



AR Remote Rendering Application

Critical Design Report



| **UCCS CE/ECE Senior Design Team**  Jack Brock  Amber Dolezal  Austin Hobbs  Amyleila Mejia  Stefano Signorelli | | **UCCS Project Sponsors**  William B. Micheal  Dr. Dana Wortman |
| --- | --- | --- |
|  |  |  |

Table of Contents

Detailed Description of Design . ……………………………………………………………. 3

User Manuals and Instructions ……………………………………………………………… 4

Introduction ……………………………………………………………………………..4

Maintenance Instructions ……………………………………………………………..4

Parts and Materials ……………………………………………………………………4

User Manual and Instructions ………………………………………………………..5

Turn On …………………………………………………………………………5

Brightness and Volume Adjustment …………………………………………8

Unity Game Engine Project Selection ……………………………………….8

Unity Game Engine Build Settings …………………………………………...9

Universal Windows Platform Configuration Settings Using Unity Game...11

Using the HoloLens Emulator for App Launch …………………………….12

Using the Application …………………………………………………………13

System Shutdown …………………………………………………………….14

Detailed Drawings of Parts, Schematics, Software Descriptions …...…………………..14

Bill of Materials …………..……………………………………………………………………18

References …………………………………………………………………………………….20

**Detailed Description of Design**

The VR/AR project has many different parts that makes the project’s purpose a reality. To start off, we have the unity build. The Unity build is a unity project which is built by the team which consists of the Hololens interfacing, C# scripts, assets, and game objects. Game objects will be created, like a Text Mesh Pro game object which will be used to help perform the user interface for the AR/VR Blackjack tool assist. The C# script helps deal with logic, and can help perform the probability algorithms needed as well. Other functionality needed which is built directly with the unity editor is a start button which will indicate the app to start the AR/VR Blackjack assist program. This button will know when it is clicked with hand gestures with the Microsoft Hololens. To have this button we will have to import the MRTK (Mixed Reality Toolkit) package. This toolkit has many objects, and tools which help build AR/VR games. After the start button is clicked, the program automatically starts detecting the player’s and dealers cards, and will perform calculations and update the UI with probabilities in percentages, depending on whether to hit or stay in the game of blackjack. In addition, the AR/VR blackjack assist will have sound effects when win/or losing a hand to make the game more interactive.

Moreover, the Unity Build is also able to install packages. These packages are 3rd party software that can be added to further add functionality to the Unity app. Examples of packages that will be used are “Barracuda”. Barracuda is used to help interface with the Machine learning algorithm YOLO. Barracuda is used to allow Unity to load a UNNX file so that it can launch YOLO itself, and the python machine learning training script. The Python script will be used to train the algorithm so that it detects playing cards, most importantly, the number specified. The card symbol doesn’t matter when playing Blackjack. Another 3rd party package that most likely will be used is “Vuforia”, Vuforia is used to allow access to AR functionality in unity, like the AR camera game object which replaces the camera in the unity game to be the AR camera for the Hololens. In addition it allows the creation and interfacing of AR images.

Finally, some other implementations that has been done with the AR/VR Project is its ability to perform a TCP connection to communicate data between Python and C#. The reason we might need this is in case we need to program with Python to then transmit data between Python and the Unity app. This data could include data needed for the probability calculation algorithm, or data needed for Barracuda and YOLO for object detection.

**User Manuals and Instructions**

# Introduction

Dr. Omid Semiari proposed a senior design project that was designed to increase an artificial reality device’s hardware performance by creating and implementing a remote rendering application. This project would prove that utilizing a host computer to conduct all the application’s processing would indeed improve the performance and battery life of an AR headset, in our case the HoloLens 1. In order to use our remote rendering application with a HoloLens 1, we have included some manuals and instructions below.

# Maintenance Instructions

For troubleshooting the HoloLens the Microsoft Ignite Documentation Online is a great source to reference. This documentation remains up to date on issues any owner of the HoloLens has faced, along with a detailed description on what to do in these situations. This information can be found at: [HoloLens (1st gen) hardware | Microsoft Learn](https://learn.microsoft.com/en-us/hololens/hololens1-hardware). This resource will also be extremely useful for users because it contains user instructions and provides links to online communities that can help with any problems that might come up.

# Parts and Materials

1. HoloLens Generation 1

### 

Figure 1: HoloLens 1 Components

1. Router
2. Cables used for Router
3. Host Computer
4. 2 Handheld Remotes

# User Manual & Instructions

# 1.)Turn ON

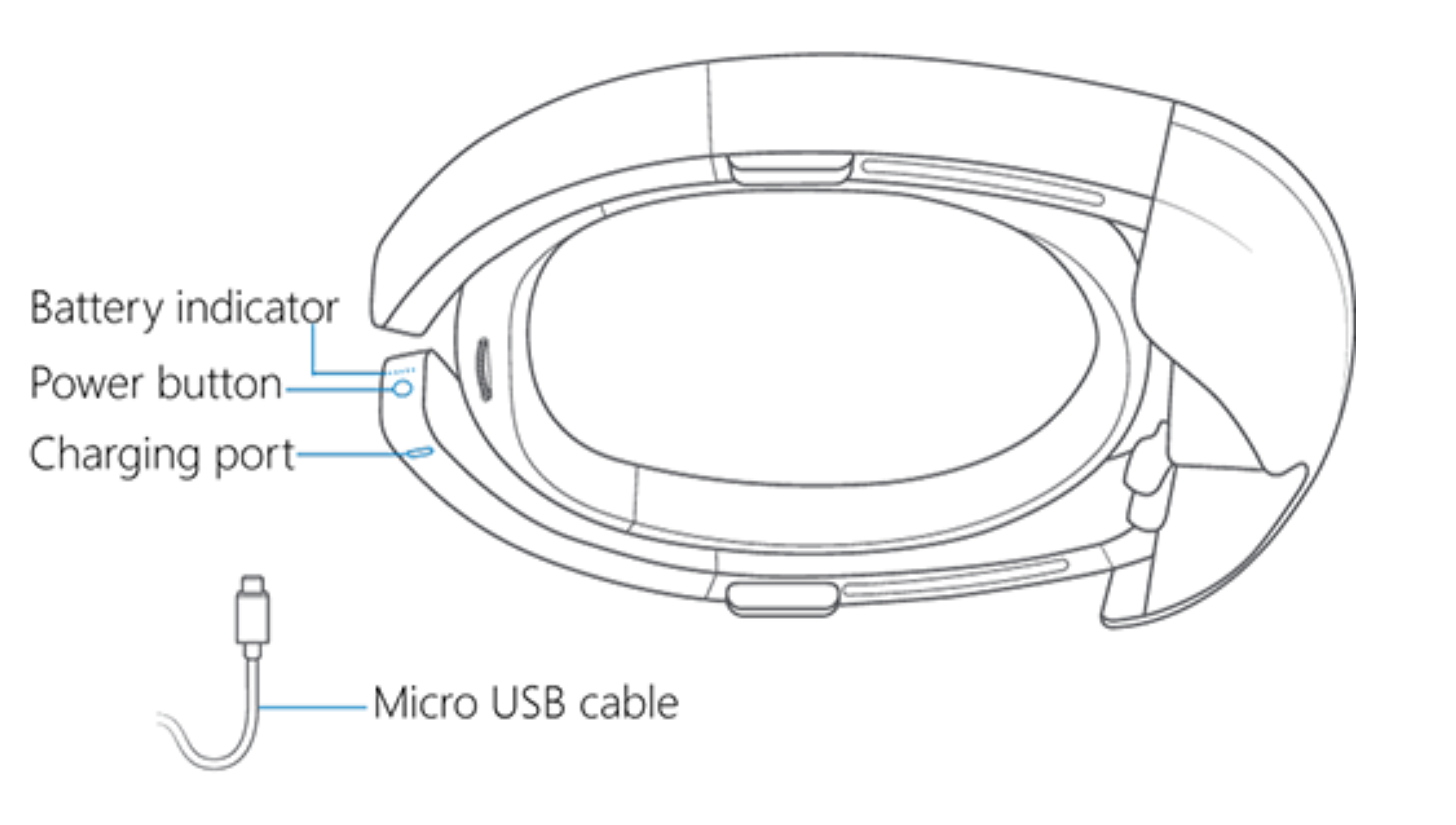
1. First place the HoloLens 1 on your head, ensuring that the device is properly sitting in place and not loose. An unsecured headset could end in injury or damage to the device. Please see figure 2 for more information on HoloLens adjustment methods.
2. To turn on the HoloLens Gen 1 there is a button located on the back of the headset (see figure 3). Hold this button down for four seconds.
3. Turning on the device also requires turning on the app for full interaction.
4. Open the project file in the host computer. The icon the user will need to find on the host computer desktop can be referenced in figure 3.
5. The user will enable and launch the app, figure 3, by interacting with the Unity Game Engine and double clicking on the app icon.



### Figure 2: HoloLens proper adjustment methods and handling.



Figure 3: Desktop version of the Unity Game Engine Launcher



### Figure 4: Power device overview. Charging and button locations can be clearly seen.

# 2.) Brightness and Volume Adjustment

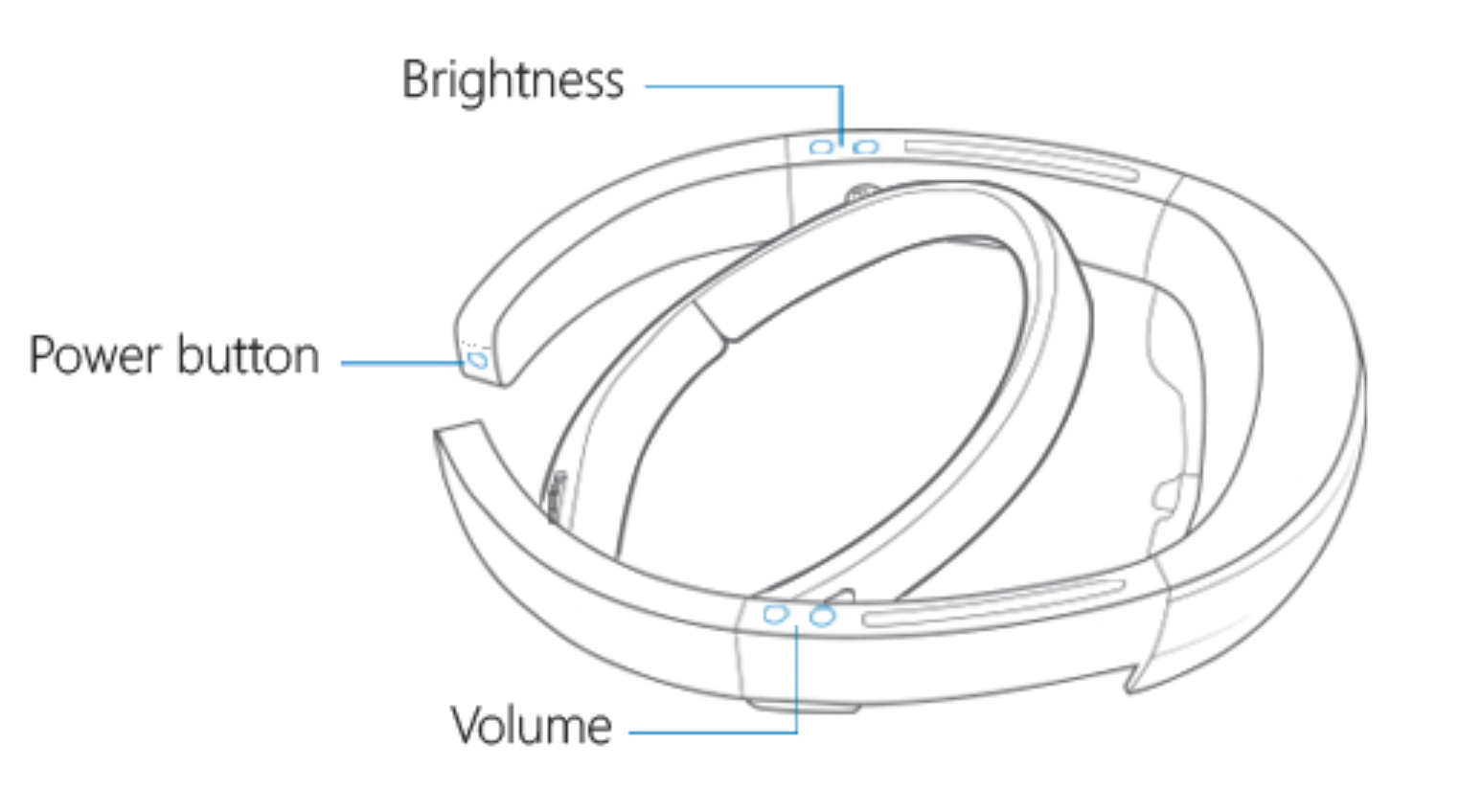


Figure 5: Power, Volume, and Brightness adjustment placement

# 3.) Unity Game Engine Project Selection

1. Once the user double clicks the app icon the project drop down menu will be opened in the Unity Game Engine and can be referenced in figure 6.
2. The user will then proceed to go to the drop down menu under the projects file, figure 5, and find the corresponding project to select. This will open up that project. The example from figure 5 is the folder my\_first\_unity.

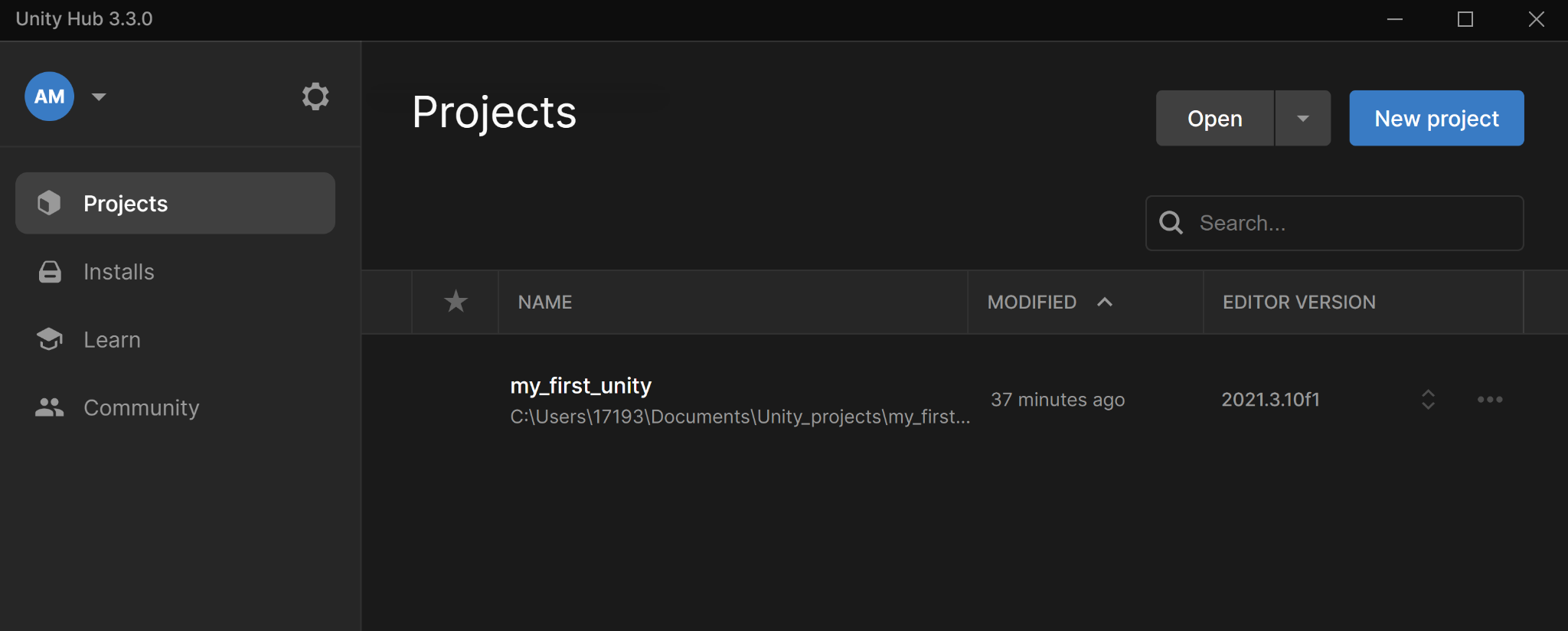


Figure 6: Unity Game Engine list of projects

# 4.) Unity Game Engine Build Settings

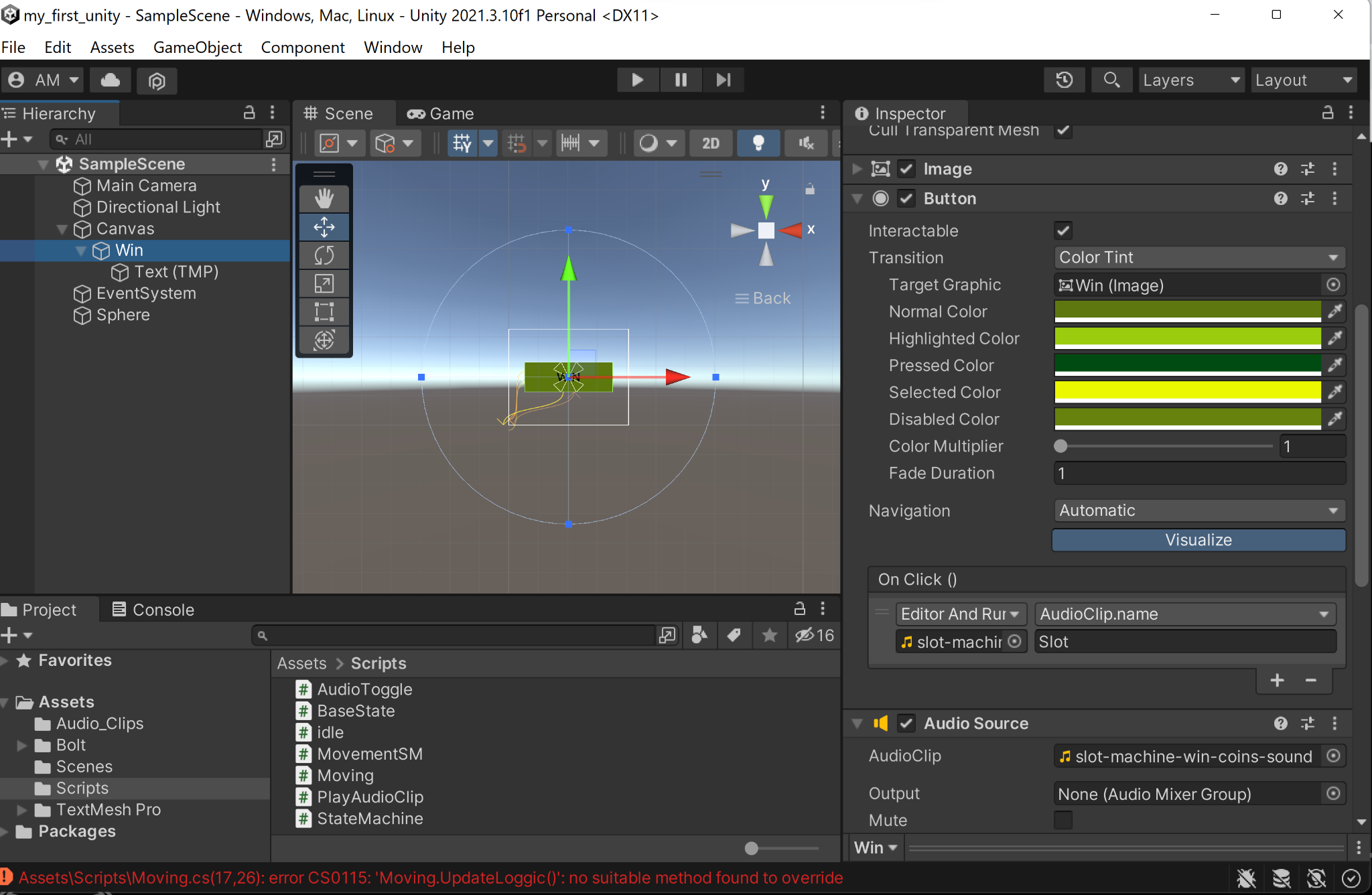


Figure 7: Unity Game Engine Project opened

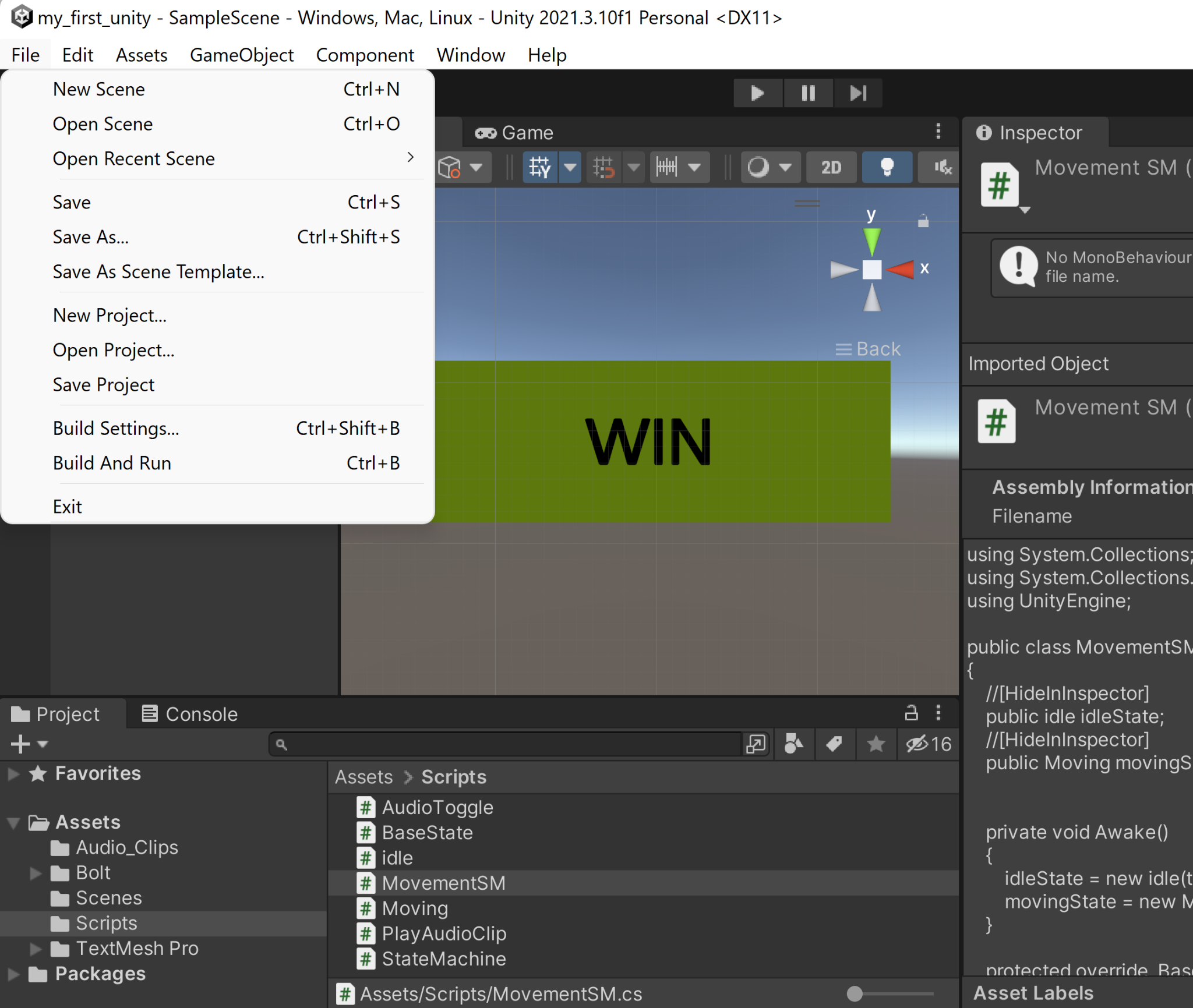


Figure 8: Unity Game Project file drop menu

1. When the user successfully opens up the project file following figure 5 the window the user will see if using the Unity 2021.3.10f1 version will look like figure 6.
2. The user will need to configure the game settings in the unity game engine by selecting the file drop down menu and selecting build settings. This view can be found in figure 7.

# 5.) Universal Windows Platform Configuration Settings Using Unity Game Engine

Note: The build settings will provide access to the Universal Windows Platform settings, figure 8. In order to successfully launch the project onto the Hololens the user will need to configure the Universal Windows Platform settings to properly render the game.

1. The first step in this process is to select the proper platform in use which in this case must be the Universal Windows Platform.
2. The user must first make sure to select the “Add Open Scenes”, this will automatically select the current project scene that is open and enable the platform configuration settings for that particular Scene/Project.
3. The platform settings must include the following configuration which can be referenced in table 2 and figure 8.
4. Once these settings have been configured the user must press build and run.

### Figure 9: Universal Windows Platform configuration settings.

| Universal Windows Platform | Settings |
| --- | --- |
| Target Device | Hololens |
| Architecture | x64 |
| Build Type | D3D Project |
| Target SDK Version | Latest Installed |
| Minimum Platform Version | 10.0.10240.0 |
| Visual Studio Version | Latest Installed |
| Build and Run on | Local Machine |
| Build Configuration | Release |
| Copy References | Unselected |
| Copy PDB Files | Unselected |
| Development Build | Unselected |
| Autoconnect Profiler | Unselected |
| Script Debugging | Unselected |
| Scripts Only Build | Unselected |
| Compression Method | Default |

### 

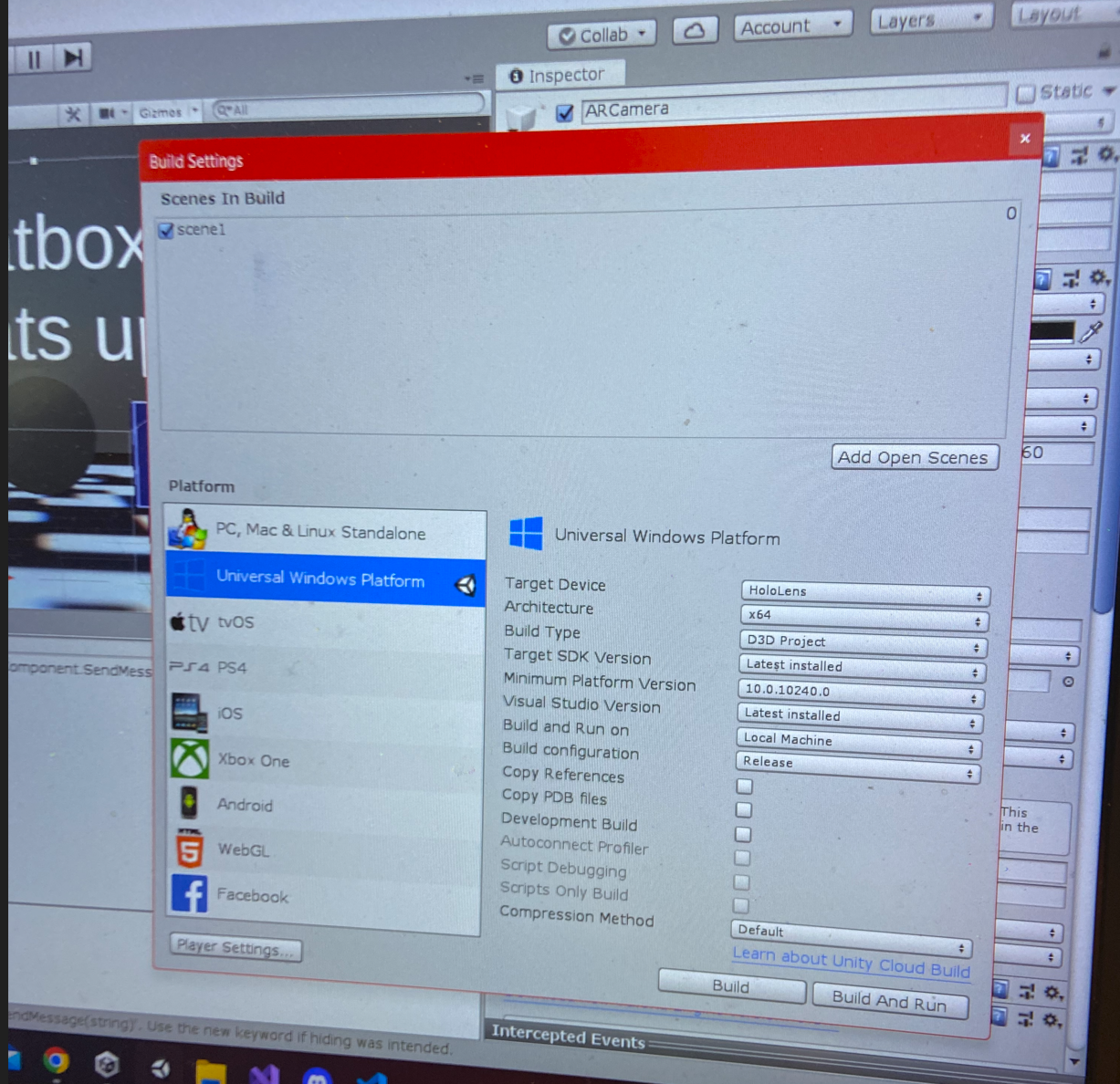


Figure 10: Unity Game Engine build settings

# 6.) Using the HoloLens Emulator for Application Launch

Note: If the user decides to use the HoloLens Emulator the configuration can vary slightly.

1. First the solution configuration setting must be changed from debug to master found on the upper left corner. That gives the user the option to run the project on the device, the HoloLens.
2. Make sure to have the HoloLens powered on and situated on head when pressing the start button on the HoloLens Emulator because the start will automatically launch the app on the HoloLens.

# 7.) Using the Application

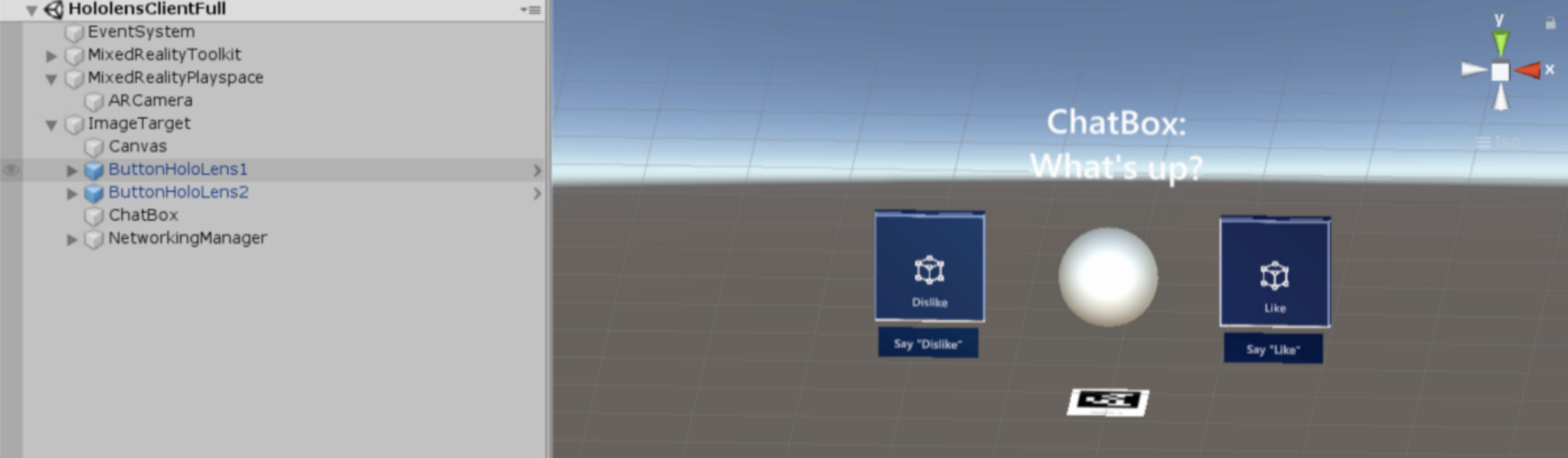


Figure: HoloLens Emulator view

1. Click on the application icon using hand held remotes.
2. Once the app launches, the HoloLens comes with a pair of hand held remotes one for each hand, the user will be able to use either remote to click on the app button to launch the game. The button will look like the buttons dislike and like in figure 9. Using the hand held remotes the user will be able to launch the app by clicking the app button.

After it is launched the user will no longer need to configure settings or click on anything. The HoloLens will interact with the user through audio events and graphical overlays but the user will no longer need to do anything until the user needs to shut down the system.

# 8.) System Shutdown

1. Power the HoloLens down. Close the Emulator by exiting the scene using the x to close down the game. The user can also use the Unity Game Engine to press the start and stop button on the top of the scene, figure 10.

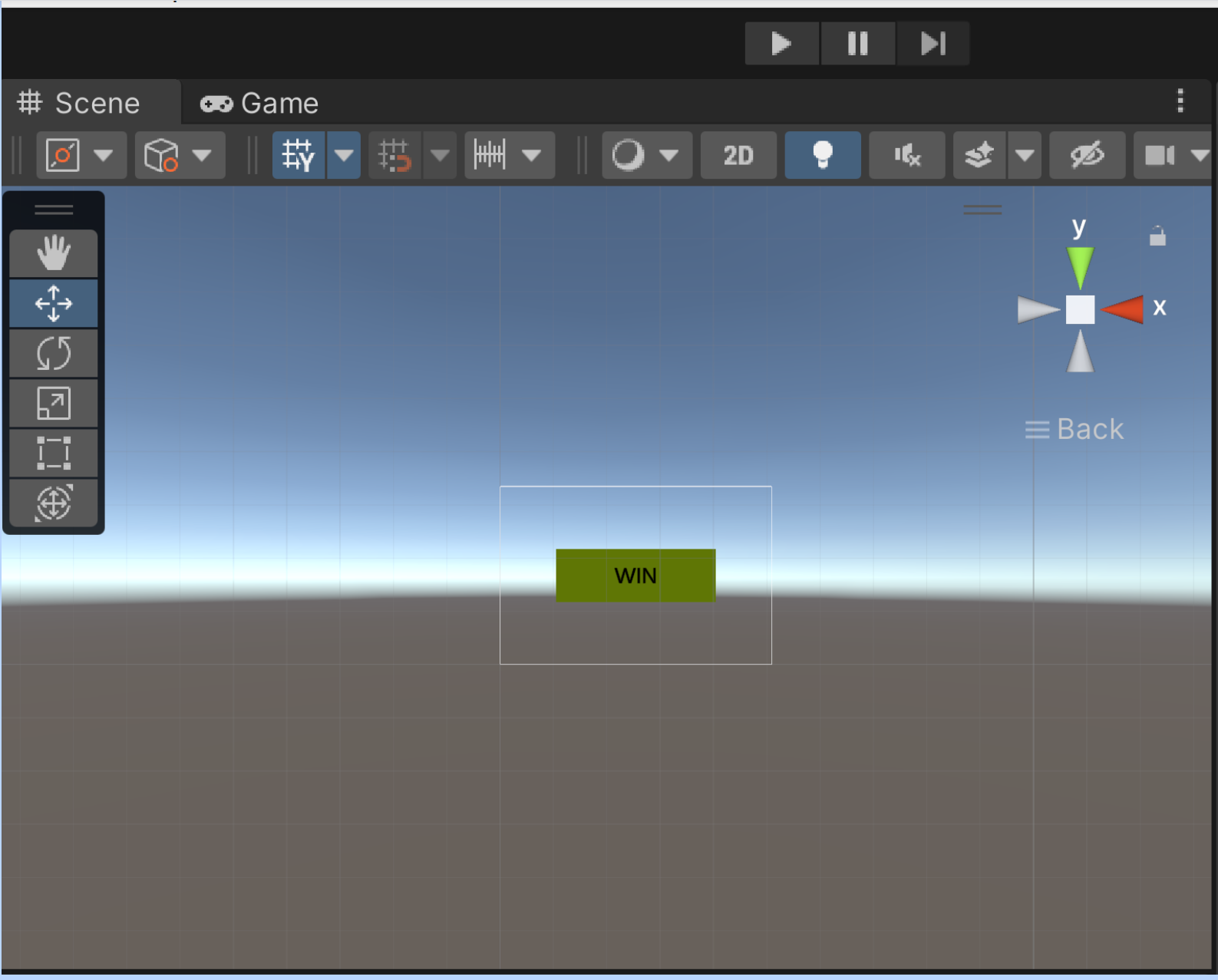


Figure 11: Unity Game Engine scene view

**Detailed Drawings of Parts, Schematics, Software Descriptions**

For detailed drawings of parts and schematics see the previous section, User Manuals and Instructions.

Software Description

The AR/VR Blackjack project relies heavily on software. Software is being built to run an AR game on the Microsoft Hololens. The software is built by using the Unity Editor. With the Unity Editor, it is possible to build a game for the Microsoft Hololens featuring graphic elements for the UI, as well as the logic required to run the game correctly. This logic includes the math required to calculate the probabilities of winning each round of Blackjack.

So far into the project, code has been obtained from a github repository to be able to perform a TCP connection via a python script, which allows communication between Python programming and the game app for the Hololens. Details on this process is that there is a Python script: “FinalServer.py '' which creates the server via a wi-fi connection. It does this by importing the module “socket”. This allows the creation of socket objects which help communicate and create the TCP server with the computers ip address, and the desired port number.

To run this Python script successfully, it is required to run on a Linux OS. This is due to conflicts of the use of file handles in Windows. Thus, Oracle VM Virtualbox will be used to run a virtual machine which has Ubuntu 16.04.7 installed to run this script.

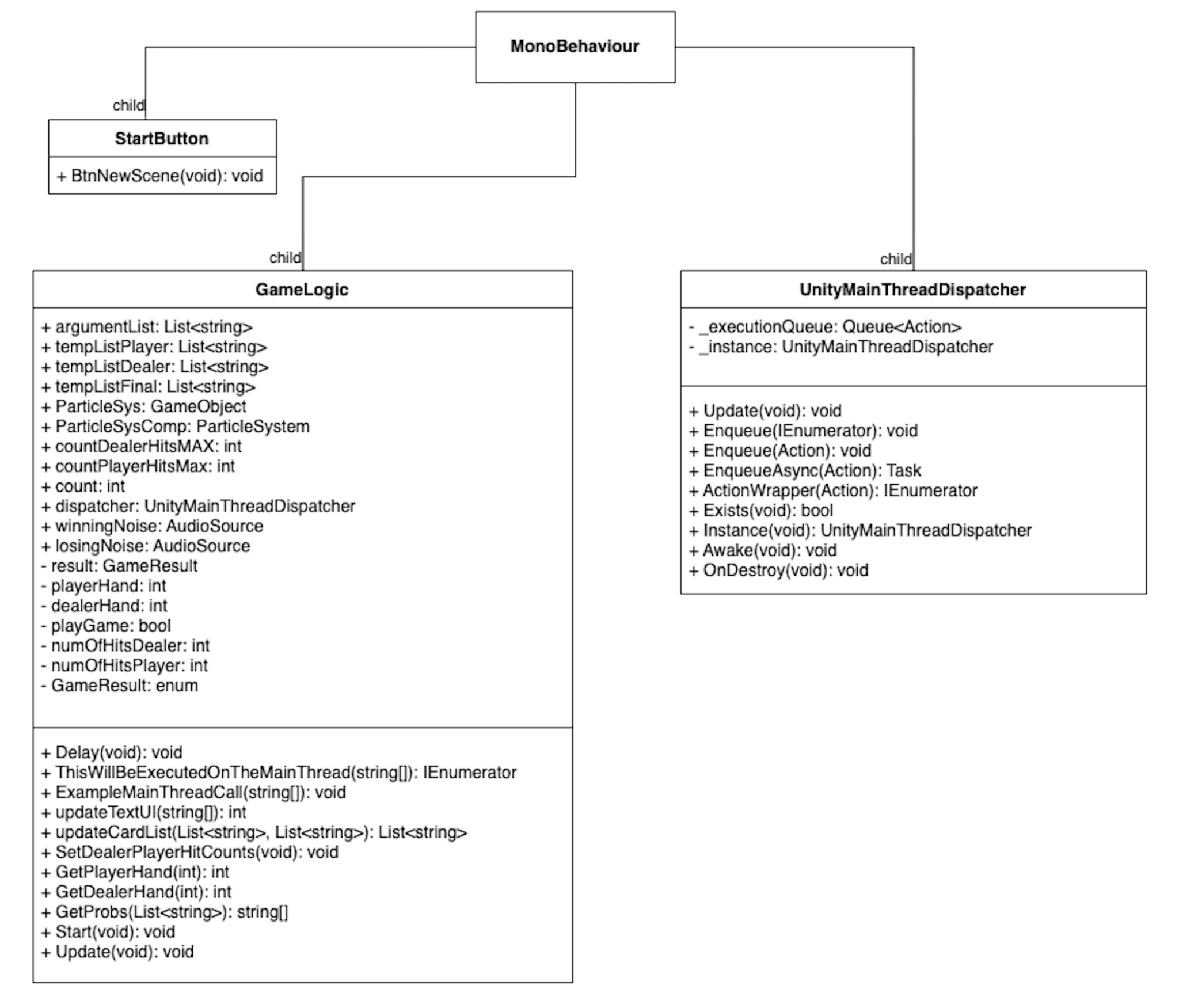
The Unity Editor has its own C# script: “UnityClient.cs” which serves as a client to the server which allows it to receive the data being transmitted by the created Python server. This data is suspected to be the results of the probability math algorithm.

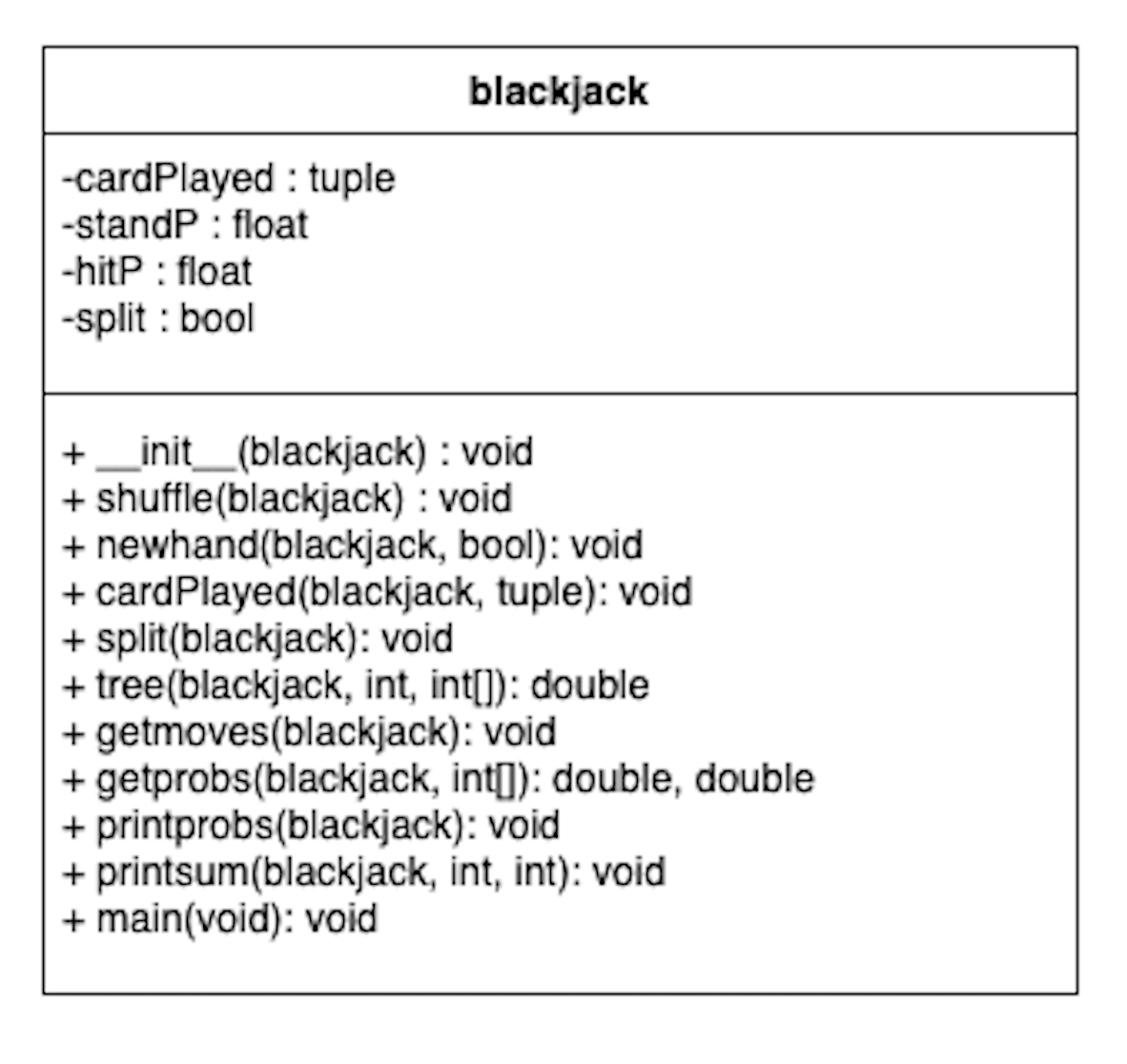
Other software that is being created for this project is the Unity Program solution, and scripts which help control Game Objects, as well as the Game Objects themselves. Game Objects allow the creation of the AR camera, UI, text for UI, and music being used for the AR/VR Blackjack assist tool. Furthermore it allows the use of Barracuda to be able to launch the object detection algorithm: “YOLO”. To do this a UNNX file provided by the github repository is used to launch YOLO, and to train the algorithm by launching a python script which performs the training for the algorithm to detect playing cards correctly. To train the algorithm, many photos of different playing cards must be fed in.

Finally, to put everything together, a C# script will be built to be able to send the object detection data over to the card probability python script: “blackjack.py”. The blackjack.py script is responsible for calculating whether it is more beneficial to stay or hit in the game of Blackjack. It does this with the help of importing the package “numpy”, which helps by easily creating arrays/tuples, and to easily perform math functions like finding the max, or the sum of these arrays. Additionally, it allows the creation of arrays filled with 0’s. Using numpy makes it easier to write the probability algorithm.

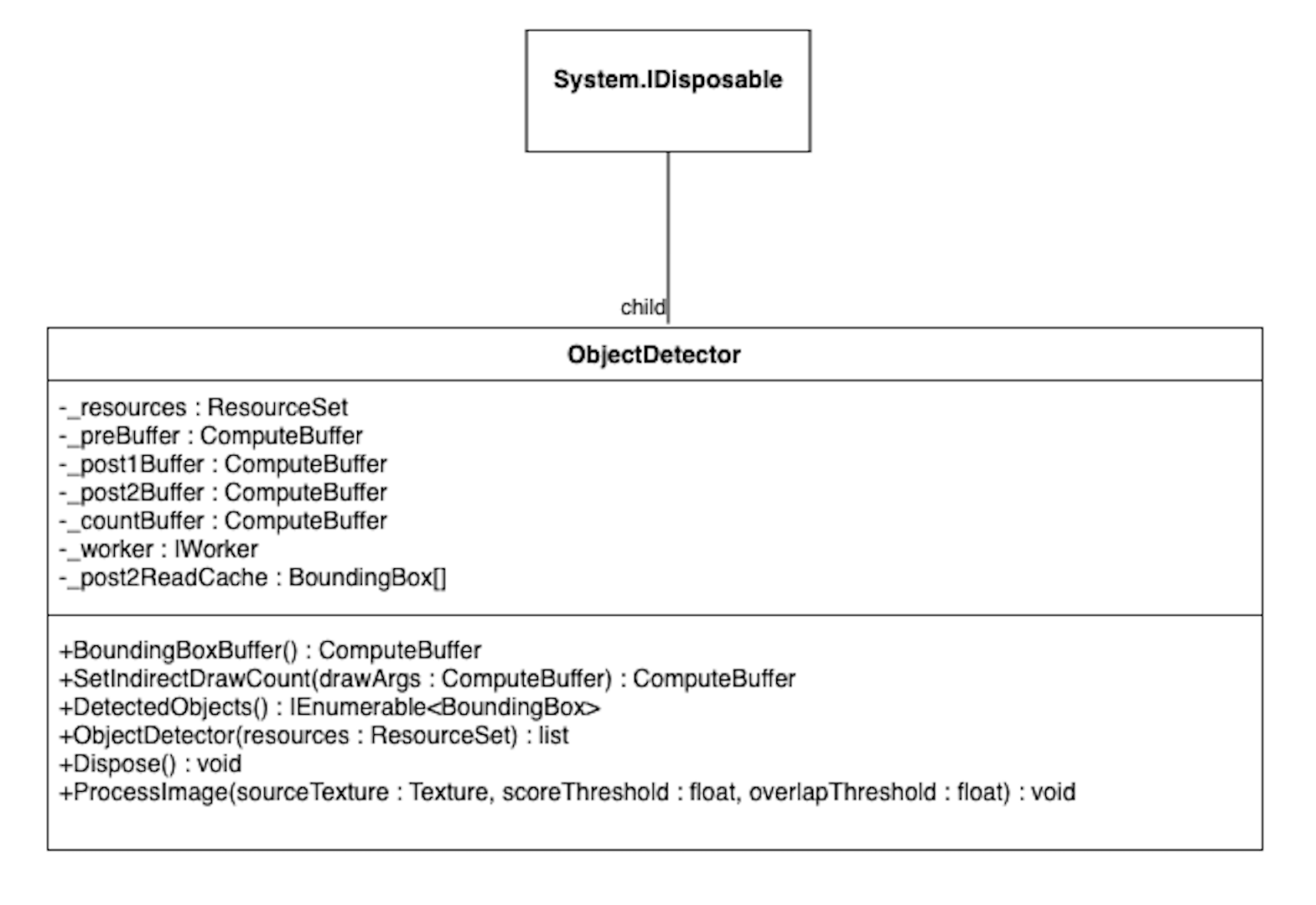
UML Diagrams

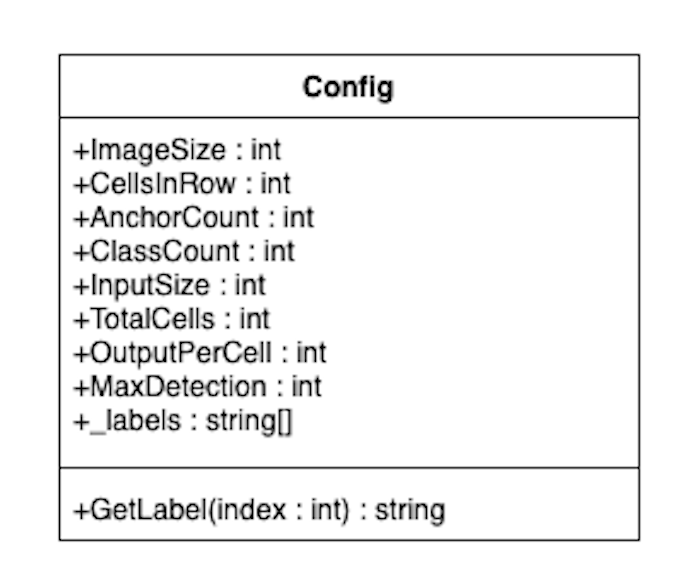
Unity Project:

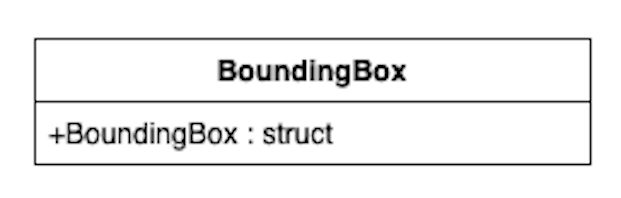


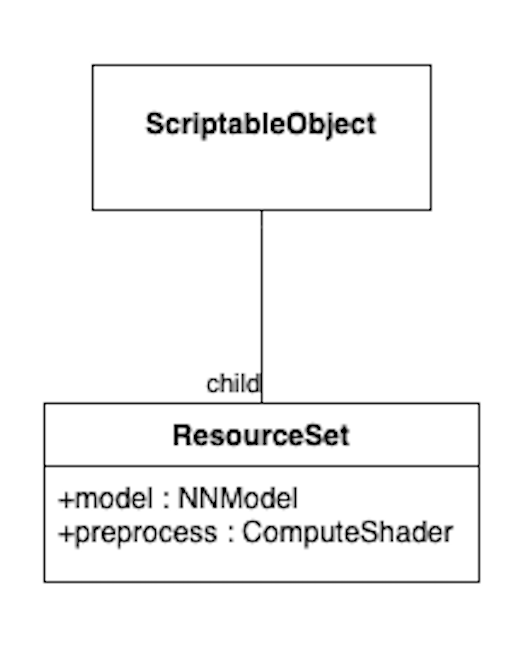


Object Detection:









**Bill of Materials**

| **Item** | **Cost** |
| --- | --- |
| Microsoft HoloLens | $810 |
| AX1800 Dual Band Wireless Router | $40 |
| Deck of Cards Bicycle Brand | $3.50 |
| **Total** | $853.50 |

**Resources**

[1] Unknown. “Microsoft Ignite.” Learn Microsoft. URL (accessed October 2, 2022). [Prepare a new HoloLens | Microsoft Learn](https://learn.microsoft.com/en-us/hololens/hololens1-setup)