Hi Robert,

We have this document here to sort of go through who did what, what challenges we faced etc.

So to start off, our team is comprised of:

Dimitry Herbut - B00146796

Lee Browne - B00153427

Richard Jako - B00162883

We formed our team a few weeks before the team deadline and decided on the name Yumili (You, me, Lee) for our team. When we read the brief for the project we went through maybe 3-4 different ideas for what to make before settling on the current idea. So what is our game?

Well essentially it is a gorilla named Captain Banzo. He is up in orbit around the Earth in a space station. An unpredicted meteor shower hits the station causing severe damage to critical systems and scattering debris everywhere. The job of the player is to successfully collect debris and repair the station.

The basic format of the game is the player spawns on the station and must exit the station to collect the scattered debris and repair the station. We broke the game down into its basics to help us assign tasks.

**Roles**

Lee: Space Station Model, Intractability, Oxygen Tank

Dimitry: Debris Model, Properties, Collisions

Richard: Movement System Including Fluid Movement, Mouse & Keyboard Controls

During the beginning stages of making the game our biggest hurdle was of all things GitHub and getting GitHub to work. We thought we got it working at one point and then we tried merging our progress and it ended up completely messing everything up so last minute we just started using google drive.

Individually speaking, Richard found the process of getting the fluid movement working very difficult which resulted in a complete rewrite of the code that controls player movement. Initially the idea was to implement an omni-directional movement system ie. The player moves on all 3 axis based on where the mouse is pointing needing only WASD movement however this proved to cause a multitude of issues so we concluded that the game should use the standard WASD along with Space bar to vertically move up and LShift to vertically move down.

Dimitry found the code in the debris spawn script wouldn’t mesh well with each other, so he improvised from what he learned through earlier tests to find the correct type of code and elements that did not conflict with other scripts or the debris prefab. And afterwards dealt with an issue concerning the attachment and allocation of the correct scripts and code used to make the debris collectable.

Lee found making the Oxygen meter (rechargeable health bar on a timer) very difficult to implement but got it working beautifully in the end by completely redoing the script. The initial script used damage taken by input but was difficult to incorporate due to the necessity of other variables. The rebuilt script used a much simpler Time.deltaTime based script.

As for the references in which we sourced advice and external code/assets, they are shown below.

Create with Code – Lessons 1 – 4 (specifically the use of Quaternion.Identity, Instantiate, Player Health Concepts, UI Basics, particle systems, collisions, loops and arrays, Raycasts and more.)

<https://youtube.com/playlist?list=PLGUw8UNswJEOv8c5ZcoHarbON6mIEUFBC&si=dHnTcH_r9ZaF9GL7>

The above youtube playlist helped us in our shortcomings to improve our concept understanding and application throughout the project.

<https://assetstore.unity.com/packages/2d/textures-materials/sky/allsky-free-10-sky-skybox-set-146014>

This was an imported asset used for the skybox from Unity Asset Store

Input Systems – These was an external tool by uniter to manage input actions

TextMeshPro – This was used as an better alternative to the standard “text” object

The things we have learnt throughout our work that were not included in our references but from our own learning was:

The anchoring of UI Elements,

Interaction Layers and Layer Masks,

Specific Customizations for Inventory and Collection Mechanics.

We used these tools and specifics to tailor the game to our custom needs.