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 incorporate 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\_, AxeyWorks, 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 initially 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

In this section we will go through the project and give you a basic idea of what we did for each component before going in depth on some of the more complicated aspects of the project.

We added reflection to portals as it was initially our idea to show the we can used reflection in Unity however we ended up also wanting to add water which was another way to how that we could added reflection while having the material of the object stay the same.

We made our player character a ball to show because the player has to move a lot we wanted to show that we can make an object translate and rotate while moving. The face on the front of the ball is simply there to show that the ball is actually rolling and not just moving without any rotation.

We wanted to make sure our levels we big enough to fit in everything we wanted to demonstrate but not too big that the player is left rolling around with no direction. All levels are about the same size although we did it all by eye and never actually calculated to see if the they were exactly the same.

To make the tree asset Matthew opened up blender and decided on an idea for a tree (ever green) and from there went on to make the tree by taking multiple cones and one cylinder and simply adding the together. Other assets that were made were the clouds which are a bunch of spheres added together and sculpted slightly to give them a fluffy texture to them. Assets that were not created by us are: the hills, mushrooms (although we made one just didn’t use it), grass, everything in level 2, and everything except the torches in level 3.

The 3rd level was made to feel like an underground area with dim lights. We used point lights attached to the torches and crystals to position the lights and change their intensity depending on whether we wanted them brighter or not. The colour of the lights were changed to give them the feel that the object was actually creating the light and not that light was just added to the area.

We decided to add tags to all our objects and in the player script we gave the player conditions depending on what object with certain tags they were touching they are able to do different things. For example, if an object is tagged as ground you are able to jump off of it.

In level 3 there was initially some clipping when adding the point lights on the platforms as the base was too long so it would flicker the spots that were not aligned properly. This was an easy fix but one that makes the game look much better as the flickering takes attention from the player to that specific part.

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.

Most of our BoxColliders had to be made into convex meshes to be able to no go through any side of them as the initial collider they are given does not protect the back or bottom of the object.

Degree of Success

We wanted to make the robots more realistic by adding arms and legs to them. The robots right now only have arms and a body and move in one direction however the animation for the arms needs some work as they both more the same direction. As well as the robots do not rotate to face a different direction. Overall the robots were a more complicated implementation than we initially thought.

For most of our colliders we used the built-in unity Box Collider however we could have made the hitboxes more realistic by editing the collider to more appropriately match whatever object it was trying to look like. Overall the colliders are not that big of a deal and they work just fine for what we intended them to do.

Our day night cycle is something we are proud of as it was not the easiest thing to implement but once we got it going it turned out really well. However the night cycle makes it fairly difficult for the player to see their shadow if they are falling.

The reflection on the water and certain portals initially we thought would be difficult to implement as we thought we would need special textures and materials for them. In reality after reading some the documentation for Unity we found you can easily implement reflection using the built-in reflection probe lightning object and adding it as a child of whatever you want to be reflective. Some of the complications with using this probe came from not understanding whether we wanting a “baked” reflection or a “real-time” one we went with a real-time reflection as it was the one that looked the best, we also had to change the material properties of the reflective surface to make the smooth and metallic. Overall reflection was not too simple to implement but the end product turned out to be exactly what we wanted it to be.

We decided to make a menu screen for the initial launch which have a camera that rotates around the first level and a 3D title of our game as well as the option to start the game or quit it. The camera rotating around the level is a little faster than we wanted it to be. Overall this was a pretty easy implementation as we found a tutorial that showed us how to do it.

In the end we feel the project was an overall success and taught us how to do all the necessary graphics techniques learned in class to create a decent looking game as well as potentially other more complicated graphics projects.

Citations

Professor Saad. (Mar 29, 2017). Unity 5 - Game Development - Moving Camera 360 degrees. <https://www.youtube.com/watch?v=hMAaGhXJR6w>

Unity Asset Store. AxeyWorks. Low Poly V2

Unity Asset Store. \_DesertKits64\_Sample.

Unity Asset Store. TornadoBanditsStudio. Low Poly Free Pack.

GameGrind. (Nov 4, 2016). Dialogue System and Interactions | Making a Simple RPG - Unity 5 Tutorial (Part 3).

<https://www.youtube.com/watch?v=mXjRR1nnC5M>