User Documentation

---------------------------

In this document I will describe the helmet convincer project, state the goal of the project, and give instructions on how to set up the helmet convincer program so that a developer new to the project can quickly understand the basics of how the project works.

Project Description

-------------------------

This project is a mobile VR application that simulates an open world environment in which the user rides a bicycle. You are both in a VR space experiencing the world you are in while riding an actual bike controlling your in-game character.

Project Goal

----------------

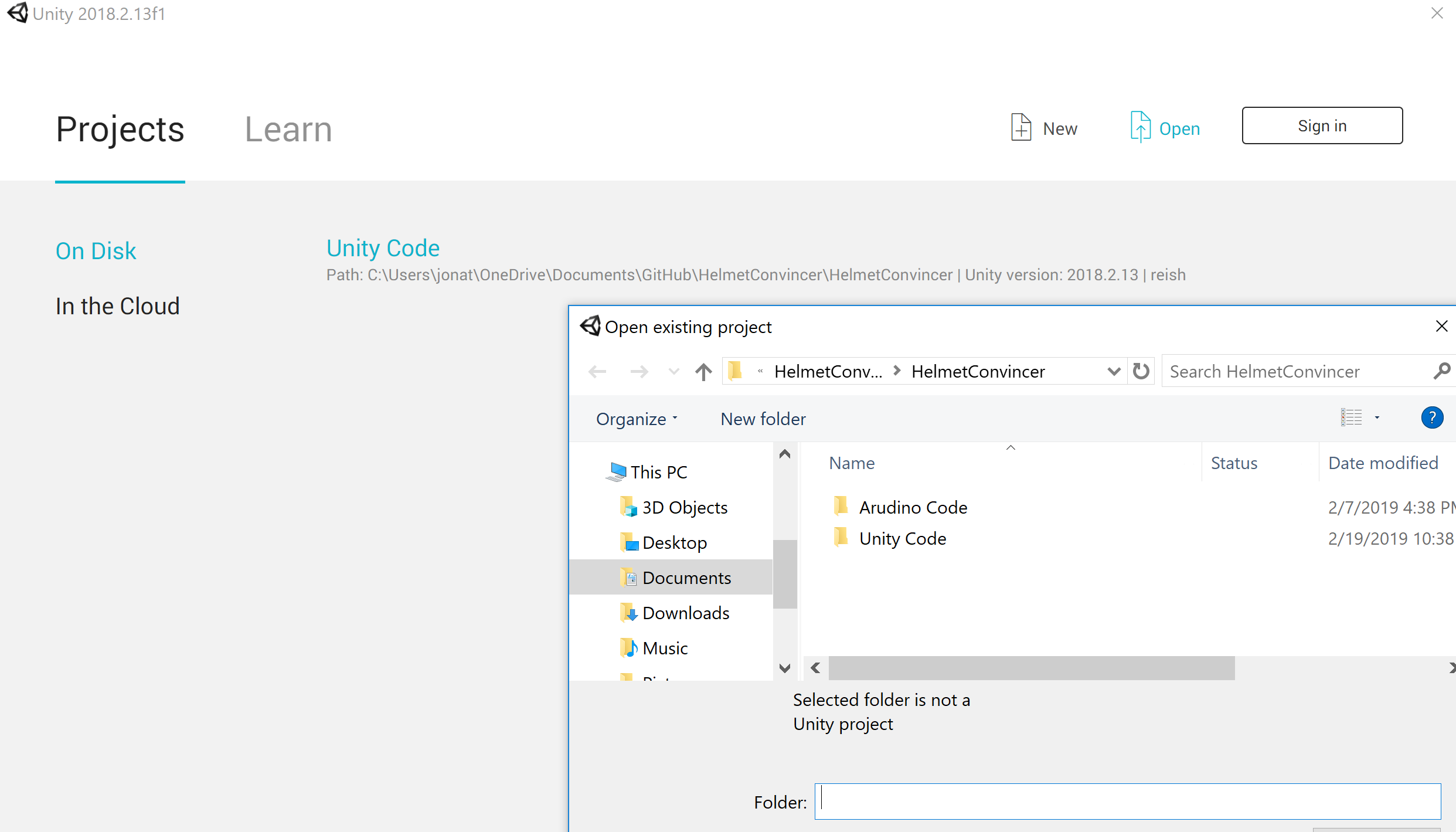
The goal of this project is to get the user to realize how important it is to wear a helmet while riding a bike.

Getting Started

------------------

In order to get a modifiable version of the helmet convincer game:

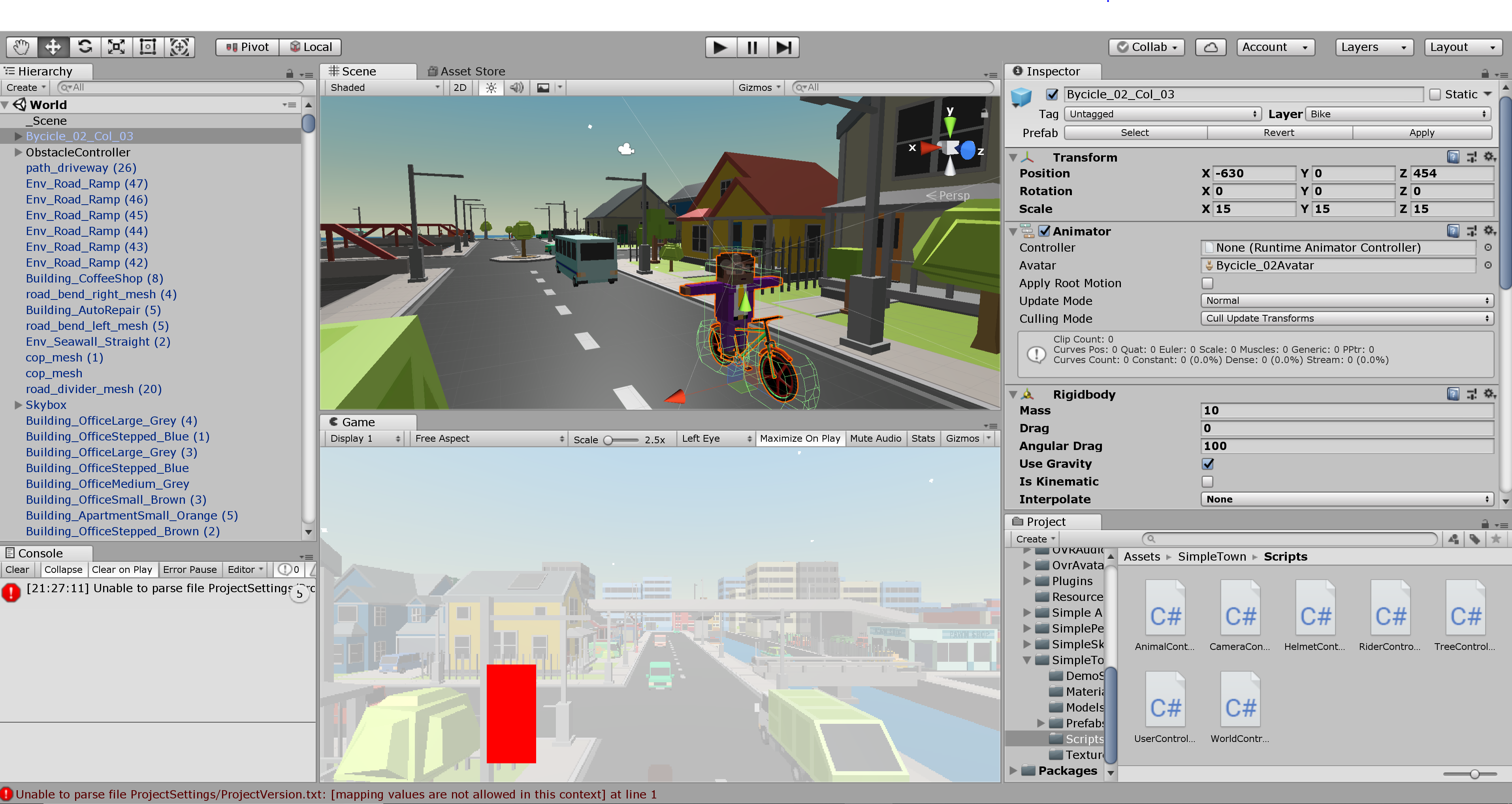
1. Download Unity 2018.2.20f1 (future versions of Unity may or may not cause bugs within the code) from the link provided (Unity 2018.2.20f1: <https://unity3d.com/getunity/download/archive>
2. Clone the Helmet Convincer repository (<https://github.com/KSU-CS-Software-Engineering/HelmetConvincer>) to your machine
3. Unzip the repository
4. Open Unity and click the “Open” button at the top of the window
   1. In the file explorer that opens, navigate to the repository with the Helmet Convincer code navigate to and click the “Unity Code” folder inside, and click “Select Folder” to open the project within Unity



1. Once the folder is selected, Unity will load all in-game resources necessary for functionality and then open a Unity window with the game inside

The Unity Layout

-----------------------



When Unity opens, a screen similar to the one above will be shown. In what follows, I will give a brief description of each window in the screen. Note that the screen above may be slightly different than the default layout; I have merely arranged the windows in a way that feels appropriate for myself.

1. Hierarchy Window: The hierarchy window gives a list of every single structure that can be found within the game. The structures are not listed in any particular order, but once you become familiar with some of them, you can search particular structures with the search bar located above the list.
2. Console Window: The console window is used mostly for sending debugging messages. I you choose to download a more up-to-date version of Unity, the window may automatically populate with warning messages related to the most recent updates.
3. Scene Window: The scene window gives a particular view of the world and allows a developer to select various aspects of the environment so that they may be viewed and modified. Scrolling over the scene window allows for zooming in and out on the world, which can be useful for getting a broader or more specific view of the game.
4. Game Window: The game window gives a first person view of the world. In the image above, the user is viewing the game from the death screen camera.
5. Inspector Window: Clicking any structure within the scene window will open the inspector window, in which the user may view and modify properties of a structure. Since I have clicked the bike, you can see the relative location of the bike on the map as well as information regarding the object’s in-game physics.
6. Project Window: The project window gives a view of all of the files in the project. Many files contain information about game textures, but unless you wish to modify the appearance of the game itself, the most pertinent files are located in the assets > SimpleTown > Scripts. These scripts contain C# code that control the way the game works.

The Code

------------

In the following section I will give information about each class, highlighting parts of the code that are the most essential or may be confusing.

1. User Controller: Determines the user’s interaction with the scene
   1. User Controller has a method called Start() which initializes some values and spawns the player on the map. Each class has a Start() method that operates similarly
   2. User Controller and several other classes have a method called FixedUpdate() which is a method built into Unity which updates information about the game at a fixed rate. In the case of User Controller, FixedUpdate() calculates some information about collisions and repawns the player if the space bar is pressed.
2. World Controller: A broad class controlling much of the behavior of objects within the game world
   1. Start() initializes port values
   2. The Repawn method contains four default locations where the player may potentially respawn. These locations are given relative to the map and may be modified if needed by modifying vector coordinates.
   3. FixedUpdate() adds information that causes the player to die automatically if time runs out
   4. Update() is a method that is called whenever the game screen updates. The method contains information about Arduino input, but the Arduno is not currently functioning. Additional code gives instructions on how to modify the player’s view whenever the headset rotates (or alternatively, whenever the mouse on the screen moves).
   5. Several additional functions are called in order to simulate death whenever the bike collides with a RigidBody. The death currently results in a change of screen.
3. Tree Controller: Controls the behavior of trees within the game
   1. The game is currently set up to have trees within the world to start falling after a certain amount of time has passed. Update merely results in certain tree behavior after this time has passed and the “falling” Boolean variable has been set to true.
4. Rider Controller: Determines how the rider body interacts with the bike and the world around it
   1. Since the rider is connected to the bike, the player current just moves with the same motion as the bike object.
5. Helmet Controller: Determines how the helmet object interacts with the world
   1. FixedUpdate adds damage whenever the helmet object collides with some other object within the game. However, we are not currently implementing damage calculation because we decided that it would be more realistic for a single crash to result in a death screen.
   2. OnCollisionEnter() is also currently being phased out of the project. Damage was previously calculated by determining how far the bike rebounded when colliding with an object, but the amount of damage required to result in death is set high enough that the player will not die.
   3. DeathTrigger() is called whenever the player dies. The function switches the camera view and previously displayed a color that determined the amount of damage that a player had accumulated.
6. Camera Controller: Controls the position of the camera during runtime
   1. Contains code that indicates how the camera will rotate whenever the player turns his or her head.
7. Animal Controller: Controls the behavior of animals within the game.
   1. Animals are currently set to chase the rider whenever the rider object gets close enough to any animal. This functionality is controlled by setting a boolean variable equal to true whenever the player enters into the animal’s established radius.