**Remember only around 2 minuets to look at portfolio**

Needed when discussing project

**What**

* What is the work?

Show it through download, source code, video, images etc

**Why**

* What project is trying to demonstrate
* What intended learnt by doing it
* Challenges and how they were overcome

**How**

* How it was actually done, i.e the source code

**General way to go about writing about the project showcasing**

* Add the labels to show general overview of project
* I sentence summing up what project is about

Projects

**Island Survival**

Collect fuel and evade zombies in this LAN multiplayer, random island generation tech demo.

Created as a second-year project, my goal with the terrain generation was to develop a highly customizable and reusable system that would speed up the development of future projects. This required thorough planning and development to ensure the creation of a modular system which is easy to use within the editor.

Along with this I was a main developer on the multiplayer system which required many optimization stages to ensure lag, largely due to unnecessary replications of the AI, could be reduced. Due to the terrains high network cost, it was decided that only the vital information, such as its seed, would be sent to the client’s machines so that they could all independently generate the same terrain.

Along with this the multiplayer system went through multiple optimization stages where issues, largly around the replication of the AI got fixed, by ensuring it only occured when nessesary. Due to the terrains huge network cost, it was decided that only the vital information, such as the seed, would be sent to all client machines, so that they would all independently generate the same terrain.

Ue4 C++ Multiplayer Procedural Terrain Student Work(Group of 2)

Video + What I worked on

Created as a second-year project, where any AI or PCG system could be implemented, this tech demo

                                has players collect fuel for their boat and evade zombies on some randomly generated islands.

                                The terrain system was implemented with the idea of its future use within

                                many other projects so that their development times can be significantly reduced. Due to this all elements

                                have been designed to be highly customisable and easily removable based on the project’s needs.

                                The added challenge of

                                island generation was implemented to allow for a more natural border to the game world.

as terrain is a large part of many games, so being able to quickly generate it at a click of a button can greatly speed up production, especially in the prototyping phase.

**Raindrop**

Created as a second-year project, this sees the player, the sun, fall in love with a raindrop, gain its trust and guide it up through various stages from the lake, all the way up to space. Being made with the two themes *The life of a raindrop* and *trust and betrayal* at its core the aim was to make an enjoyable game which followed good design principles. As the main programmer I endeavoured to ensure that the elements developed were highly modular and easy to use across all the various levels of the game.

**Pacman**

Made as a first-year project with the goal of making a recreation of the game Pac-Man with your own unique style. Due to the special constraints of the Pac-Man maze design, the generator contained some unique challenges, such as all inside walls needing to be at least 2 thick. As well as this the use of different assets based on whether the tile bordered the edge of the map or a path.

Within this sheep AI one main challenge, especially on larger maps, was creating an efficient

                                            and accurate navigation system that would be more precise than just choosing the direction at each intersection.

                                            <b>A\* pathfinding</b> was implemented to solve this, but with the condition that it would only calculate a path

                                            up to 40 tiles away, instead of the entire map, in order to get optimal performance with reliable acciracy.</p>

so an **A\* Pathfinding** system was implemented, replacing the initial setup of just choosing a direction at each intersection

was in ensuring accurate path movement which was more advanced than just

Unity v2019.3.3 C# Gameplay AI Student Work

Video

This first year project, developed largely between **August 2020 -** **October 2020(10 Weeks)** was undertaken to learn the fundamentals of game development. Tasked with recreating the entirety of the game Pac-Man with a distinctive style as well as creating a unique design innovation. Here I challenged myself by implementing an entire procedural maze system, which works around the constraints of the Pac-Man level design.

**Showcased at the UTS 2020 Spring Showcase**

Through implementing the well-known game Pacman, the project was undertaken to learn the fundamentals of game development within the Unity engine. Created as a first year project between **August 2020 -** **October 2020(10 Weeks)** the first **6** **weeks** involved creating the

The last **4 weeks** involved the implementation of the various systems such as enemy AI, player movement etc

Throughout this all sprites, animations, particles and code were created by me.

Some features to highlight are:

Sheep AI

Four types of AI exist:

* 1st Sheep(blue): move away from player
* 2nd Sheep(orange): move towards player
* 3rd Sheep(pink): move in random direction
* 4th Sheep(red): follow map border in clockwise direction

One notable challenge here was the 4th Sheep as the waypoint system used had to be easily implementable within the maze generation system. This was solved by making each green tile act as a waypoint with the sheep mealy finding and navigating between each in turn, in a clockwise direction.

Accurate navigation was also a substantial challenge, especially on a large map so an efficient **A\* pathfinding** system was implemented to give accurate results.

This was achieved by making the green border tiles themselves act as the waypoint system, with them individually storing the neighbours tile location, within a clockwise direction.

Procedural Maze Generation

* Generates maze border
* Spawns in centre piece and teleports
* Uses a xxx backtracking method to generate the maze

This was a particularly challenging system to implement as all inside walls needed to be at least 2-thick requiring a number of extra conditions to be determined and checked against.

* Spawn in correct map tiles with appropriate orientation

Other important features

* Character / AI movement via tweening
* Score / Lives System
* Saving / Loading via XXXXXXX
* Minimap system

Two

The maze generation system was implemented challenge myself beyond the core essentials of the assignment by gaining an understanding of how maze generation works. A core challenge was implementing the appropriate wall tile sprite in the right location and the fact that all inside walls needed to be a minimum of 2 tiles thick.

Play in browser here

See source code on github

**Island Survival**

**Raindrop**

**FEIT Prototype Exhibition**

**- Spring 2021**

**FEIT Prototype Exhibition**

**- Spring 2021**

**Deep Learning Page**

<p>In order to be able to fully understand how reinforcement learning works as simply and easily as possible, one of the simplest algorithms, Quality Learning was implemented.

                                        As designed to solve simple sequential problems it was largely tested with OpenAI Gym’s <b>Luna Lander environment</b>.

                                    </p>

To further my understanding of reinforcement learning, and to really challenge myself, a more complex algorithm, Deep Q-Networks, was implemented. As it’s designed to solve more complex sequential problems, ones that have a discrete action space and a continuous state space, it was largely tested with OpenAI Gym’s <b>Luna Lander environment</b>.

Unlike Q-Learning, the use of a multi-layered perceptron neural network, replay buffer and target network within the algorithm made its implementation a lot more difficult. As well the additional hyper-parameters, such as the <b>replay buffer size</b> and <b>target network update rate</b>required further analysis and fine tuning to ensure issues such as catastrophic forgetting were not occurring.

This was a key issue as on multiple occasions I would see issues that looked like catastrophic forgetting, tweaking the parameters and re-training only to eventually find the issue being some bug in my implementation. Such as the <b>backpropagation</b> being incorrectly setup, the <b>Mean Squared Error Loss</b> having the target and predicted values around the wrong way or even the <b>matrices</b> having the rows and columns flipped.

, but created more hyper-parameters that needed careful tuning to avoid

An exploration into reinforcement learning using some OpenAI environments