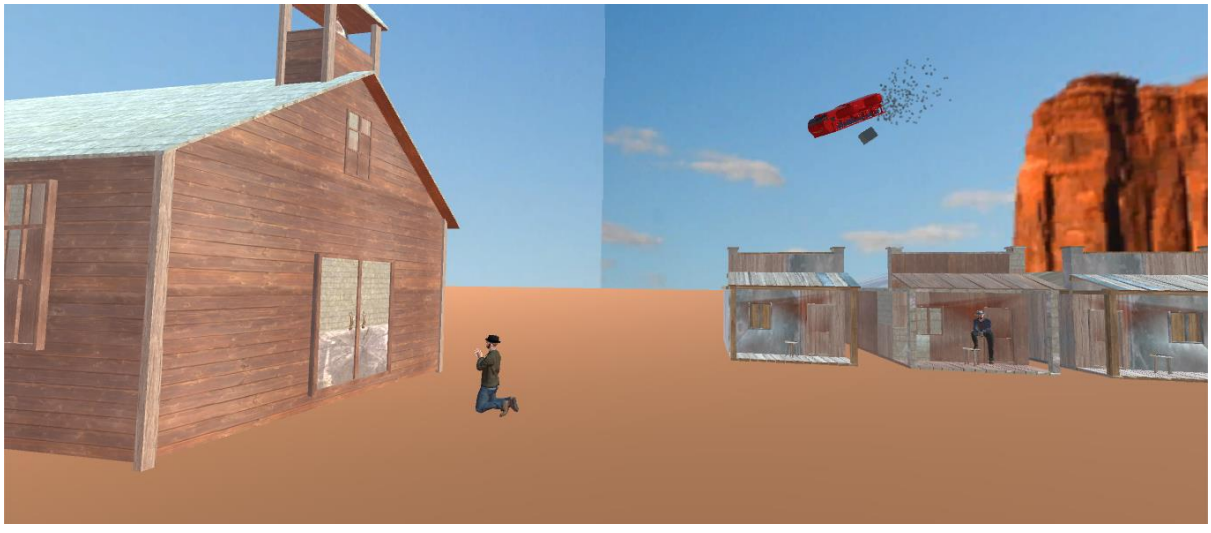
The Town.

Button to interact with NPCs.

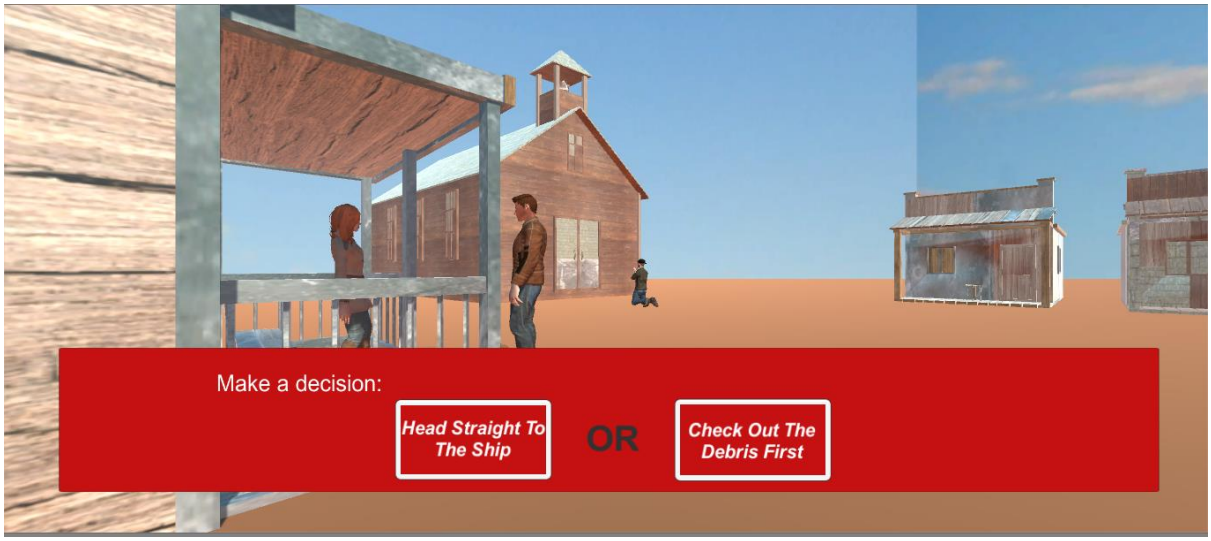
Dialogue Panel.

Walking Animation.

Ship crash cutscene after talking to NPC Wendy.

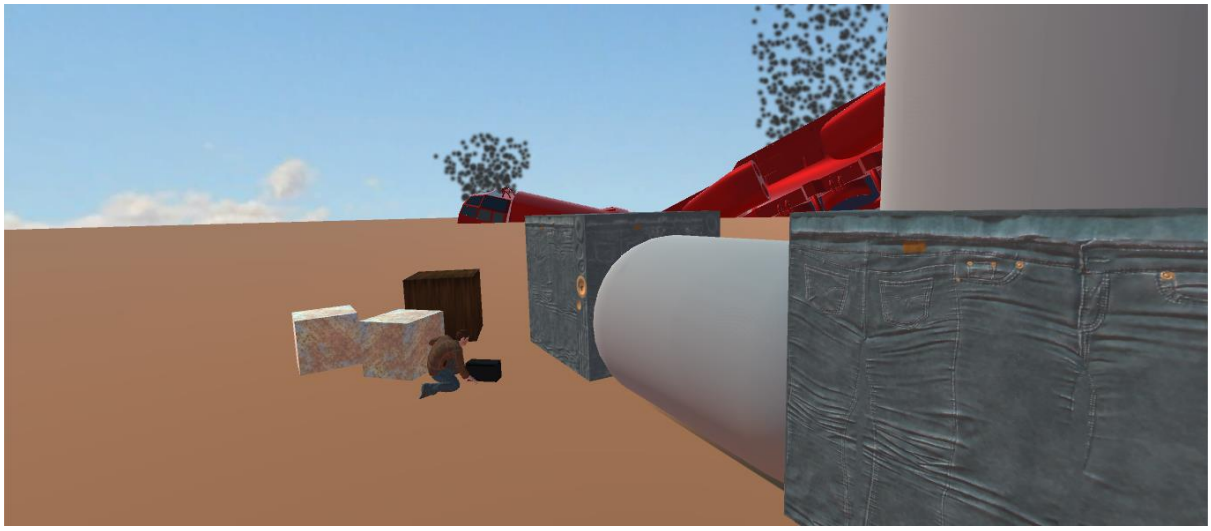


Choice between levels.

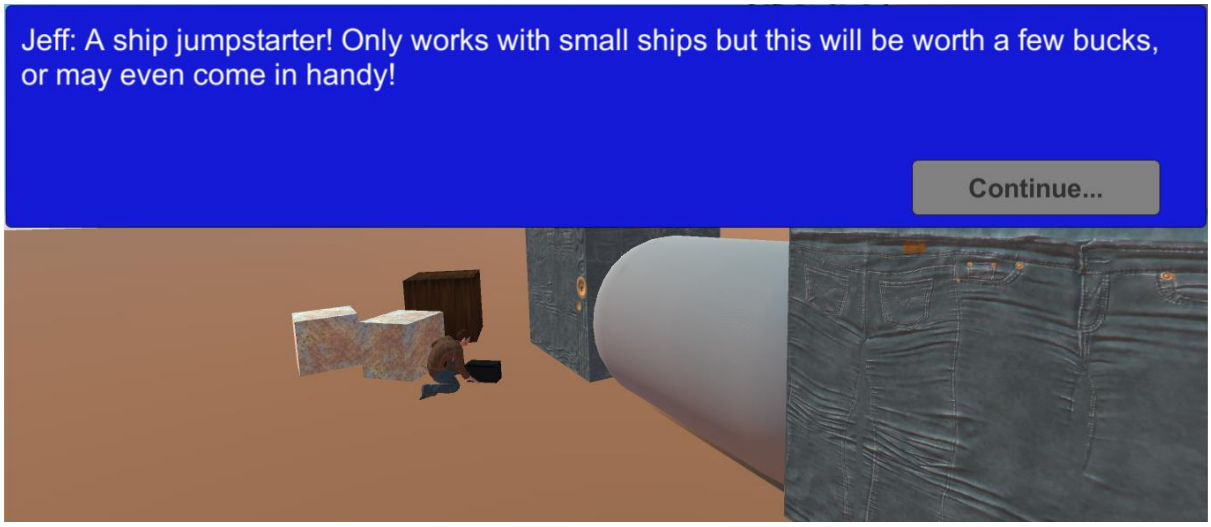


Debris Level

Cutscene to find jumpstarter.



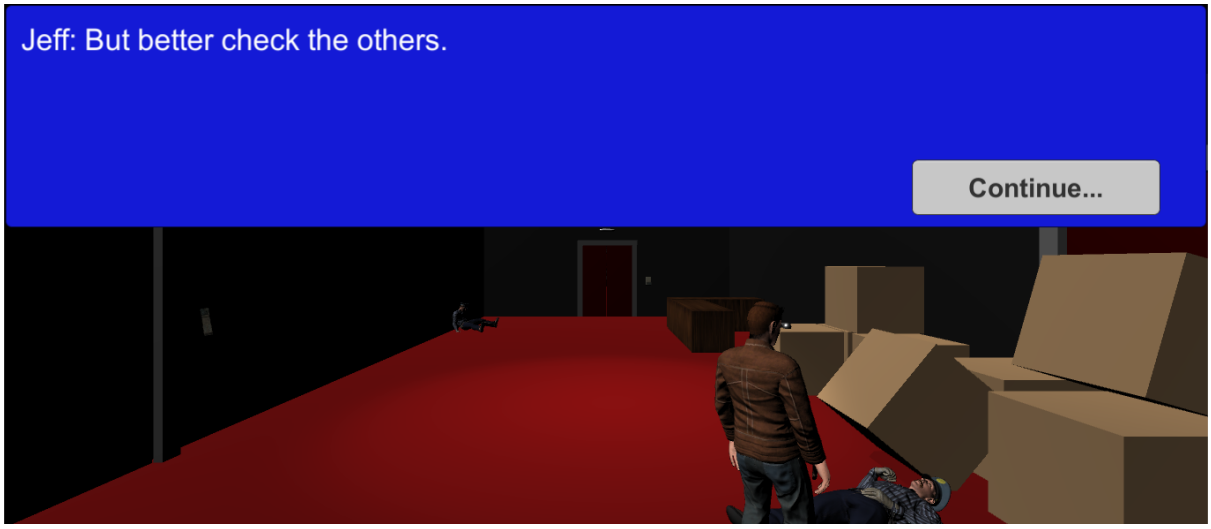
Dialogue in this scene is above to provide a better view.



Crashed Ship Level

The scene.

Dialogue for scene.

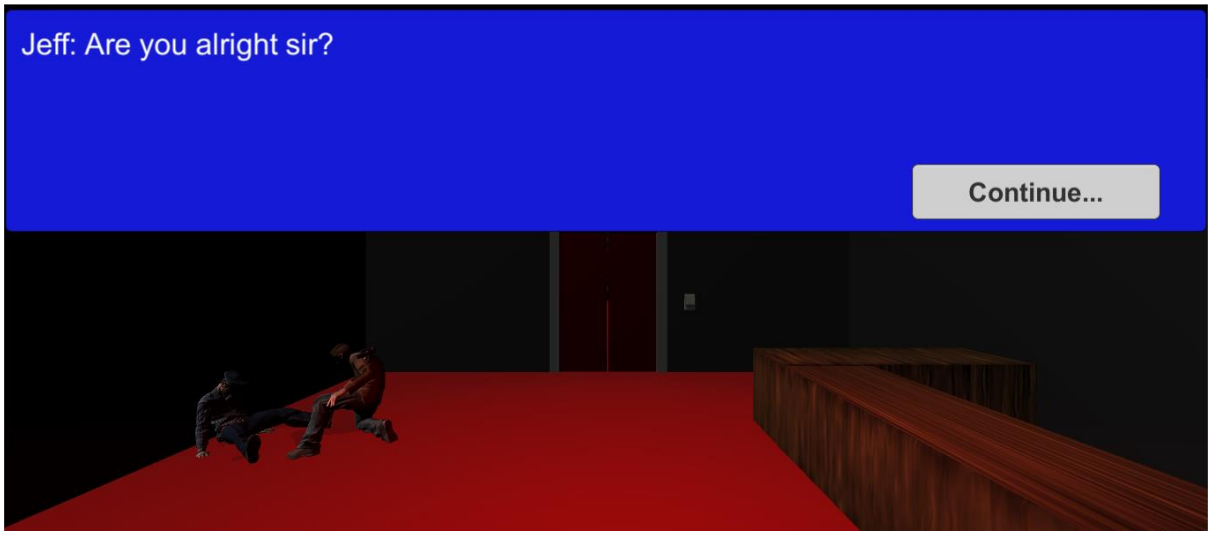


Picking up gun animation before dialogue.

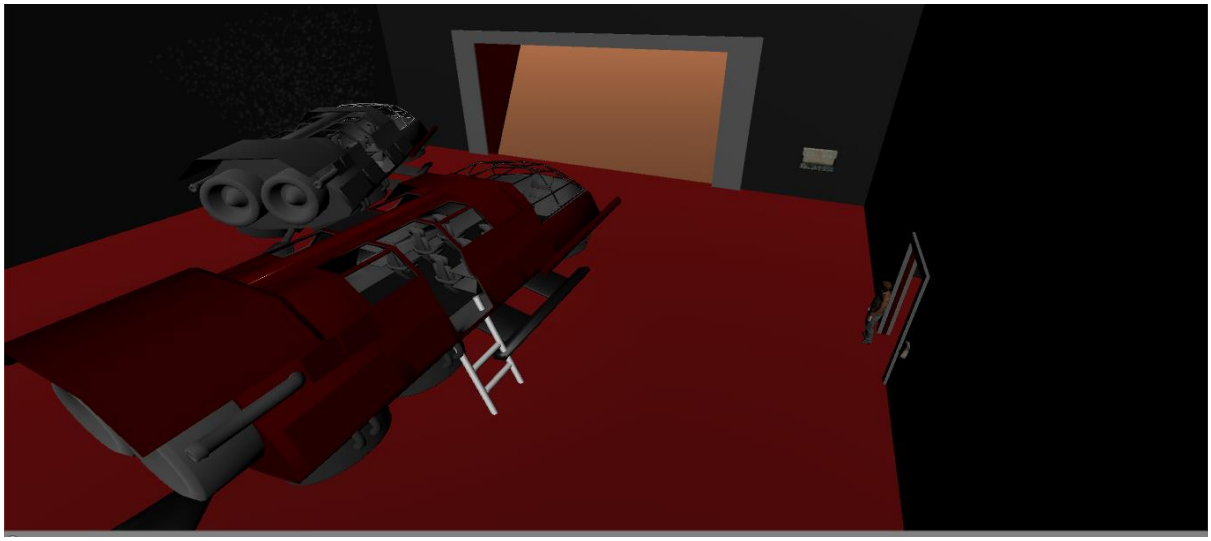
Gun added to model after being picked up.

Cannot walk through door without talking to guard first.

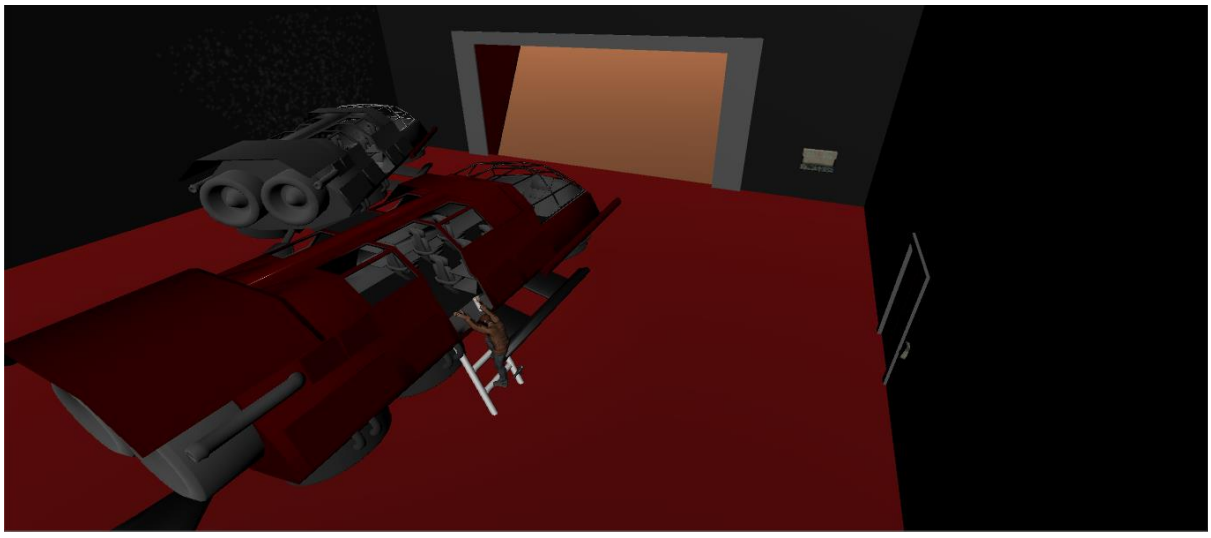
Crouching animation when talking to guard.



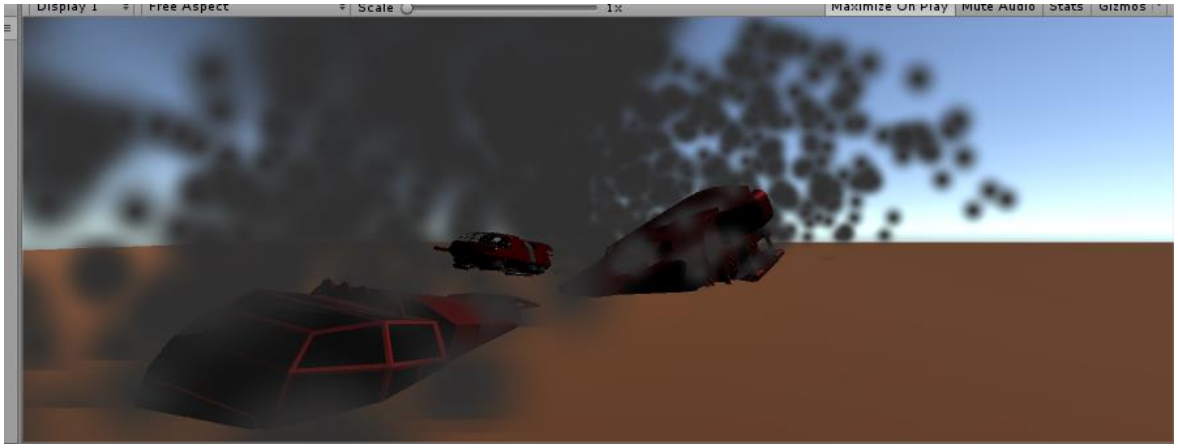
Door then opens and lets player through to hangar.



Climbing Ladder to finish level.

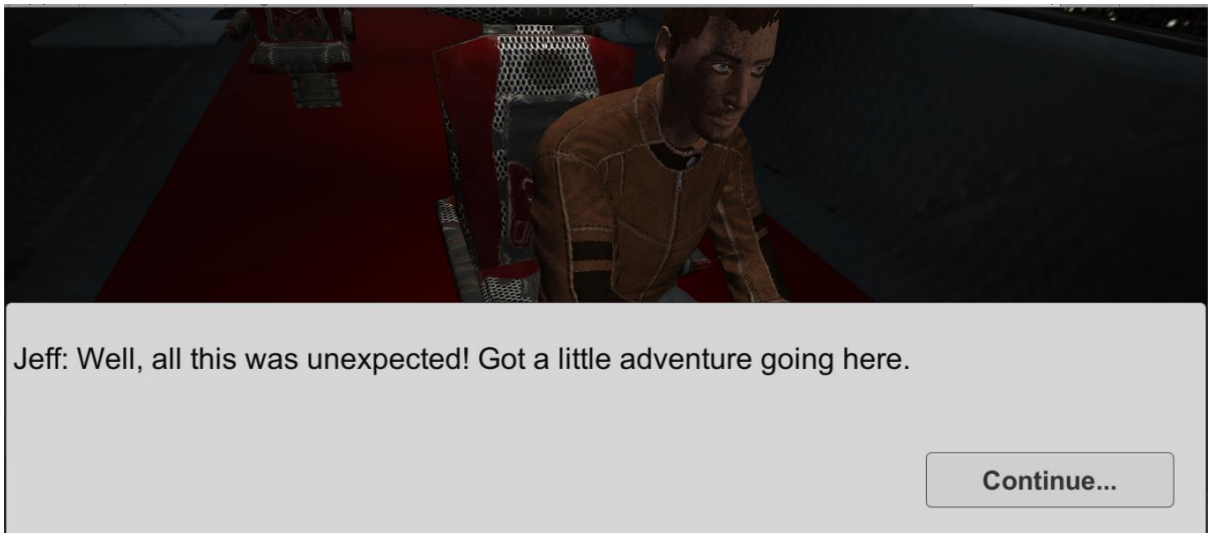


Small cutscene before switching levels.

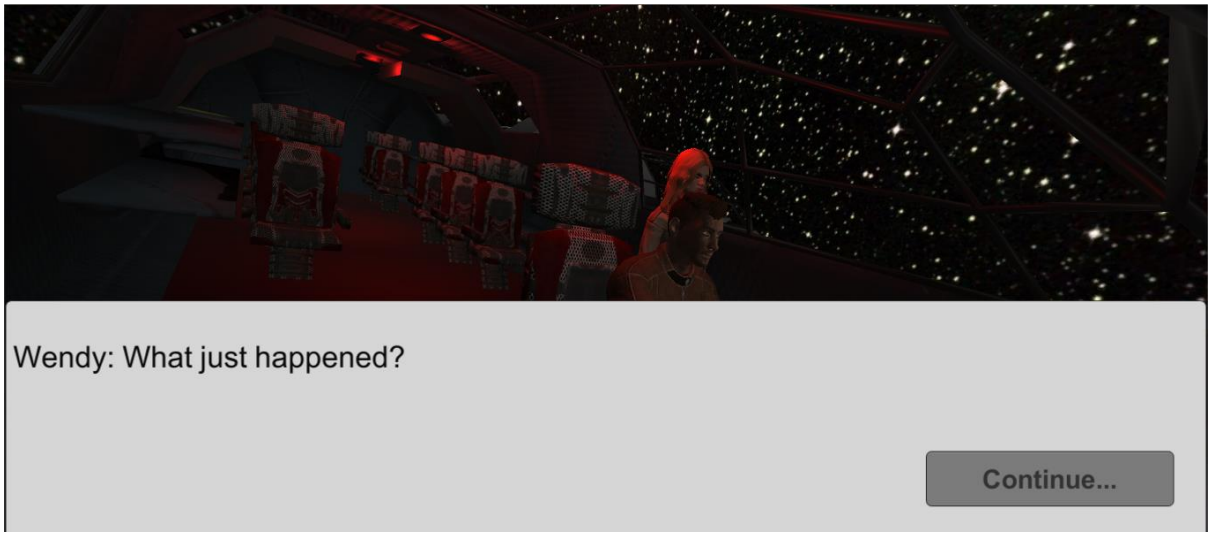


Inside Ship Level

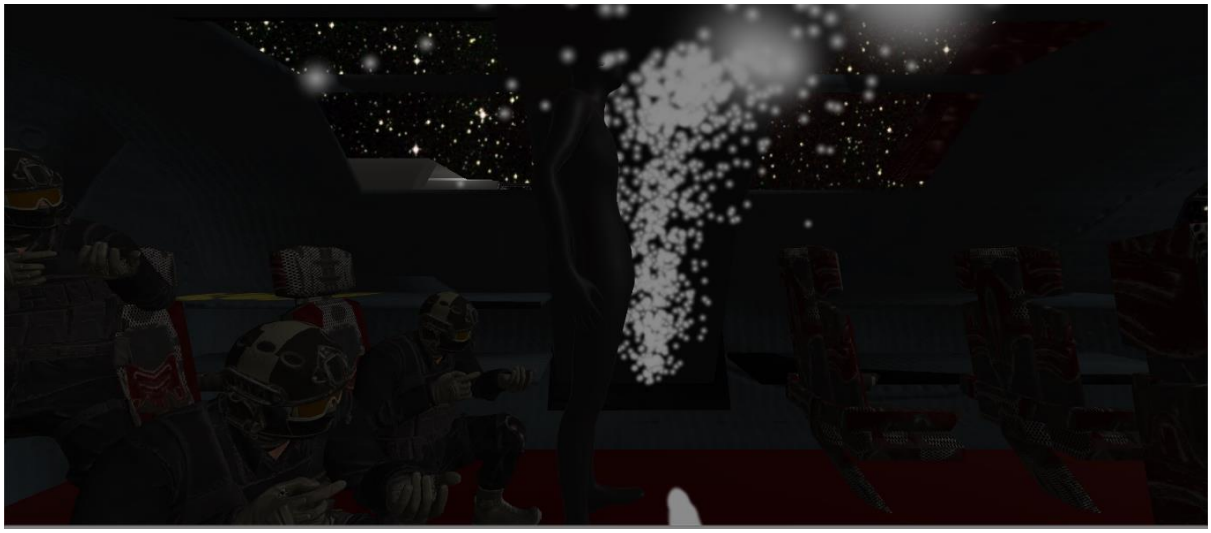
Dialogue back at bottom of screen for better view. Every major scene change, colour of dialogue panel changes. (Red in Town Level, Blue in ship levels, White in Inside Ship Level).



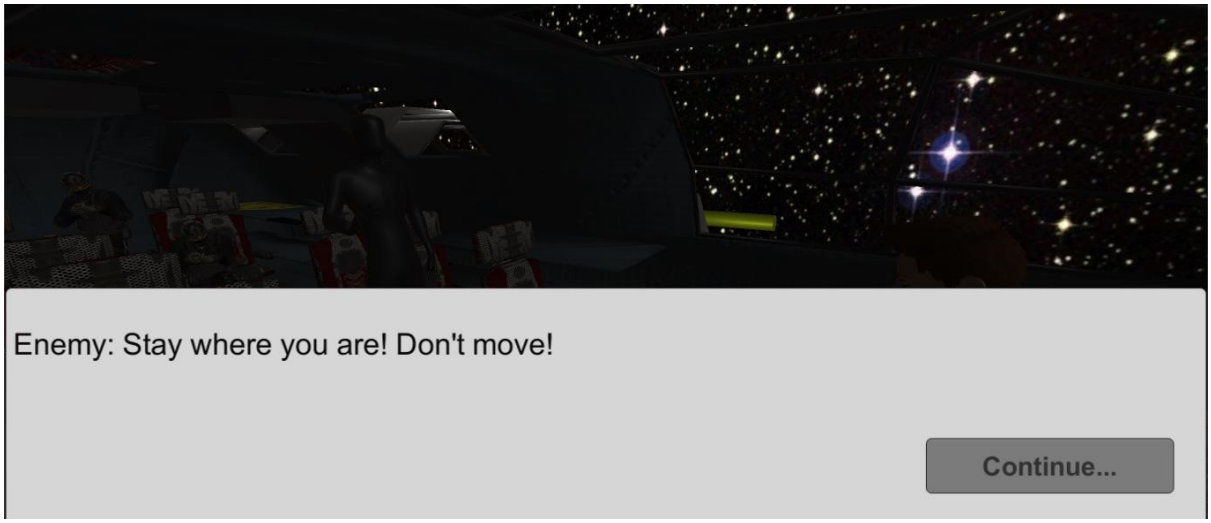
Switching from white lights to red flashing lights.

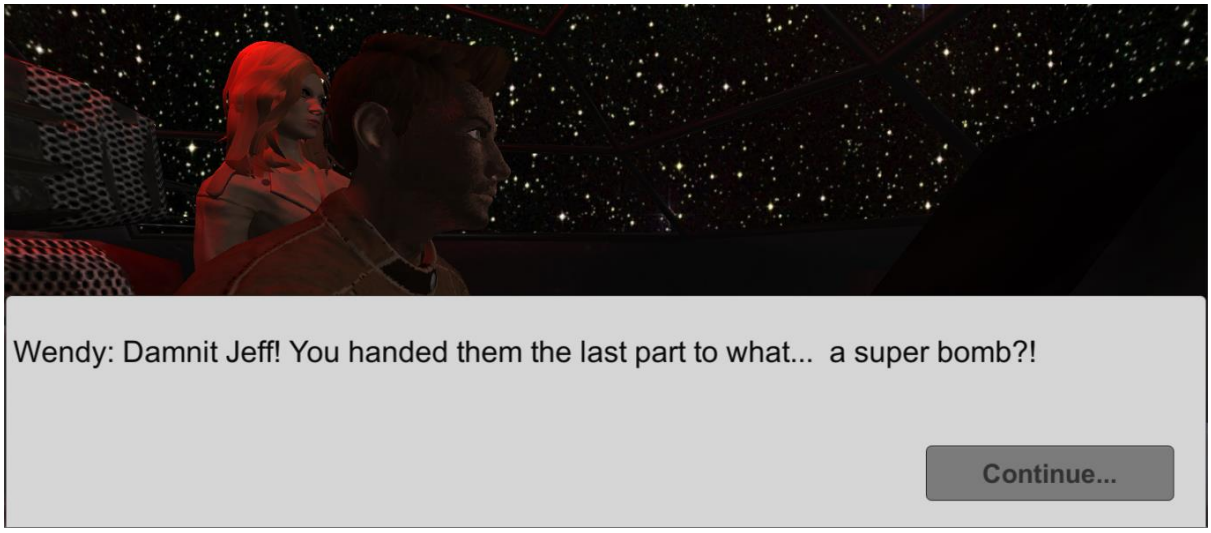


Smoke effect with enemies spawning behind it to act as a breach.

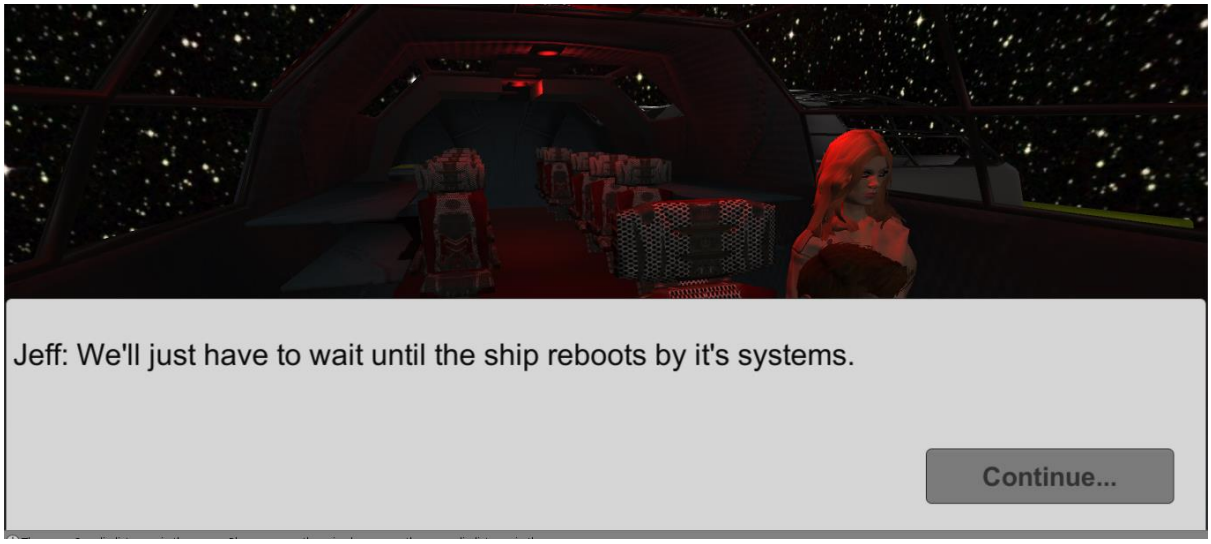


Conversation with main enemy. Enemies model using motion capture for this scene.



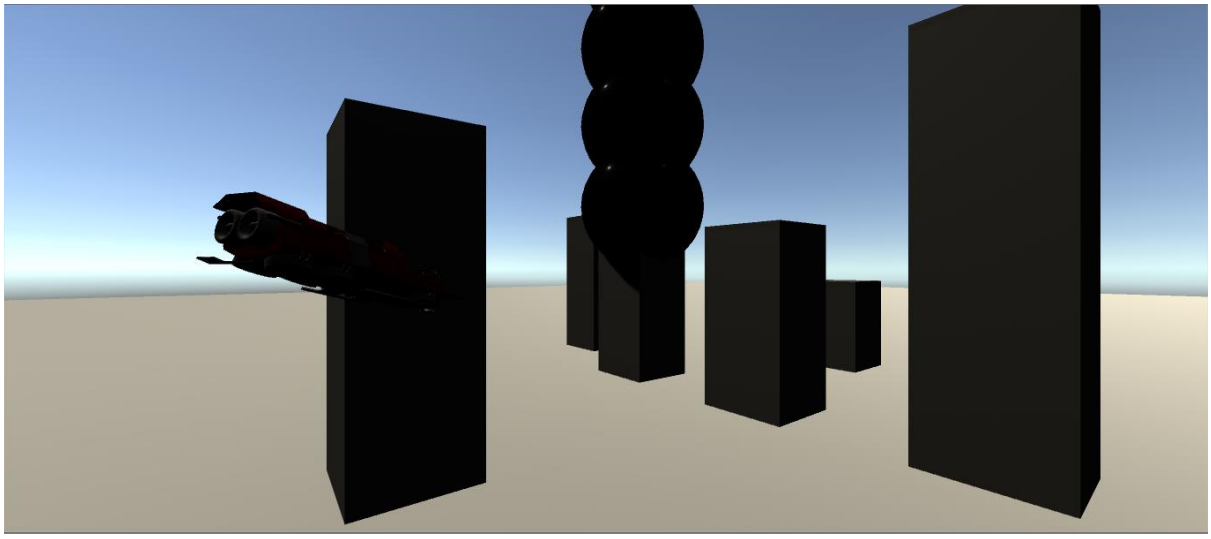


If the player passes through the debris level, they will have a jumpstarter and a different dialogue will be shown. If they do not pass through the debris level, this dialogue will show.

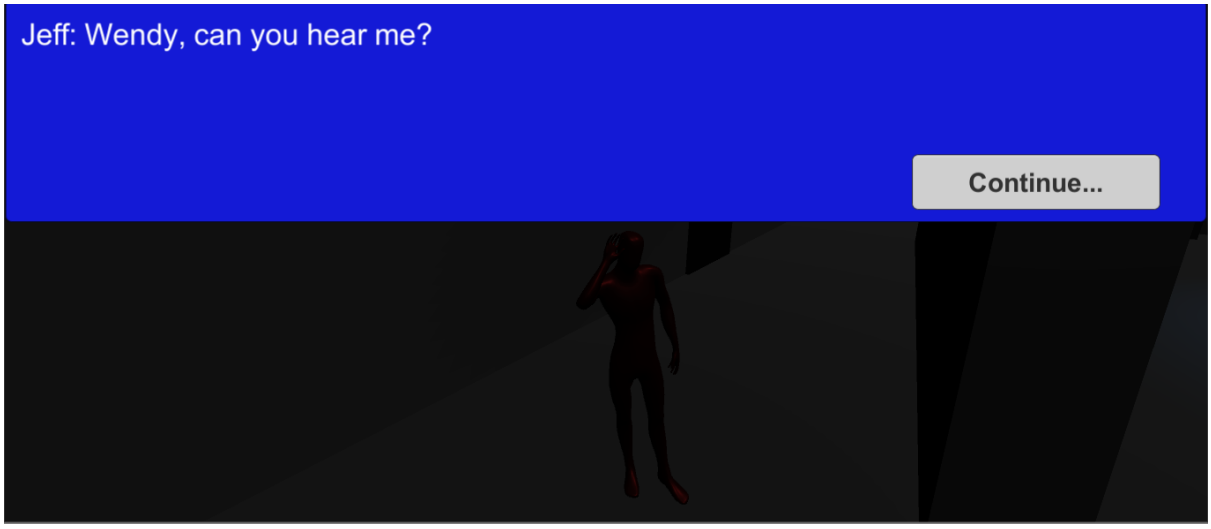


Final Level

Intro cutscene of ship flying through a city above the clouds**.**

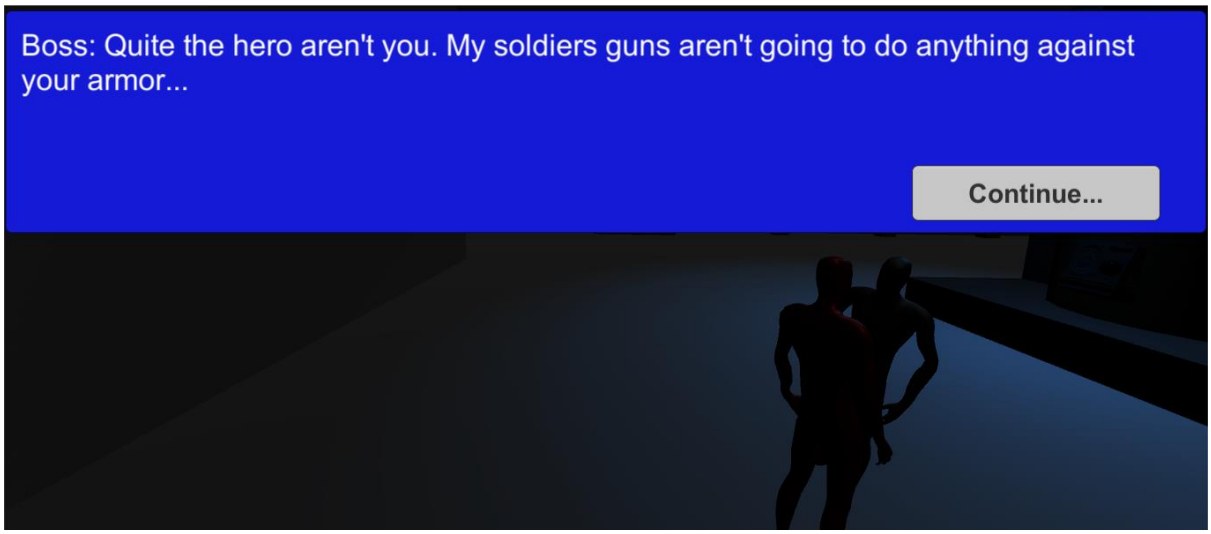


Motion capture used for Player model and enemy



The fight sequence

In a dim lit room, the fight for the “superweapon” is about to take place



5 Testing

5.1 Strategy for Testing

Tests were executed as the game was being made. Whenever a major addition was made to the game, it was tested and previous game functions would sometimes also be tested to ensure that any new additions did not break other functions.

5.2 Executing Tests

Test Number	Test Description	Desired Outcome/Functionality	Actual Outcome	Notes
Main Menu				
1	Menu 'Exit' button exits the game	Game closes	Game closes	No action needed
2	Menu 'Start Game' button loads the next scene	Next scene loads	Next scene didn't load	Scene name was misspelled. Name corrected
3	Menu 'Start Game' button loads the next scene	Next scene loads	Next scene loaded	No action needed
Cutscene				
4	Checking the animations of the cutscene	All model animations in sync and not model overlap	Ship models crossed through each other	Moved position of a ship in its animation
5	Checking the animations of the cutscene	All model animations in sync and no model overlaps another	All model animations in sync and not model overlap	No action needed
Town Level				
6	Walking forward and backwards. See Code 1 in Appendices	On press of forward button, move forwards. On press of backward button, move backwards	On press of forward button, moves forwards. On press of backward button, moves backwards	No action needed
7	Turning player	When player presses left or right buttons on keyboard, player rotates in that direction	Player did not turn with simple changing .rotation.set code	Looked into best ways to turn a player in game and looked into Unity code examples. Then modified the code in the 'turn' function in CharacterControl. Quaternion used
8	Turning player	When player presses left or right buttons on keyboard, player rotates in that direction	When player presses left or right buttons on keyboard, player rotates in that direction	No further action needed
9	Camera controls test Code 2 in appendix	Camera stays in offset position from player to give a 3rd person view. Follows the player and rotate as the player model does	Camera moves position but doesn't rotate. Player is also not visible on screen	Upon closer look, the offset postion of the camera to the player model has been modified

10	Camera controls test	Camera stays in offset position from player to give a 3rd person view. Follows the player and rotate as the player model does	Camera follows, player visible but doesn't rotate with player	Once eulerYangle is angled to look at player, the cameras rotation wasn't changed to it. Code added to do this
11	Camera controls test	Camera stays in offset position from player to give a 3rd person view. Follows the player and rotate as the player model does	Camera stays in offset position from player to give a 3rd person view. Follows the player and rotates as the player model does	No further action required
12	Walking into objects as player	Player should not walk through objects	Player walks through some objects	Found out this was because I forgot to add collision boxes to some of the building models
13	Walking into objects as player	Player should not be able to walk through objects	Player doesn't walk through objects	No further action needed
14	Checking that collision with player triggers animation of button to move onscreen	This script checks for if the object that's triggered it has the name of 'Player'. If it does then the button animation to put it on screen is run, done using a boolean	Button appears on screen	No action needed
15	On button click, button animation takes button offscreen	With the onClick script the button goes offscreen. This test helps by making sure the buttons offscreen during dialogue	With the onClick script the button goes offscreen. This test helps by making sure the buttons offscreen during dialogue	No action required
16	Dialogue panel	To display the dialogue sentences in the dialogue trigger script component	Dialogue panel didn't show at all	Concentrated on the code for displaying the dialogue that I forgot to add code to change Boolean and activate the animation to bring the panel on screen
17	Dialogue panel	To display the dialogue sentences in the dialogue trigger script component	Dialogue panel displayed and text matches that in the script component	No action needed
18	Wendy Dialogue Manager	Added a separate dialogue manager for Wendy NPC to trigger the end of the game once it finishes dialogue	Didn't work correctly, panel would show up at start of game	Added code to push panel offscreen at start of game
19	Wendy Dialogue Manager	Wendy dialogue manager to work	With code added to push panel off at start of game,	Unsure what this problem is caused by exactly. Switched Wendy dialogue

		correctly. Show panel when it should	the player is unable to move	to use the normal dialogue manager
20	Wendy Dialogue Manager See Code 3 & 4 in Appendices	Wendy dialogue manager to work correctly. Show panel when it should	Using the normal dialogue manager this works but doesn't end the level as the extra function isn't included	Added code to the normal dialogue manager so that if the dialogue name is equal to "Wendy" then it runs a new function that shows the end cutscene
21	End scene function in dialogue manager	A camera switch occurs, animation is played (this acts as a cutscene), using IEnumerator function and the available WaitForSeconds built in once a set time has passed the camera switches back to the main camera and triggers new dialogue	Camera switches, animation plays, camera switches back, but dialogue didn't show afterwards	Unsure what the problem is, could be due to the dialogue manager being used in its own script. Changed so that when the camera switches back, it switches to a third camera which has a script attached to it. When this script is run, the dialogue is triggered
22	End scene function	Once camera switched to third camera, dialogue should be displayed	Dialogue shown	No further action required
23	Checking NPC 1	When player walks into collider, display dialogue panel with correct sentences, at end dialogue close the panel	Works correctly	No action needed
24	Checking NPC 2	When player walks into collider, display dialogue panel with correct sentences, at end dialogue close the panel	Works correctly	No action needed
25	Checking NPC 3	When player walks into collider, display dialogue panel with correct sentences, at end dialogue close the panel	Works correctly	No action needed
26	Checking NPC 4	When player walks into collider, display dialogue panel with correct sentences, at end dialogue close the panel	Works correctly	No action needed
27	Checking NPC 5	When player walks into collider, display dialogue panel with	Works correctly	No action needed

		correct sentences, at end dialogue close the panel		
28	Checking NPC 6	When player walks into collider, display dialogue panel with correct sentences, at end dialogue close the panel	Works correctly	No action needed
29	Checking NPC couple	When player walks into collider, display dialogue panel with correct sentences, at end dialogue close the panel	Works correctly	No action needed
30	End scene choice panel display	Once the script run by the third camera has run through, another panel is to display two buttons	Panel shows	No action needed
31	End scene choice buttons to load correct scenes	When the choice panel is displayed and the play presses a button they are taken to the correct level	Player is taken to the correct level	No action required
Debris Level				
32	Debris Level on start, camera animation	Camera animation should play instantly and pan across the scene	This works, camera animation starts on level start	No action needed
33	Debris Level, after camera animation finishes, the player model animation should play	Players animation plays and they walk over to the item	Play animation plays	No action needed
34	After player model animation has finished, dialogue displays	Dialogue should display notifying the play what they have earned from choosing this scene	Dialogue shows but panel is slightly off screen	Panel position moved in animation key frames. Re-tested and works
35	During dialogue check	Animations should stop and dialogue is visible	Dialogue visible but player model animation was looping	Changed animator setting to not loop the animation
36	During dialogue check	Animations should stop and dialogue is visible	Works	No further action required
37	At end dialogue, switch to crashed ship level	Load crash ship level once the dialogue queue in the debris level reaches 0 sentences	The crash ship level loads correctly	No action needed

Crashed Ship Level				
38	Spot light should flicker randomly. See *Code 5 in appendices*	Light should flicker randomly within a set range of time	The flickering intervals are too long	Random time range modified to low numbers
39	Spot light should flicker randomly	Light should flicker randomly within a set range of time	Light flickers as wanted	No further actions required
40	Red flashing light to flash at a set time interval to act as a warning light	Light should flash every half a second	Light flashes every half a second	No action required
41	Player controls, although using same script as that used in town level, should be tested again in this level	Player moves up and down with arrow-up/arrow-down or W/S. Left or right with arrow-left/arrow-right or A/D	Player controls work correctly	No action needed
42	Camera controls should follow the player but only on one axis to move up the corridor	Camera follows player on one axis	Camera doesn't follow at all	Changed the script so the camera doesn't follow the player but moves as the buttons are pressed
43	Camera controls, with the change of script I was now able to make it so the player can edge ahead of the camera and gradually walk further ahead	Camera moves slightly slower than player and only on one axis, no rotation change	Camera moves slightly slower than player and only on one axis, no rotation change. However, when player turned around they could walk past the camera and get lost	Collider added to the camera so if player model collides with the camera it stops them from walking past
44	Camera controls retest	Camera moves slightly slower than player and only on one axis, no rotation change and player shouldn't be able to walk past camera	Camera moves slightly slower than player and only on one axis, no rotation change and player can no longer walk through camera and player walks gradually further ahead to add a more cinematic look	No further action required
45	Interaction with guard	Using similar dialogue manager and dialogue plan to the town level. Button should display on colliding with object then once clicked should show the dialogue panel and close once ended	After clicking the button, dialogue panel did not pop on screen	Found out this was because I had not assigned the correct panel to the dialogue manager and so the animation boolean was different
46	Interaction with guard	Button should display on colliding with object then once clicked	Worked correctly	No further action needed

		should show the dialogue panel and close once ended		
47	Interaction with gun	Dialogue panel to notify player of what they have just picked up and the laser pistol to be added to the players model	Dialogue shows and pistol added to back of player model	No action needed
48	Interaction with objective guard	When colliding with objective guard, button onscreen, on button click the dialogue manager runs as usual		
The trigger script for when the player collides with the objective guard contains code so that once the player clicks the button to start the dialogue manager, a Boolean is changed. The Boolean is stored in the script of the objective guard/talk_to_guard script and the door script reads this. The door checks this Boolean and based on its value, opens the door or leaves it closed.				
49	Interaction with door before talking to guard	Door reads false value from Boolean and remains closed	Door remains closed	No action needed
50	Interaction with door after talking to guard	Door script reads true value from the talk_to_guard class and opens the door	Door opens	No action needed
51	On Camera switch, activate animation to open ship door	Door should lift on the shuttle ship	Ship disappeared	The ship contained the animator of the ship from the cutscene, changed it to the new animator
52	On Camera switch, activate animation to open ship door	Door should lift on the shuttle ship	Door didn't open	Tried various methods including making the door a separate object altogether but ended up just removing the door as too much time was being spent on this issue
53	On Camera switch, ship door is open	Doorway is open	Door is open	No further action required
54	Climbing on to ship	Player moves up ladder to get into ship	Player does move up ladder but another force from the shuttle is pushing the player away too	This issue is still in the game, I could not find the source of the force
55	Once the ladder has been climbed, switch to another camera and start shuttle animation	Once the player climbs up the ladder (timed function), the camera switches and an animation of the shuttle flying past the camera starts	All works fine, just some camera position adjustments to make	Camera position adjustments made

56	Load next level	Load the next scene	Next scene loads	No action needed
Inside Ship Level				
57	In start function of camera, dialogue starts	Dialogue appears at start of scene	Dialogue appears	No issues
59	After the dialogue ends, switch camera and start new dialogue with NPC Wendy	Dialogue and Wendy model visible	Dialogue and Wendy model visible	No issues
60	After 2nd dialogue	change camera, wait, disable lights, wait, activate red flashing lights, wait, start third dialogue	Works but waiting time needs to be tweaked	Wait time lowered
61	After 2nd dialogue	change camera, wait, disable lights, wait, activate red flashing lights, wait, start third dialogue with appropriate waiting time	Works fine	No further action required
62	After light event	Switch camera, start new dialogue (3rd dialogue), start camera animation to show ship, camera animation speed set to 0 to stop it in the correct position after a set time and enemies breach	Works as expected	No action needed
63	Enemies breach	With camera in position, particle effect activates on floor, waits a few seconds to cover whole ship, enemies' models spawn in, smoke particles clear, switch camera	Works as expected but smoke particles are too slow and lifetime is too short	Modified particle effect settings. Particle speed is slightly higher, lifetime extended, and shape of dispersal changed to hemisphere
64	Smoke particle effect	Particle effect should start at correct time and cover the whole width of the ship before the particles stop spawning. Enemy models spawn while smoke covers ship, the models should not be able to be seen until the smoke clears	Works as expected	No further action required

65	After particles disappear, change camera	Change camera after particles disappear and enemies are visible	Works as expected	No action needed
66	Camera pans down corridor	Camera should pan down corridor and animation speed set to zero at end	Works as expected but camera end position needs to be adjusted	Camera end position changed
67	Dialogue to play	Dialogue plays	Dialogue plays	No action needed
Adding Character Models and Animations to All Levels				
68	Switched out all placeholder models with correct models	All NPCs have detailed model. Player has detailed model	All NPCs have detailed model. Player has detailed model	No action required
69	Idle animation (Player)	While not moving, plays this breathing animation	Animation plays	No action required
70	Walking animation	While walking, play this animation and when stop moving, go back to playing idle animation	Idle to walking worked but once walking the animation wouldn't stop after stop moving	Found out this was due to the walking animation being on a loop
71	Walking animation	While walking, play this animation and when stop moving, go back to playing idle animation	Idle to walking worked. Walking to idle worked	No further action needed
72	Player pick up animation when picking up gun (Crashed Ship Level) Code was adjusted to play the animation before the dialogue is shown	Pick up animation plays from both walking animation and idle animation	Works correctly	No action needed
73	Player crouch animation when talking to guard (Crashed Ship Level)	Made so that the crouching animation speed is set to 0 after two seconds and then the dialogue starts	Works correctly	No action required
74	Various idle animation applied to NPCs in all scenes	All idle animations to run correctly and in a loop	Works as expected	No action needed
75	Added walking animation for one NPC in Town Level	NPC should keep walking forward in walking animation	Walking animation runs through once but then stops	Experimented with the animation settings and changed loop settings
76	Walking animation for one NPC in Town Level	NPC should keep walking forward in walking animation	Walking animation runs through once but then stops	Removing the walking animation and giving the NPC an idle animation instead
Inside Ship Level Continued…				
78	Using a static Boolean in the same script in Debris Level and the Inside Ship	Pick up jumpstarter in debris level and see the correct dialogue if done so	Worked as expected	No further action required

	Level, it is possible to check if player has jumpstarter			
79	Using a static Boolean in the same script in Debris Level and the Inside Ship Level, it is possible to check if player has jumpstarter. *See code 7 in Appendices*	If player does not have jumpstarter, the correct dialogue should show	Worked correctly	No action needed
80	Load next level after all dialogue is finished	Load the final level	Loaded final level	Works Correctly, no work needed
Final Level				
81	Mo-Cap animation testing, should be able to map to any humanoid skeleton model	Motion captured animation to work on players model	Did not work correctly	Changed a few options in the animation settings and changed base model from humanoid to generic
82	Mo-Cap animation testing, should be able to map to any humanoid skeleton model	Motion captured animation to work on players model	Did not work correctly	Tried applying the humanoid rig provided by the Cinema Mo-Cap asset to the players model
83	Mo-Cap animation testing, should be able to map to any humanoid skeleton model	Motion captured animation to work on players model	Did not work	Change the mo-cap humanoids rig with the players rig
84	Mo-Cap animation testing, should be able to map to any humanoid skeleton model	Motion captured animation to work on players model	Did not work, Mo-Cap humanoid received position changes but no animation	Decided to use the humanoid model as the players model for the final scene and as the enemy in the last two scenes. Explained in story as futuristic armor
85	After set number of seconds, switch cameras	Cameras should switch	Cameras did switch	No action needed
86	Fight Sequence	Prompt to press button, part of animation runs then stops, next prompt appears, repeat until end	Used too many if statements within if statements. Didn't work correctly	Broke the code up into separate scripts
87	Fight Sequence	Prompt to press button, part of animation runs then stops, next prompt appears, repeat until end	Works correctly	No further action required

6 Critical Evaluation

This section reviews the project, evaluates the work completed, how well the final project has met the aims and goals and what improvements could be made to the project.

6.1 Aim

The aim of this project was to produce a game using a game engine that was story based and used animations recorded using motion capture.

The aim of the project has been met but could be improved. The game was developed in the Unity game engine and is story based. The dialogue is quite heavy and one of the major parts of the game. In some of the levels, such as the Inside Ship Level, the dialogue may be a bit too much and makes the game too boring which is one of the problems that I had planned to avoid.

Motion capture didn't work with the placeholder models due to the skeleton but did work with the new models. However, most levels are using free animations found online on Mixamo but the Final Level, Debris Level and Inside Ship Level use motion captured animations and so the aim of using motion capture in game has been met.

Gameplay elements have been added, this includes walking, talking to NPCs and picking up items. And player choice was implemented in dialogue once and through picking up optional items.

The aim has been met in all aspects but not to the quality that would have been preferred.

6.2 Objectives

Convey a story

This objective has been achieved. The game uses cutscenes, dialogue and the level scene to develop a story for the player.

Implement dialogue system

A dialogue system was implemented although basic. A multiple-choice dialogue was implemented but used only once to choose which level the player wishes to go to next. A fully implemented multiple choice dialogue where the NPC sentences are different based on what the player chose to say would have been preferable but this was not accomplished during this project.

Use of motion capture

Although motion capture animation wasn't the only animation used, the amount used in the Final Level is enough to have achieved this objective and some motion capture animation was used in the Inside Ship level.

Gameplay elements

Walking, exploring and interactive are important features used in the Town Level and Crashed Ship Level. Player interaction in the final level is done by the user pressing the correct buttons that show on screen to allow the animation plays out.

Player choice implemented with noticeable differences

The choice the player must make appears when they have the chance to choose whether to travel to the crashed ship or head to the debris first. If the player chose to travel to the debris then they pick up a ship jump starter which is used in the Inside Ship Level as part of the story to get the ship working again. If the player chose to travel straight to the crashed ship then they would not find the ship jump starter and so the story of getting the ship working in the Inside Ship Level is different. I

believe this choice implementation achieves this objective but it would have been preferable to have had more decisions like these added into the game.

6.3 Design

The final design of the game followed closely to what was planned during the design section. A few small changed were made along the way to improve the game such as moving the Exit button on menu sue to it blending into the background where it was originally positioned. Another change made was to place the dialogue panel in crashed ship scene and debris scene at the top of screen so characters are more visible, especially the player.

The police level of the game was dropped as I struggled tackling a few problems in the Town Level as I was getting to grips with using Unity and coding in C#, but believe this was for the better as I could concentrate on the fewer levels and complete them with a better understanding of how the more complex parts of the code works.

6.4 Project Management

The first half of the work timeframe I believe was managed poorly, I spent too much time learning and experimenting with Unity on only one level of the game. After the mid-project demonstration, I realised that time management was an issue and was disappointed in progress of project at that point but I rearranged the time plan and added extra time to make up for lost work. I am happy with what I have managed to make in the time we've had to make this project and the learning experience that came with it.

6.5 Testing

Test were executed on the game sometimes without even realising. When adding new functions and scripts it's a habit to test them at each functional checkpoint to make sure it does what it is meant to do. If there was more time to work on the project now I would let various people play the game to see what faults and problems in the game need prioritising over other. It would also be good to know what features are good for the game and which may not suit it.

6.6 Improvements to the Game

The game created in this project is nowhere near the standards of a professional game released these day but for the timeframe of the project and my knowledge of making games at the beginning of the project I believe this final game is a good start and when I try to develop more games or apps in the near future I will have the knowledge to start it comfortably.

Some improvements that could be made to this game are listed below:

- **Multiple choice dialogue added in to ordinary dialogue rather than just major decisions.**
  One of the main features this game is missing is the variation in dialogue. Most major story based games have some form of multiple choice dialogue system.

- **Better story.**
  I'm not a story writer and the story for this game is basic and captivating, a good story heavy game needs that captivating story to keep the players wanting to know what happens next.

- **Improved motion capture.**
  Using the Kinect for motion capture was fun and useful for a beginner of motion capture such as myself. However, noticeable random movements and shaking occurs when using the Kinect as a motion capture sensor. This can't be dealt with by either using an improved

motion capture kit or by tidying the motion captured animation after capture but can take a while. One of the main problems in the motion capture of this game was the fact that the animations would not map to any other model other than the ones that had exactly the same rig as the default. Animation retargeting is an option and I am open to learning about it but for this project there were other aspects I had to focus on, especially closer to the deadline which is when most of the animations were recorded.

- **Add a style to the game**.
  A game like this would be best suited to an art-style. One idea could be to have the style follow a painting kind of look with brush marks over objects and low-quality textures or a popular style used in some games now is cell-shading.

- **Add music to the game.**
  Some background music to listen to in-game would help make the game a more enjoyable experience, especially in the walking/exploring levels.

- **Add voice over to the dialogue.**
  Having voiceovers for characters as well as the text dialogue would help make the game a more immersive experience.

Improvements to the Project

After working through this project once, I know how a lot of features work where a lot of time was spent during this project trying to understand them and that time would be better spent if this task was repeated. That being said, the learning curve for this project has been an enjoyable and useful experience as well as sometimes frustrating. What I've learned from this project will be used again.

The project could have been improved by the following:

Better Time Management
Poor time management before the mid project demo mean that there was some work to try to catch up with and this slowed down progress in other areas of the game and lead to the final level finishing at a lower quality to the rest of the game. Improved time management after the mid project demo shows that a plan and time management can help a great deal in accomplishing task efficiently, if this was done from the beginning the game may look more polished.

Narrower Scope
When drafting a plan, I would always think about extra features or levels that would be a good addition (e.g. a separate mission, flying a ship). Trying to figure out new features while the features already in the game were incomplete wasted a small amount of time also.

Efficiency
In the Town Level, each NPC has their own dialogue manager. However, this could have been reduced to only one dialogue manager had I only implemented it correctly. This was the first time the dialogue manager was used in the game and by the time I got it all working, I was behind the work plan.

Appendices

A Third-Party Involvement

Kinect Motion Capture (Cinema MoCap)

More information on Cinema MoCap can be found here:
https://assetstore.unity.com/packages/tools/animation/cinema-mocap-2-markerless-motion-capture-56576

Adobe Fuse CC

More information on Adobe Fuse CC can be found here:
https://www.adobe.com/uk/products/fuse.html

Mixamo

More information on Mixamo can be found here:
https://www.mixamo.com/#/

B Ethics Submission

To my knowledge, no ethics form required for this work.

C Code Examples

1. Character Control

```
public class CharacterControl : MonoBehaviour {
    public float inputDelay = 0.1f;
    public float forwardVel = 30;
    public float rotateVel = 100;

    static Animator anim;
    Quaternion targetRotation;
    Rigidbody rBody;
    float forwardInput, turnInput;

    public Animator btnAnimatorEnd;

    public Quaternion TargetRotation
    {
        get { return targetRotation; }
    }

    void Start()
    {
        btnAnimatorEnd.SetBool("IsOn", false);

        anim = GetComponent<Animator> ();
        targetRotation = transform.rotation;

        if (GetComponent<Rigidbody>())
            rBody = GetComponent<Rigidbody>();
        else
            Debug.LogError("Prob");

        forwardInput = turnInput = 0;


    }


    void GetInput()
    {
        forwardInput = Input.GetAxis("Vertical");
        turnInput = Input.GetAxis("Horizontal");
    }

    //When the player is walking, sets bool to true which plays the
walking animation
    void Update()
    {
```

```
        GetInput();
        Turn();

        if (forwardInput != 0)
        {
            anim.SetBool("isWalking", true);
        }
        else { anim.SetBool("isWalking", false); }

    }

    void FixedUpdate()
    {
        Move();

    }

    //Takes forward input and multiplies it by the velocity value an
d applies the change to the rigid body
    void Move()
    {
        if (Mathf.Abs(forwardInput) > inputDelay)
        {
            rBody.velocity = transform.forward * forwardInput * forw
ardVel;

        }
        else
            rBody.velocity = Vector3.zero;
    }


    //Turns the model left and right
    void Turn()
    {
        targetRotation *= Quaternion.AngleAxis(rotateVel * turnInput
 * Time.deltaTime, Vector3.up);
        transform.rotation = targetRotation;
    }
}
```

2. Camera Control

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CameraControl : MonoBehaviour {

    public Transform target;
    public float lookSmooth = 0.09f;
    public Vector3 offsetFromTarget = new Vector3(0, 50, -70);
    public float xTilt = 10;

    Vector3 destination = Vector3.zero;
    CharacterControl charController;
    float rotateVel = 0;

    //Camera looks at the target, which is the player
    void Start()
    {
        SetCameraTarget(target);
    }

    //Checks if the target exists and if it does, finds it's charact
er controller component
    public void SetCameraTarget(Transform t)
    {
        target = t;

        if (target != null)
        {
            if (target.GetComponent<CharacterControl>())
            {
                charController = target.GetComponent<CharacterContro
l>();
            }
        }
    }

    void LateUpdate()
    {
        MoveToTarget();
        LookAtTarget();
    }


    //Sets camera position to the characters position and adds the o
```

```
ffset value (places the camera behind in this case)
    void MoveToTarget()
    {
        destination = charController.TargetRotation * offsetFromTarg
et;
        destination += target.position;
        transform.position = destination;
    }

    //Angles the camera to look at the character
    void LookAtTarget()
    {
        float eulerYAngle = Mathf.SmoothDampAngle(transform.eulerAng
les.y, target.eulerAngles.y, ref rotateVel, lookSmooth);
        transform.rotation = Quaternion.Euler(xTilt, eulerYAngle, 0)
;
    }
}
```

3. Simple Dialogue Manager

```
public class End_Dialogue_Manager : MonoBehaviour {

    public Text dialogueText;

    public Animator DialogueAnimator;

    private Queue<string> sentences;

    void Start()
    {
        sentences = new Queue<string>();
    }

    //function that applies the sentences of the dialogue class to t
he queue
    public void StartDialogueEnd(Dialogue dialogue)
    {
        Debug.Log("1");

        sentences.Clear();

        foreach (string sentence in dialogue.sentences)
        {
            sentences.Enqueue(sentence);
        }

        DisplayNextSentence();
    }

    //Displays the next sentence in the queue
    public void DisplayNextSentence()
    {
        if (sentences.Count == 0)
        {
            EndDialogue();
            return;
        }

        string sentence = sentences.Dequeue();
        StopAllCoroutines();
        StartCoroutine(TypeSentence(sentence));
    }

    //Displays each separate letter one at a time with a small but n
oticebale delay between each
    IEnumerator TypeSentence(string sentence)
    {
        dialogueText.text = "";
```

```
            foreach (char letter in sentence.ToCharArray())
            {
                dialogueText.text += letter;
                yield return null;
            }
        }

        void EndDialogue()
        {
            DialogueAnimator.SetBool("IsOpen", false);
        }
    }
```

4. Complex Dialogue Manager

Runs through different functions after dialogue based on the name of the dialogue

```
public class DialogueManager : MonoBehaviour
{
    public Text dialogueText;

    public Animator DialogueAnimator;

    private Queue<string> sentences;
    private string nametxt;

    public Camera firstCam;

    public Camera secCam;

    public Camera thirdCam;

    public float sec = 12;

    public GameObject FreighterObj;
    public GameObject DebrisObj;

    public Animator DecisionPanel;

    public Animator btnAnimatorEnd;

    void Start()
    {
        secCam.gameObject.SetActive(false);
        thirdCam.gameObject.SetActive(false);
        sentences = new Queue<string>();
        nametxt = "";
    }

    //function that applies the sentences of the dialogue class to t
he queue
    public void StartDialogue(Dialogue dialogue)
    {
        DialogueAnimator.SetBool("IsOpen", true);

        Debug.Log("Starting conversation with " + dialogue.name);

        sentences.Clear();

        nametxt = dialogue.name;
```

```
        foreach (string sentence in dialogue.sentences)
        {
            sentences.Enqueue(sentence);
        }

        DisplayNextSentence();
    }

    //Displays the next sentence in the queue
    public void DisplayNextSentence()
    {
        if (sentences.Count == 0)
        {
           if(nametxt == "Wendy")
           {
               StartCoroutine(EndFunction());
           }

           if(nametxt == "Ending")
           {
               Debug.Log("Step 1");
               DialogueAnimator.SetBool("IsOpen", false);
               Debug.Log("Step 2");
               DecisionPanel.SetBool("IsOpen", true);
               Debug.Log("Step 3");
           }

           EndDialogue();
            return;
        }

        string sentence = sentences.Dequeue();
       StopAllCoroutines();
       StartCoroutine(TypeSentence(sentence));
    }

    //Displays each separate letter one at a time with a small but n
oticebale delay between each
    IEnumerator TypeSentence (string sentence)
    {
        dialogueText.text = "";
        foreach (char letter in sentence.ToCharArray())
        {
            dialogueText.text += letter;
            yield return null;
        }
```

```
    }

     void EndDialogue()
     {
        DialogueAnimator.SetBool("IsOpen", false);
     }

    //Code for the ship crashing in the Town Level
    IEnumerator EndFunction()
    {
        Debug.Log("Working");
        firstCam.gameObject.SetActive(false);
        secCam.gameObject.SetActive(true);
        DebrisObj.SetActive(true);
        FreighterObj.SetActive(true);
        yield return new WaitForSeconds(sec);
        btnAnimatorEnd.SetBool("IsOn", true);
        secCam.gameObject.SetActive(false);
        thirdCam.gameObject.SetActive(true);
    }

}
```

5. Flashing Warning Light

```
public class Flashing_Light : MonoBehaviour
{
    Light flashLight;


     //flashes light in intervals of half a second
    void Start()
    {
        flashLight = GetComponent<Light>();
        StartCoroutine(Flashing());
    }

    IEnumerator Flashing ()
    {
        while (true)
        {
            yield return new WaitForSeconds(0.5f);
            flashLight.enabled = !flashLight.enabled;
        }
    }

}
```

6. Door Check

```
public class DoorCheck : MonoBehaviour {

    public Animator DoorRight;
    public Animator DoorLeft;

    public GameObject Guard;

    private Talk_to_Guard checkingOther;

    public bool granted;

    public Camera MainCam;
    public Camera HangarCam;


    //Using a copy of the script used at the guard by the door in th
e crashed ship level
    //It is possible to check the variables to see if the player has
 interacted
    //If the player has interacted, the door will open. If not, the
door remains closed
    void Start ()
    {
        checkingOther = Guard.GetComponent<Talk_to_Guard>();
    }

    void OnTriggerEnter(Collider aCollide)
    {
        granted = checkingOther.access;

        if (aCollide.transform.name == "Player_2" && granted == true
)
        {
            Debug.Log("Access Granted");
            DoorRight.SetBool("Open", true);
            DoorLeft.SetBool("Open", true);

        } else if (aCollide.transform.name == "Player_2" && granted
== false)
        {
            Debug.Log("Access Denied");
        }
    }

    void OnTriggerExit(Collider bCollide)
```

```
    {
        if (bCollide.transform.name == "Player_2" && granted == true
)
        {
            DoorRight.SetBool("Open", false);
            DoorLeft.SetBool("Open", false);
            MainCam.gameObject.SetActive(false);
            HangarCam.gameObject.SetActive(true);

        }
    }
}
```

7. Checking between scenes (using static)

```
public class CheckBetweenScenes : MonoBehaviour {
    static bool jumpstarter = false;
    public bool jumpstarterYes = false;


    //This script allows the value of jumpstarter to be read from an
y object that contains this same script as jumpstarter is static
    void Start () {
        jumpstarterYes = jumpstarter;

        if (jumpstarter == false) {
            jumpstarter = true;
            Debug.Log ("Doesn't have Jumpstarter");
        } else {
            Debug.Log ("Has Jumpstarter");
        }
    }

    void Update () {

    }
}
```

Bibliography

[1]. Quinn, Zoe. “A Beginner's Guide To Making Your First Video Game.” *Kotaku*, Kotaku.com, 28 Jan. 2013, kotaku.com/5979539/a-beginners-guide-to-making-your-first-video-game.

[2]. “How to Start Your Game Development.” *Unity*, unity3d.com/learn/tutorials/topics/developer-advice/how-start-your-game-development.

[3]. Brackeys. “How to Make a Dialogue System in Unity.” *YouTube*, YouTube, 23 July 2017, www.youtube.com/watch?v=_nRzoTzeyxU&list=PLb1_EkgxMv_VUYf4u3fkXcT61RtyWskQ5&index=11&t=646s.

[4]. Renaissance Coders. “Unity3D Character Controller in 20 Minutes.” *YouTube*, YouTube, 20 May 2015, www.youtube.com/watch?v=BBS2nIKzmbw&list=PLb1_EkgxMv_VUYf4u3fkXcT61RtyWskQ5&index=10&t=421s.