Graphics Project (Trapped) Report

By: Matthew Green (7727974) and Zhikan "Jason" Xu (7778937)

Initial Idea

We started the project wanting to make a game that could incorportate all the graphics techniques that we learned over the course of the term. This lead us to coming up with making a platformer game as we could add different lightning for different levels and some reflections throughout the levels. We had to also decide on how to manage our version control and thankfully GitHub's desktop app made that very easy for us to work with.

The name for the game "Trapped" we thought of quickly as we felt it didn't have much to do with the actual project itself however the name comes from the idea that your player character is actually a person trapped in a ball and you have to get to the bottom of a cursed pyramid and open a chest which "may" free you from your entrapment.

When deciding where to make our game we had many options of different engines to choose from (i.e Three.js, Unreal Engine, Unity, etc...) however we decided on Unity as Matthew was familiar with it already and it has a low learning curve for Jason to learn for what we needed. We initially thought of using Three.js however with our need to use physics Unity was a perfect fit as it has built-in physics colliders for all our assets that we needed. For creating assets we used a few pre-built ones from the Unity Asset Store (\_DesertKits64\_, AxeyWork, and TornadoBanditsStudio) as well as using blender to create a few of our own. Using blender was a learning process as neither of us had used it before therefore we didn't go too much into different abilities that blender can allow us to do (i.e animation and complex texture mapping).

When planning out our levels we wanted to make sure we could get a broad array of graphical environments to demonstrate different techniques. We initally had the idea of doing a forest, desert, and snow covered area however ended up using an underground inside a pyramid instead of the snow covered array as it made the transition to levels look better. The levels were designed by both of us coming together and planning stuff out with quick drafts and then implementing there after. Using an underground level also allowed us to give the effect of dim lightning in a darker area via torches.

Later on throughout the project we came up with the idea of adding a menu screen to give the game a better sense of being a polished game as well as being able to add some UI graphical ideas. We also came up with some collectibles to give to let the player collect throughout their journey as the game has little to difficulty we wanted to allow them to have something else to work towards. Collectibles add little to actualy game progression and are not needed for actually finishing the game.

Work distrubition was decided on during our few meetings we had throughout the project and for the most part was done through a to-do list in the repository. Jason did most of the scripting for game objects (i.e player movement, enemies, platforms, etc..) where as Matthew did the graphical position of objects as well as creating assets in blender (i.e the trees, clouds, and coins).

We decided to add a few out there ideas for the game like a Day/Night cycle for changing lighting, a few enemies with some arm animations as well as them just looking like android robots for simplicity. Along the way through the game the player will find traps some of which move and some that don't the traps will send you back to your last checkpoint.

Breadth

Depth

We designed two types of enemy robots in our game. The first kind moves among the patrol points we assigned to it. The second detects the players in a certain range and moves towards them.

To make enemy robots look more complicated, we added arms and movement of arms using Unity built-in animation. It is challenging because the arm we designed is composed of two parts. To make arm movement look smooth, we had to mimic how human arm moves and apply animation to our robots.

Collectible objects vary on different levels. There are mushrooms, coins, and crystals. To make them look more eye-catching, rotation movement was applied. Further, the crystals in level 3 has mysterious glowing effect.

Optimization of these collectibles was also considered. After we knew the fact that static collider requires CPU to calculate collision each frame, we searched online and found that adding a RigidBody transforms a static collider to dynamic which doesn’t demand CPU to do these calculations.

Traps and challenges add more fun to the game. There are static spikes, dynamic spikes making a protrude motion and wall crushing the player. Moving and falling platforms were carefully designed to make sure players enjoy the challenges instead of being overwhelmed. Essentially, we tried to put platforms in different positions and tested it out ourselves. It gave us a good sense of how difficult it could be for a player to make through the game.

To complicate the global lighting in level 1 making it look realistic, we added a day-night cycle system. The idea is to simulate Sun and Moon orbit around the Earth. Thus, two directional lights were added to the scene. One represents Sun generating yellow bright light while the other is given blue light standing for Moon. These lights were then programmed to rotate around the stage and take turn showing up in the sky.

used built in unity reflection probe in order to add reflective surfaces to different meshes

made the feel of dim light in underground areas

used convex mesh on hills in order to stop from being able to jump from beneath as it adds collision boxes underneath