

Hugh Tyson's Portfolio

DES310 - Professional Project

LAGOON

GamePlay Programmer

Technology

Engine

Our first decision as a team was deciding what engine we would use to produce and develop the prototype. Our client provided us with the option of either Unity or Unreal Engine. As a group we decided to use Unity for a number of reasons. Firstly, it is an engine that both of the programmers have knowledge and previous experience with. Secondly, our fellow team members showed a keen interest in working with and expanding their skills within this engine. Finally, Unity hosts multiple online resources and guides which can be used to aid development.

Additionally, the team opted to use the High Definition Render Pipeline package as it allowed our artists to create and develop shaders within Unity in the form of PBR Graphs in a way they felt more skilled and comfortable with. There was documentation on this pipeline published online.

Source Control

A GitHub repository was decided as the method of Source Control as both programmers felt confident in their abilities with this application. GitHub Desktop was used by anyone that was accessing the repository.

Additional Resources

Some of the game's key mechanics involve the controller vibrating, however Unity does not automatically handle controller vibrations so an additional asset would have to be used. After considering multiple extensions, we decided on using the XInput extension due to it being a free open source extension which could be easily implemented.

Mechanics

The following section covers the Research and Development of Mechanics for the prototype. This section offers a General overview of the mechanics that I personally worked on. Technical aspects on these can be found in the Technical Design Document that has been submitted along with this document.

Third Person Movement and Camera

Movement

As Movement is one of the core mechanics of our prototype, it was imperative that it had the correct feeling. Due to the importance of this aspect, M. DePlacido and I used highly acclaimed games such as The Witcher 3 and Rime - both Third Person Player games – for reference.

One of the key decisions of the project regarded the implementation of the character's movement and further which method to utilise to achieve the desired result. Two quick prototypes were developed: one using the Rigidbody component and the other using the CharacterController.

Rigidbody

The Rigidbody method worked through applying a force to the character based on the player's input. This method also allows for realistic collision to occur while also simulating physics.

Character Controller

The Character Controller method worked by using the Character Controller's *Move* function. It allows for semi-realistic collision, resulting in the character to move up slight inclines. However, it does not react to Unity's gravity system, hence the Y-component would have to be altered.

Through the development of these two quick prototypes, it became apparent that the Character Controller component would be the most suitable method to carry further into development.

Camera

All information on Camera that I developed for the project can be found between page 84-91 in the TDD.

The Third Person Camera Component, which can be used while the player is in the Explore and Fishing state, was an aspect of our prototype I developed. This component's purpose is to follow the Player Character when traversing the Island. The camera's purpose then changes when fishing. The user will have limited movements when the line has been cast resulting in the bob always being in view.

This camera component further makes use of the Camera Collisions and Occlusion component which was developed. Four rays are cast between points surrounding the camera to the player. If any of these rays collide with an object the camera is moved forward so that the Player's view is not obstructed.

Player States

There are multiple states the player can change between. I was responsible for the PlayerExploreState and the transitioning from this state to others. In order to achieve this outcome, the prefab Trigger was created which held a Collision Box. When the player was within the Collision Box a UI prompt would be displayed allowing the transition between states.

I was also responsible for the development of the Player's Repair and Sleep state.

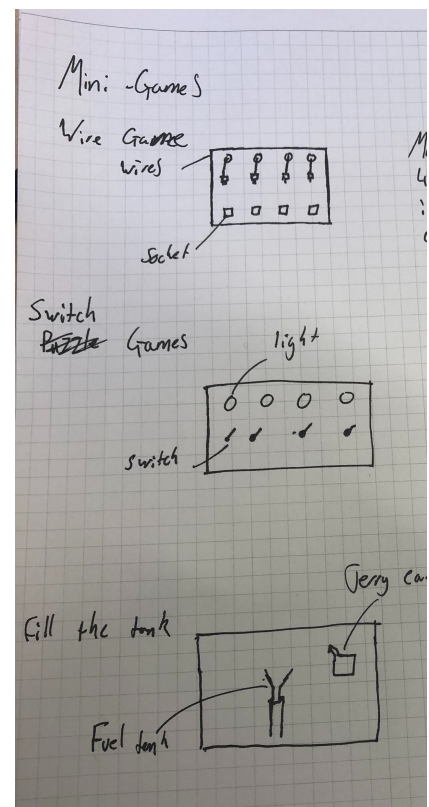
Repair

The Plane is divided into multiple segments which the player aims to repair. For the prototype only one section had to be repaired by the player. After discussing different mechanics, the group settled on the decision that small puzzles would be used to repair each Plane Segment. We drew our inspiration from games such as Rime and the Resident Evil Series for these mini-games. After playing these games, with Lead Artist M. DePlacido, three puzzle ideas were developed and put forward to be considered by the group.

Firstly, we considered a puzzle consisting of four wires. In which players aimed to plug these wires into their corresponding sockets. These wires would be coloured or have patterns that would indicate the matching pairs of wires and sockets. Through discussion with the team's other programmer T. Hudak, it was theorised that the wires could be moved and stretched using Verlet Integration.

Secondly, we proposed the idea of the Switch Game. Each switch would, in turn, activate certain lights while switching off others. The mini-game would be complete when all lights have been activated.

Finally, our third idea evolved around the Player refuelling the plane by pouring oil into the engine. This mini game involves the Player moving the Jerry Can with the left stick and rotating it with right. An additional element of challenge



could be added to this mini-game such as the pilot's hand would be unsteady, resulting in a reduced ability to aim the shaking can.

The group looked at all three options and decided that the Switch Game would be taken into development.

Plane Segments

A system was developed allowing users to view all of the Plane Segments and their state of repair. Each Plane Segment holds vital information about itself - such as the mini-games that it contains and the repair status of these segments.

The following are three messages that may be displayed to the Player when they view a Plane Segment.

“Needs Parts” - This plane segment is not fixed and the player does not have the required parts to fix the segment.



“Needs Fixed” - This plane segment is not fixed however the player does have the tools required to fix it in their inventory.



“Fixed” - The Segment is no longer requiring repair. As only one segment is required to be repaired in the prototype, all other segments have already been “fixed”.



Information on Plane Segments can be found on pages 97 and 98 of the TDD

Switch Game

This mini-game is made up of three scripts - the Games Logic; Switch Logic and Light Logic. As the game is initialised all switches are given information about what lights they will turn on and off and their initial position and rotation is set on the Plane Segment. Once the game has been complete the Player is restricted from viewing the mini-game again. Technical information on the switch game can be found on pages 98 and 99 of the TDD.



Plane Camera

The Plane camera is activated when the player enters the PlayerRepairState. This camera would need the ability to move around the plane and look at the segment which is currently

selected. For this aspect, I researched games which also involve planes and found the game Bomber Crew. I took the camera used when moving around the base in Bomber Crew as inspiration for the movement of the camera in the Repair Mechanic.

A system was developed which allowed the user to transition between point to view the different segments. The below GIF displays the transition between the Propeller Segment to the Tail Segment.



Conversation

Supply Drop

Information on the SupplyDrop and SupplyCrate can be found between pages 104 and 107 of the TDD.

The Supply Drop Event is called once in the prototype. When the Event is triggered, the Comrade Plane's Animation begins which allows the Plane of a secondary character to fly across the scene to deliver Supply Crates to the Player. This animation was produced using Unity's Animation System. There are three Collision Boxes in the Sky and when the Plane comes into contact with each these a Supply Crate is spawned which will fall from its position into the water below.



When a Supply Crate is spawned, a random amount of Inventory Items are stored within it . This results in every Player receiving different items when playing the game. The only Item that has substantial impact upon the gameplay is the Switch Item which is used when Repairing the Plane.

Conversation Camera

Another aspect I developed is the conservation camera that activates when the Player enters conversation state. As a group, we had numerous ideas as to where the camera should be positioned when the Player was having the conversation. We discussed the advantages of the camera looking at the radio or slowly moving around the island, ultimately coming to the conclusion that the Camera would look out over the Island and onto the horizon.

An effect was created for when the Supply Drop Event is invoked. As the Plane flies over the environment the camera will shake. The shake effect is faded in and out based on if the Comrades Plane is in view.

Journal Pages and Contents

Early in the development of our prototype, we discussed the concept of a Journal feature. This was proposed as a way that the Player would be able to keep track of their gameplay statistics and progress. Such as; the number and variety of Fish Caught, the repair status of the Plane while furthermore displaying the items which they had received from the Supply Drop.

Inventory

As the Supply Drop provides items that are used to repair sections of the Plane, we decided to have multiple additional objects dropped for the Character. Four items where created for the

purpose of repairing of the plane, meanwhile six additional items were developed that would be considered useful by a character stranded on a Desert Island.

These items are randomly spawned in the Supply Crates based on how common they are. Each item is given a value on how likely they are too spawn, the higher the value the more common the item is.

Further information on the Inventory can be found between pages 99 and 101 of the TDD.

Stats

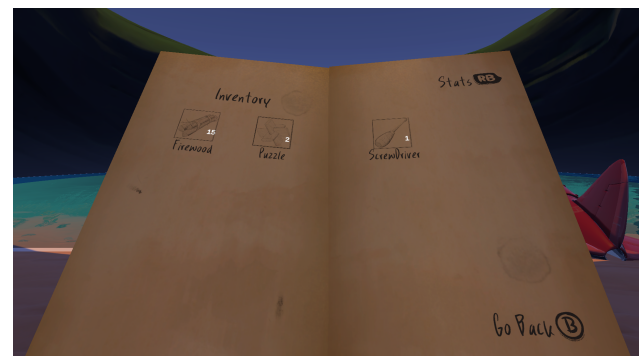
The Stats Manager holds information about the Player and their achievements. With multiple public functions, Stats such as the fish the player has caught, the length of time the player has spent on the island and the repair status of the Plane can be updated.

Further information on the Stats manager can be found between pages 101 and 103 of the TDD.

Inventory Page Pair



First Journal Iteration

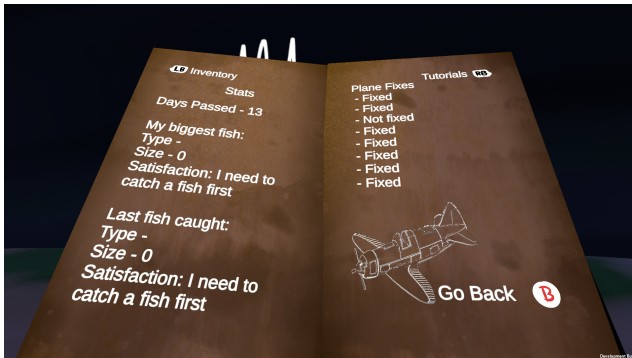


Final Journal Iteration

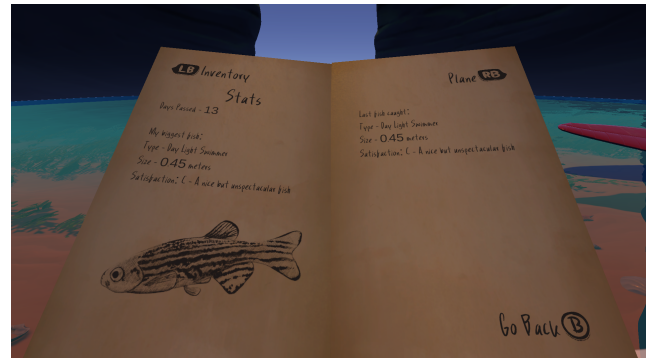
The first iteration of the Inventory Page simply displayed text without images. Through the use of the Special Text property, Inventory Items which had been added to the inventory or had been increased in quantity would be highlighted to the player. This was done through the change of the text's colour while simultaneously passing a wave through the characters. The final iteration introduced images allowing the player to have a visual representation of items in their Inventory. This resulted in the *InventoryItemDisplay* prefab being created. This prefab is

made up of 2 TextMeshPro boxes and an Image Element. All of which can be filled with data from the Inventory.

Stats Page Pair



Stats Page - First Iteration



Stats Page - Final Iteration

In the final iteration, the Stats pages was divided into two page pairs. Previously, the first iteration had both the Fish Stats and the Plane Repair Stats on the same page pair. This was changed for the final build as the Plane Repair Stats got their own page pair to be displayed upon.



Plane Page

Further information on all three Page Pairs can be found between pages 79 and 82 of the TDD.

DayNightCycle

Developing the Day Night Cycle was a fundamental aspect of my work on this project as it was a Barrier which had been included in the Conversation Graph. Moreover, as the

Character's story line is progressed over several days, implementing this feature provides more depth and reality to the gameplay. After testing, it was decided that a full cycle would last 10 minutes. Information about the Day Night Cycle can be found on pages 103 and 104 of the TDD.

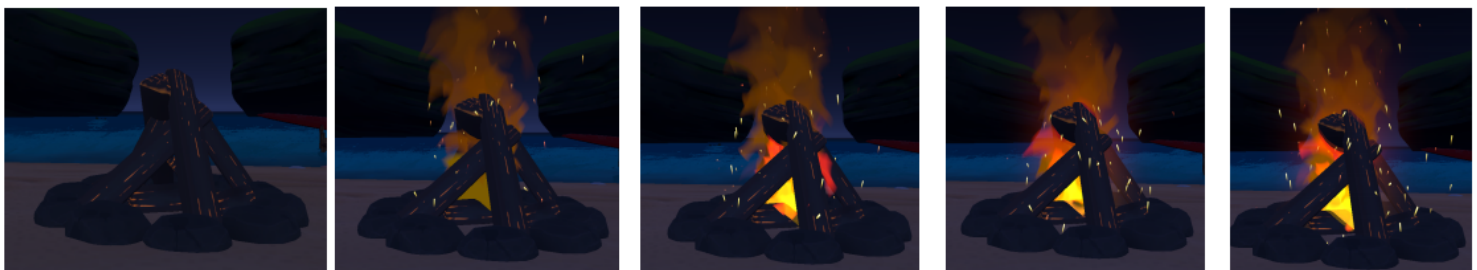
Sleep State

The Player can enter the Sleep State within the Hut. When this happens the speed of the Day Night Cycle is increased, while the camera's position is altered to look over the Fire onto the horizon.



Fire

As we reached the end of production, I approached M. DePlacido about the creation of a Fire Pit that would be activated when the game transitioned into the Night phase of the Day Night Cycle. He created a layered particle system for the Fire while I developed a system which fades the Particle System in and out along with the Emissive Material on the logs.



In order to take our concept further, I created a Fire VFX Graph that can be used to depict sparks emitted from the Fire. This Graph was also used in the simulation of Fireflies which appear on the island at night.



Stars

Initially, the team aimed to have stars appear in the Night sky which would rotate at a different speed and angle to the Moon. For this feature, a dome was placed around the environment with a texture applied that simulated Stars in the sky. The StarsManager script was created, which changed the alpha of the material fading them in and out. However, the Stars feature did not reach the final build due to a rendering problem.

Tutorial and WayPoints

One of the key pieces of feedback we received throughout development of our prototype was that the aims of the game were not made clear enough to the players. In order to improve on this feedback, the Fishing Tutorial was implemented along with the UI Sprite Animations for the Radio, Repairing of the plane and Sleeping.

Fishing Tutorial

The fishing tutorial is delivered to the player through a Display Box which appears on screen and provides instructions for the player. This method is very simple yet effective. There are four scenarios where a Tutorial Box will be moved onto the screen; the first time the Player starts fishing; when the Fishing Bob first hits the water; when a fish has been caught and finally, if no fish is interacting with the Bob.



Casting Tutorial



Catching Tutorial



Reeling Tutorial



Retry Tutorial

Waypoints

Three Sprite Animations were created by UI artist, S. Jarosz. These are activated based on the Barriers within the Conversation Graph or items within the Players inventory



The Animation to demonstrate that the a NPC is trying to contact them on the radio



The Animation to demonstrate that the Player can Sleep. This appears when the NEXT_DAY Conversation Barrier is active.



The Animation to demonstrate to the Player that they can Repair a segment of the Plane

Information on the Fishing Tutorial Waypoints can be found between pages 107 and 109 of the TDD.

Additional Work

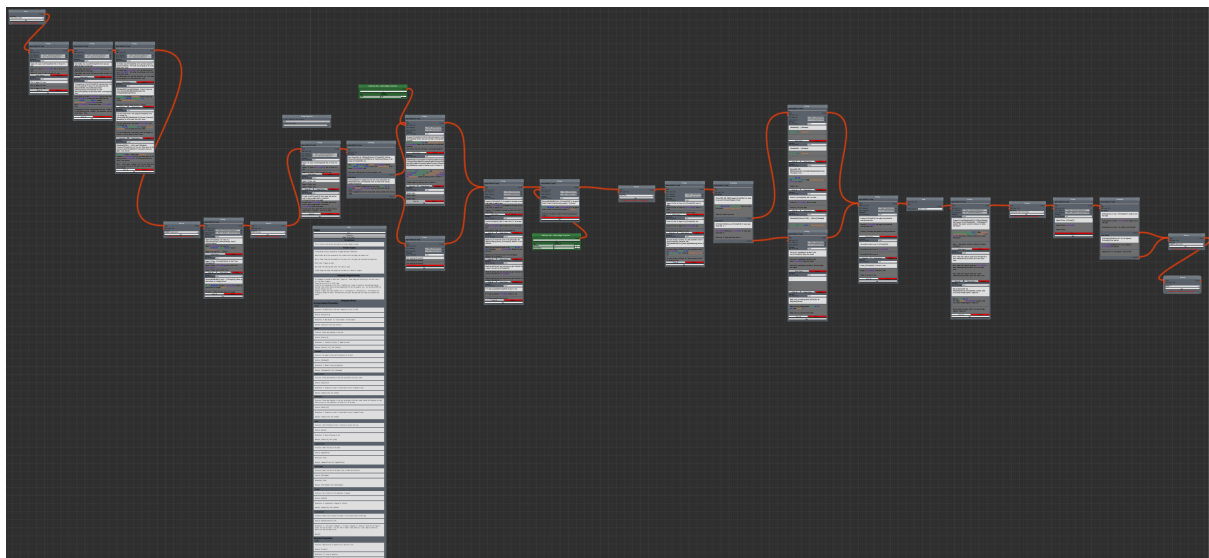
Along with developing Gameplay mechanics, I also took on further responsibilities throughout the development of the Prototype.

Audio

I sourced and implemented Audio for the mechanics which I had worked on. Through using websites such as FreeSound and FreeSFX. Furthermore, I used a Text to MorseCode Converter to create the sound of the Radio Transmission. The SFX attributions can be found in the Press Kit.

GameLoop

Once the core gameplay mechanics were complete, I was responsible for implementing the Gameplay loop. Two conversations had already been written by the games Producer W. Deighton, however the team was not completely satisfied that this was enough for the build. The text document Story, which has been submitted, was the initial idea I put forward to the group. This idea was then developed upon and implemented using the ConversationGraph - created by the tool's programmer T. Hudak. Below is the Conversation Graph that was implemented.



Through using the SpecialText properties, key mechanics were highlighted by a blue text with a wave being passed through them.



Reflection

I have genuinely enjoyed both the challenges and accomplishments that I have experienced throughout the development of this project. When this project began, I had a limited knowledge of working in the Unity Engine, over time however, I learned new methods such as Event Based Systems which will greatly aid me in future development. If there were more time for development I would like to re-develop mechanics, such as the Tutorial system to use new techniques which I have learned. My time spent in development has further provided me with a greater understanding of other disciplines and how we as a team can utilise each other's strengths and communicate effectively to produce high-quality work.

Upon reflection, I feel like this project has overall greatly improved my programming knowledge and ability to work within a multi-disciplinary team.

References

Technologies, U., n.d. *Unity - Manual: Unity User Manual (2019.3)*. [online] Docs.unity3d.com. Available at: <<https://docs.unity3d.com/Manual/index.html>> [Accessed 16 January 2020].

Docs.unity3d.com. 2020. Getting Started With High Definition Render Pipeline | High Definition RP | 6.5.3-Preview. [online] Available at: <<https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@6.5/manual/Getting-started-with-HDRP.html>> [Accessed 1 February 2020].

2017. Bomber Crew. Runner Duck.

2015. Witcher 3: Wild Hunt. CD Projekt.

2017. Rime. Tequila Works.