Created by Louie
And Daniel

PIPJUMP 2D

GAMEPLAY

- The game is about a lab rat who tunnels through tubes and hamster playgrounds in a psychological experiment regarding rat behavior. The rat is put a few tests to gauge its capabilities and test its cunning when placed into an environment difficult to traverse, it's also used to test the rat's nature for food and how willing it would be to make the extra effort to obtain it. Eventually after completing all the tests the rat will eventually find a way to make, it outside of the playgrounds it was being tested in and enter through a crack in the wall and escape outside.
- The aim of the game is to use the Rats given move set of platforming tools to grab keys throughout parts of a level and open doors to progress over to the next part of the level. The Rat will have three health points allowing them to get hit twice before having to restart at a checkpoint. If the player touches a checkpoint they could then restart from where they set it off and have their progress saved.
- The specific platforming mechanics go as follows. A double jump function that limits the player to only two jumps until they land on the ground. A wall jump where if the player character clings onto a wall and presses the jump button the player will bounce in the opposite direction they are facing on the wall and a dash mechanic that allows the player to cross a far that extra bit of distance.
- There are no enemy types in the game but there will be spikes to deplete the players health .
- The Scoring system isn't unique outside of the multiplier gimmick, but the games focus on have tricky platforming challenges along with bonus cheese that increase your score further gives the game a little bit of skill ceiling along with some depth to the normal gameplay loop.

UI ELEMENTS AND VISUAL DESIGN

- Ui in the game is very minimalist and almost nonexistent. The reason being that A the game doesn't have too many outlandish features that need extra Ui to communicate them and the health and scoring systems are very basic and don't need much outside of the bare minimum to showcase them.
- The design for the rat was conceptualized first and it was used so there would be visual reference for the player characters move set on screen. After a while, the rat design stuck, and the rest of the game was themed around that initial design.

PREPARATIONS AND CODING

- We made a few preparations before we started making our video game. We watched countless YouTube videos to study methods, classes etc.
- We tried to study some art on YouTube for the artwork for our game but we could use some more practice.
- In our project we used mostly public classes as we needed most of our scripts to take references from other scripts to make the game work.

DEVELOPMENT

- The game was developed in unity with the code being in C sharp and with assistance provided by Unity's built in components.
- The player movement was handled with a mixture of a physics modifier that reduced friction and drag, a rigid body 2D for general gravity, a list of code that applied forces at specific positions to influence the players position and the usage of unity's built in input system to link inputs with the force appliers and allow for movement.
- Colliders like the box, circle and capsule colliders were used so the player could collide with other objects in the game world. An extra collider was used underneath the player to detect the ground and allow it to jump only when it overlapped with the ground. (This also meant we tagged the ground to a ground layer and referenced it in the player script so that it only jump when the box specifically overlapped with the ground.)
- The jump was created in a similar fashion to the movement only it used Unity's old method of input by directly referencing the rigid body and applying that every time the space bar was pressed the rigid Body's velocity upwards would increase.
- The respawn and checkpoint system were created by having a script check if the players collider hits the checkpoints collider, when this happens, an animation is played to show the player, they have got a checkpoint and there spawn point is set. The score was created by creating a specific tag for a group of objects and having a counter go up each time the player collided with one. The colliders for the score objects would be turned off if the player collided with them to prevent the player from abusing the score system.
- The animations were initially created in photoshop and were linked together in Unity's built in animation system and to make sure they played when appropriate each animation was programmed to play after each input and action was press.

DIFFICULTIES AND CHALLENGES

- There was an initial struggle when attempting to get the left and right movement sorted the first draft of it was to push the vector position of the players rigid body by a specific amount every time the left and right arrows were pressed. However, it was very janky, didn't let the player move when the keys were held down and it didn't allow for any precision in the movement so afterwards it was tweaked and edited with .delta time and forces pushing it at horizontal positions. The last piece to complete the movement was to flip and cut the force and allow the player to move left and right without excessive momentum and allow for greater accuracy.
- We had a big issue with getting the inventory system to work and the wall jumping is very janky.

EVALUATION

- Overall, the development process was rather rough but interesting as it provided a lot of valuable lessons on not only how Unity as a game engine works, how C sharp works in comparison to other programming languages but also how game development works amongst a team or at the very least between two people.
- The initial growing pains for learning unity were incredibly rough especially when trying to figure out C sharp in comparison to Java but the components and built in systems allowed for a much easier time understanding how to construct a working line of code in the language. An aspect that also made the learning process easier was the visual feedback of the game as it helped give a better understanding of what to work on and how to improve certain aspects of the gameplay.
- We have enjoyed using Unity and C# we have learned how to use serialized fields and game managers that allows you to alter loads of different variables from one object.