For the design paradigm that we chose to use for our prototype, we decided on using an object oriented design. The reason we used objects was due to us deciding to make a game inside of Unity. The language of our choice is C#, and in itself C# and Unity has a lot of good interactions when it comes to using objects to make the game. For example, one of the classes that we have is a player object. Inside of this object, we have definitions for variables such as the default speed the player moves, the height a certain player can jump, as well as other variables. This allows us to define some of these variables as public if we desire to use them in any of the other classes, and this would extend to the different methods that we also wrote inside of the class. Another reason I would say that the way we are programming is object oriented is due to the way that all of the C# scripts are set up inside of unity. By default when you create a C# script, there will be a Start() and Update() functions that are available to edit. The Start() function is basically a function that will run once when the object is created, and the Update() function will update for every frame the game runs. I found coding in this way similar to C++ constructors, and it was easy to pick up the nuances of how the scripts worked because of the similarities. Overall, the design paradigm we chose to use was object oriented due to both the way we setup the classes, as well as some of the default characteristics of C# files in Unity.