*A Brief Introduction:*

This project was the result of hours of work by the five of us. The work was broken down to reflect what our core competencies are, as well as what we each wanted to explore. Two members of the team largely worked on the frontend design, particularly the stage and menu, while the remaining worked on backend elements, particularly the movement pattern, computer player, shooting, and character health.

This project began during BostonHacks 2019, with Dev and Ryuichi struggling to come up with an idea for this project. We experimented with a few different alternatives before coming across Unity. We spent the rest of the hackathon figuring out the very basics of manipulating Unity, by placing a basic rectangular stage, a shooting pattern, and a movement pattern.

We then iterated on the design by building complex stages, more robust movement patterns, a good computer player, and camera. We hope that you enjoy our game!

*Technical Brief:*

Though Unity already provides characters and objects that do several of the things that were required for this project, **it seemed more prudent that we create the program from scratch in order to better learn C# and the Unity Environment**, and to test our programming skills in an unfamiliar setting.

To build our project we used Unity to create the necessary objects for the game. **The objects we created include the circle for the player, minor enemies, final boss, components of the stage, and bullets.** Each object then has specific functions like how each is supposed to move and act. The player has specific functions like shooting, jumping, and moving horizontally. Each enemy is also able to move horizontally and shoot. Both the player and enemies have health bars that decrease when hit by a bullet. This was created by using a slider and a function that checks whether a bullet is in the same spot as the player or enemy. If the bullet overlaps with the player, the slider will decrease. Since there's components of the stage that are able to move, they must also have their own functions for this. The background is just a constant image with a camera that is locked onto it. Also, the final boss has similar characteristics to smaller enemies, but it is able to take more hits and has more attack capabilities.

*How the Player works:*

Using the input.getkeystroke() function in the Update() function, we were able to successfully control the player’s movement. We use a similar pattern to control the enemy’s movement. The player’s movement includes horizontal movements and jumping. The user can move using the arrow keys and shoot using the spacebar.

*How the Stage works:*

Some stages components can move. In the powerpoint, you can see that one of the components spins based on images of the gameplay.

**About the class hierarchy: There are 2 base classes for stage elements: BasicBlock & TrigerElement**

For stage -- BasicBlock class derives all the elements that follows a programmed route (rotate, move up and down) or perform certain reaction when it is hit by player or bullets (reflect bullets). TriggerElement derives elements that contain a public trigger function and can be called by other objects (Buttons, enemies).

*How the Enemy works:*

**For enemy -- base class is called EnemyObject, which is an abstract class. There are three types of enemies, which are Enemy1(Miguel), Enemy2(Burak), and Enemy3(Densmore).** Each enemy has an instructor's face. There are several numbers of Enemy1 and Enemy2, but there is only one Enemy3, which will be our final boss. Enemy is able to move around, but it can only move forth and back. It will move forward for a certain distance from its starting point, then it will start moving backward for a certain distance. It is able to shoot bullets, and each enemy has its own shooting pattern. Enemies except for Enemy3 take three hits to get destroy. Enemy3 will take 10 hits to destroy.

*How the Camera Works:*

The main camera is fixed on player. This means that if the player moves, the main camera also moves. The background always displays the same background picture. There is also a health bar that follows the player and indicates damage taken. The final boss also has a health bar to show how much damage the player has dealt.

*How the Bullets Work:*

There are four types of bullets. The first type is a bullet that travels in +x direction, the second type is a bullet that travels in - x direction, third type is a bullet that travels in both +x and +y directions, and the last type is a bullet that travels in -x direction and the +x direction. Bullets will destroy themselves once it hits a player or enemy. This destroy function is called in the object that it hits.

*Difficulties:*

One of the main hurdles in this was the somewhat steep learning curve presented by Unity. That being said, the learning curve was lessened due to the similarities between C# and C++. After understanding this, we needed to rely on dozens of youtube videos to start to apply the Unity fundamentals to the project.

We found exceptional difficulty in creating a more robust jump mechanic. We wanted to implement a system in which we could tell if a rigid body was touching our stage but had many difficulties in implementing it. To circumvent this, we simply placed a time delay in our jump function.

*Summary:*

Creating the game was a good challenge for the group because everyone had to learn the differences between C++ and C# and how to use Unity. These are skills that we might be able to use in the future, and we will also have a finished project that we can show to employers during interviews to prove we are able to program. This project helped us learn how to properly make a project as a group, which is harder than we originally thought. Sharing the code that we each individually made through GitHub rather than through email was something that we weren’t used to before this project. We were able to create a pretty fun game with a good backstory, and we all had a good time doing it.