Technical Design Document for

‘Highbound’

Produced by Team Q

Producer - Kaine Smith
Lead Programmer - Dean Ford
Lead Designer - Finley Goddard
3D Modeller/UI Design - Odin Branigan
3D Character Modeller/Texturing - Leon Wilson
3D Modeller - Josh Morris
Level Designer - Rubin Bayfield

Target platform and considerations

The target platform this game will be played on is gaming PCs. With this in mind we will need to consider what the average specification of a pc is and if it will be able to run our game at a comfortable frame rate. We will also need to consider the different resolutions of monitors that people play on as this can affect the placement of the UI in the game. If we don't consider this, then we could get half of the UI cut off or have the UI stretched when a monitor is at a lower or higher resolution than what we test on. We also considered mobile as a platform for our game however adding a game to the app store conflicts with our budget, therefore we decided that pc would be the best fit for our project.

Technology choices

To make this game we will be using the Unity engine as it is very versatile with all the different components and assets you are given to help. Some of the features we will be using in our game include:

- Premade components and scripts that are fully customizable.
- Easy to use animators
- Hierarchy list to organise parent and child objects in a scene/prefab
- Many useful libraries with helpful functions for coding
- Scene view window to visualise the game objects and scene you are using
- Asset store with premade packages

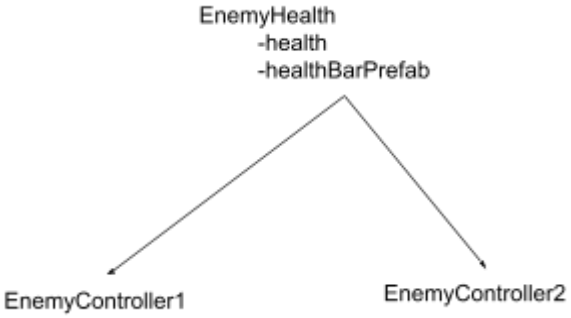
We decided to use unity over game engines like unreal since our lead programmer has more experience coding with unity, however using unity also had its cons. One of which is that it is harder to make the game look how you want it to.

Game architecture

-Enemy AI

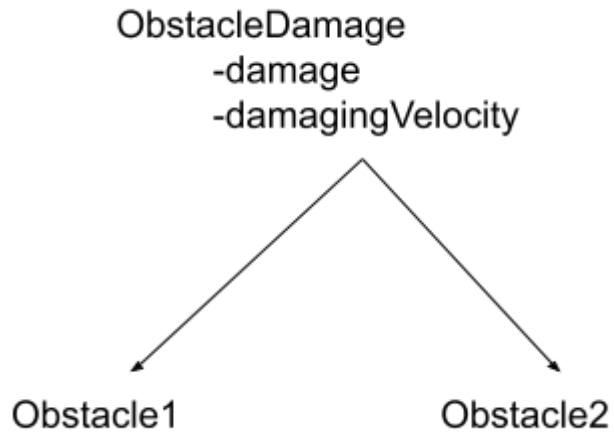
Most enemies in our game will have a very simple AI that simply tracks the player position and when the player gets close enough, they will follow the player to attack them. All the enemies will use an Enemy Health class to give them a customisable health stat. when they take damage this script will create a health bar UI above their head to display how much health they have. Other than that, each enemy will have a

separate controller script that handles the movement and attacking of that enemy as each one reacts differently.



-Obstacle AI

The obstacles AI will consist of collision boxes and a small script to allow them to move and damage the player. The spinning ball and pole will both share a damage script but the ball will have a separate movement script as it will use a physics based movement system. The pole will spin using a movement script that updates the quaternion rotation every frame whilst the ball will react to gravity and have force applied to it every frame to get it to spin.



-Player Scripting and components

The player will have multiple scripts to control different actions. The player will have a stats script that will handle the player's health and health bar. The player will have a controller that handles the movement of the player including the jumping, walking and attacking of the player. This controller script will interact with the player's animator to change the animations

when needed. The player will also have multiple collision boxes to handle the base collision and to check it the player is touching the ground.

Hardware Requirements

The game we are creating won't be that demanding. Most of the models we made have a low count of polygons so rendering them wont take up much memory. We are also going to add some post processing to make the scene look better. This will take up a bit more memory but not much. This means that most PCs will be able to run this game easily at a smooth and comfortable frame rate.