



AR Remote Rendering Application

Conceptual 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

Description of solution concepts ……………………………………………………………. 4

Comparison of solution concepts to problem requirements ……………………………… 5

Decision matrix ……………………………………………………………………………….. 6

Concept recommendation …………………………………………………………………… 6

Plan to complete the design ………………………………………………………………… 7

Test Plan ……………………………………………………………………………………… 9

Introduction …………………………………………………………………………… 9

Methodology and Procedure ……………………………………………………….. 10

Results and Analysis ………………………………………………………………… 15

Testing Report ……………………………………………………………………………….. 17

Introduction …………………………………………………………………………… 17

Methodology and Procedure ……………………………………………………..… 17

Results and Analysis ………………………………………………………………… 23

Discussion ……………………………………………………………………………. 30

Conclusion …………………………………………………………………………… 31

References …………………………………………………………………………………… 33

Appendix ……………………………………………………………………………………… 34

**Description of Solution Concepts**

The AR/VR device is at the core of the project composed of two primary constituents with the primary being modeled towards the device’s performance which includes the memory, battery life, management app, and runtime. While the secondary is customized towards achieving a higher quality player’s game for unskilled gamers. This will be realized with the addition of visual effects such as overlays and player performance statistics, object identification, and audio effects. Here the hardware and software improvements would amplify the player’s experience towards an informed and option driven run.

There have been a few algorithms that the team has considered for object identification. By implementing this feature into our AR application would allow an unskilled player to utilize the AR/VR device to identify the playing cards the player holds. The algorithm being pursued right now in this phase of the project is YOLO. YOLO is marketed for its speed and accuracy. It appealed to the team because it can be utilized with any programming language and therefore, we have been exploring the several programming options to determine which one would be best suited for the AR/VR device and the management app software. As a team, there exists uncertainty around commitment to a singular language due to some programming language barriers. The plan currently is use both Python and C# languages.

Over the Summer one of the team members had a dry run on the plan of execution for a particular chosen software and there were issues with language translation between different interfaces. Therefore, the teams have chosen to remain open and try numerous software models before moving forward to see which yields the best result and overall device interface.

The original sponsor of this project wanted to push the limits of the device and venture into hardcoding new technology specs for the AR/VR device. To balance the technical performance of the device, audio and visual effects were decided upon for implementation. While the game/application is running, visual effects, object detection, overlays, and player statistics will be consistently rendered. The player statistics will be using an algorithm focused on learning the player’s chance of winning by counting cards of the game. The object detection was discussed in the section above concerning the YOLO algorithm. Concerning the overlays, originally the discussion included overlays that were more player engaging and fun using color scheme for next move options that included the statistic for winning. Other overlays included a box encapsule for the object being identified. Unity game engine will be used for most of the overlays and audio effects as class actors, game actors, and child actors of classes (e.i.) sound can be a child actor of the colored statistics being displayed. Individual sounds and color will be components of the parent class actor statistics. Visual scripting will be the first approach to explore audio and visual effects. Hardcoding in the engine will require language research on macros and language specifiers which will take longer and may face more interface backlash. If the algorithms interfacing do indeed require hardcoding then that route will be taken, otherwise live scripting in Unity Game Engine will be the first approach.

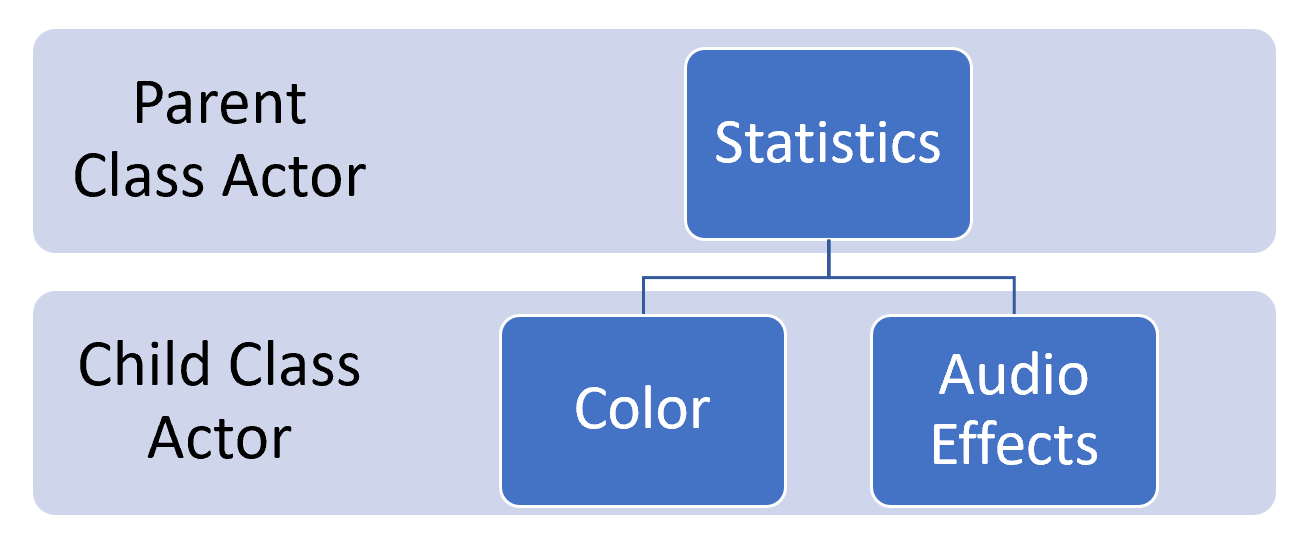
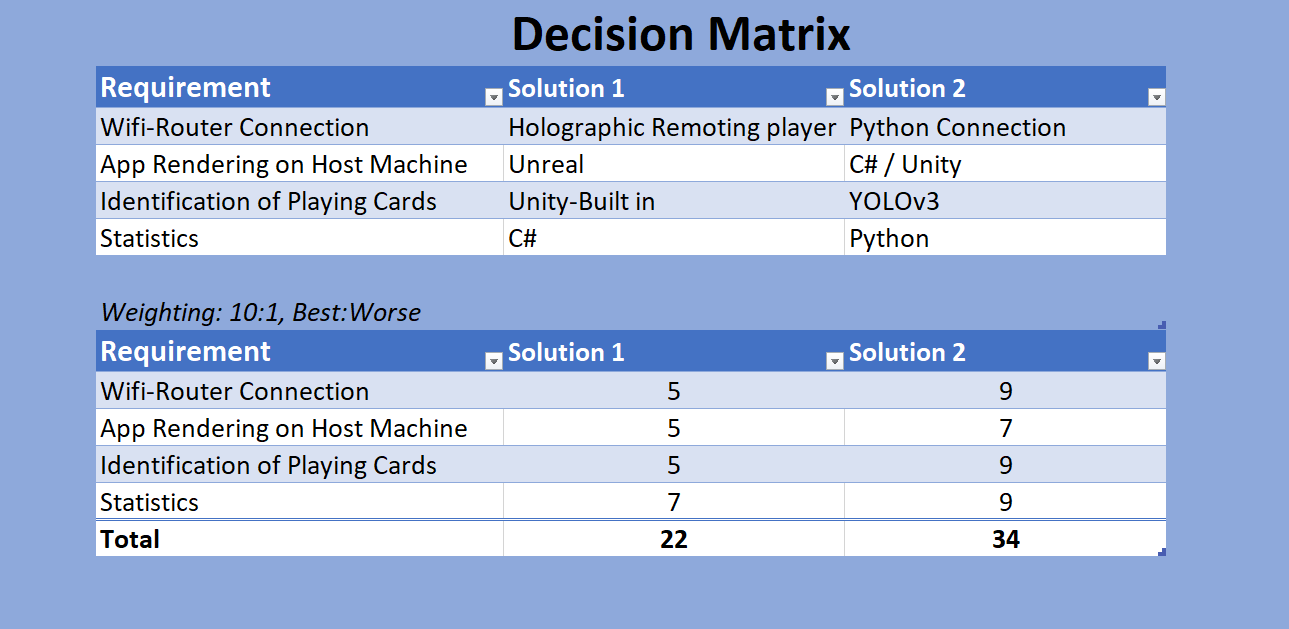
****

Figure 1

**Comparison of Solution Concepts to Problem Requirements**

1. The connection to the wifi router will be done via python TCP connection. The game will be streamed from the host computer in Unity.
2. The app rendering will be done in Unity, which allows for all built-in helper functions for the hololens to work properly. This solution also allows easy deployment to the hololens via python.
3. The Battery runtime shall be improved since the rendering will be done on the host machine. Since the application will be created in Unity, we can compare the runtime of the application on the hololens itself vs running the application remotely.
4. Object Detection will be done via the YOLOv3 machine learning Object Detection algorithm. First, models of playing cards will be trained so that the algorithm can easily identify the number and suit of each card. Then the number of the card will be displayed on top or next to the card in real time since the processing will have been done before.
5. The logic for the statistics will also be done in python and then fed into unity for use in the program. Statistics known: (1) probability to stand, (2) probability to hit

**Decision Matrix**



**Concept Recommendation**

A lot of ideas have been brainstormed with respect to a solution for the Microsoft HoloLens AR/VR Blackjack game. The main factor was in the decision on how the object-detection will be done. Thus, for the project concept, it is recommended that the best solution is to use YOLOv3 which is an AI that is trained to detect objects through a camera. Using YOLOv3, we can train the AI to detect cards for their symbol, and the number (A, J, Q, K, and 2-10).

YOLOv3 thus gives us the top-scoring solution to perform object-detection with machine learning opposed to trying to do object-detection using some built-in Unity code, which is uncertain that it will work. Using Unity for doing object detection is uncertain because it seems to only detect the environment around in some form of pattern using something called Meshes, it does not seem to do object detection.

Another imperative recommendation for the concept, is to use a game engine that will be able to interface with the Microsoft HoloLens, with the options of Unity or Unreal. There does not seem to be any other ways to interface with the Microsoft HoloLens. It has been decided to use Unity because there seems to be more resources/documentation and seems to be more simple compared to using the Unreal game engine. Additionally we know that YOLOv3 is compatible with Unity. Thus we can gain access to the HoloLens cameras. Thus, we chose to use Unity as the host for running our app, as opposed to Python by itself.

Using a Python script, a TCP server will be implemented to the HoloLens project. The TCP will be used to send any data that is needed regarding AR, or the YOLOv3 data for the object detection functionality. Furthermore, the HoloLens will be able to connect to WiFi, and then use the client can formed through a C# script on Unity, and a TCP connection can be made.

The WiFi connection is needed for the TCP connection to work. This solution is chosen in case the transmission of data needs to be made through Python, which might make the programming for the project easier and more effective. This solution is a top-scoring solution with a weight of 9, opposed to a weight of 5, when using the Holographic Remoting Player. The Holographic Remoting Player runs apps/games over a WiFi connection, and doesn’t provide a TCP server.

Finally, the Statistics and probability calculations will be done with Python as opposed to C#. The reason being that Python is being used already for the TCP server, and that Python is more simple to use, and offers a lot of modules that can easily be installed which could help.

**Plan to Complete the Design**

**Phase 1 (Mar 4, 2022 – Apr 15, 2022):** Research was conducted on AR devices and remote rendering applications by the entire team. The entire team consulted with Dr. Omid Semiari to get the specifics of the project. William B. Micheal was contacted in order to collect the equipment that will be utilized for the duration of this project: a HoloLens One and a WiFi AX1800 Dual Band Wireless Access Point Router. The Requirements and Specification Document was written including defining team and project sponsors, the project specification intent, project requirements, parameters and target values, and the agreement to the project specification.

**Phase 2 (Apr 15, 2022 – May 6, 2022):** During Phase 2 the team worked on the Requirements and Specifications Presentation. This presentation went over the project’s problem, current standards, constraints, the requirements and specifications, the system design expectations, issues, timeline, budget, and societal concerns.

**Phase 3 (Aug 26, 2022 – Sept 12, 2022):** Phase 3 was focused on writing the Conceptual Design Report excluding the Testing Report. In this report the team has put together a decision matrix, concept recommendation, a plan to complete the design, and the testing plan.

**Phase 4 (Sep 9, 2022 – Sep 16, 2022):** The Conceptual Design Presentation will be created and given in Phase 4. This presentation will present a summary for each section of the report that was put together in the previous phase.

**Phase 5 (Sep 16, 2022 – Sep 30, 2022):** In this stage each member of the team will contribute to the Critical Design Documents including Detailed Description of Design, User Manuals, and Instructions, Detailed Drawings of parts, Schematics, Software Descriptions, Bill of Materials.

**Phase 6 (Sep 30, 2022 – Oct 7, 2022):** After finalizing the Critical Design Documents in the previous phase, our team created and practiced a presentation that showcased the important aspects of the documents.

**Phase 7 (Oct 8, 2022 – Nov 21, 2022):** Although the team had been working on the actual design of the project since May, in this phase the team solely focused on completing the AR Blackjack Assistant Remote Rendering Application which also incorporated the application’s testing.

**Phase 8 (Nov 6, 2022 – Dec 7, 2022):** In this part of the project the team finished the Conceptual Design Documents by writing the Testing Report. This summarized the testing that was completed at the end of Phase 6. During this timeframe, the team also concluded that object detection integration was no longer a feasible option due to a time constraint.

**Phase 9 (Nov 21, 2022 - Nov 27, 2022):** Phase 9 is where our team concentrated on writing this report, the Final Report. This report includes an Executive Summary, an Introduction, a Problem Description, a Conceptual Design, a Finalized Design, a Testing Analysis Summary, a Conclusion, and the Final Design Package.

**Phase 10 (Nov 28, 2022 - Dec 9, 2022):** A final presentation will be developed and given discussing the project, problem, conceptual design, finalized design, and the testing and analysis.

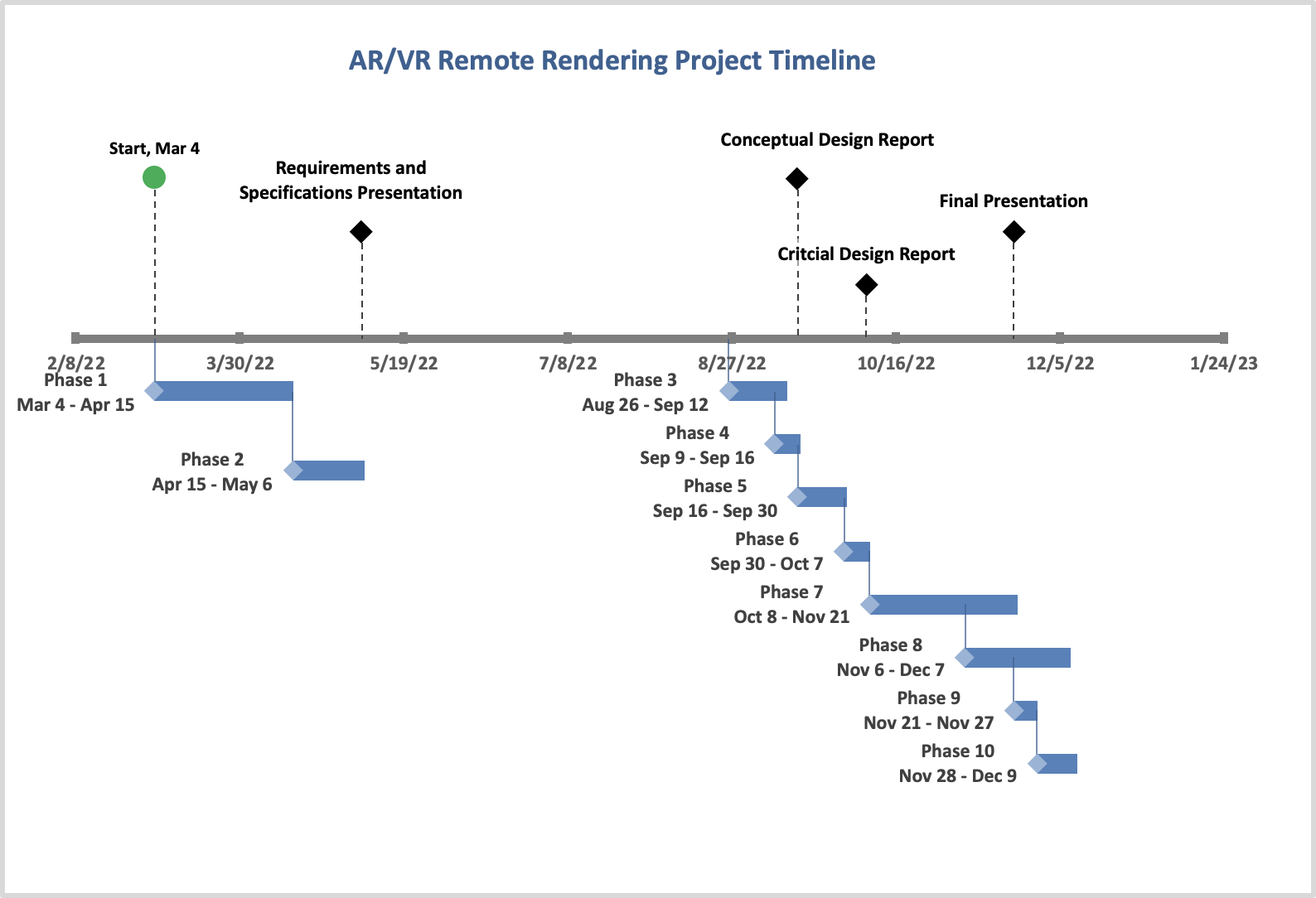


Figure 2

See Appendix for Gantt Chart.

TEST PLAN

**Introduction**

The test plan is a verification process that demonstrates the software and hardware features of the AR/VR device functioning correctly. The test plan is written to outline the steps that are necessary to produce a high-quality product that meets all requirements listed on the specification sheet. The AR/VR device testing will accomplish verification of successful connection to the Wi-fi router. Testing a solid connection to the router will allow the device to connect to the host computer for processing with the app. All these tests will allow observations on the battery life of the device. Our projections are that having a host computer process will allow battery life improvement.

Another aspect of testing is also attributed to the user and their interaction with the final product. Once the hardware and software have been fully assessed, user interaction is something that will need to be tested to fulfill the project's overall mission. This test plan will allow the team to study and gather data on the operations of the system in individual components and as a whole.

**Methodology and Procedure**

**Methodology**

This AR remote rendering application will be tested utilizing the Hololens 1, Wi-Fi, a computer, and a bench-mark app (non-remote rendering).

**Procedure**

**Testing Classes individually**

GameLogic

1. Create two variables, one for the total card count for the player and one for the dealer.
2. Change the variables so that you can test each possible outcome.
   * + Settings
       1. Surrender - Player hand = 0, Dealer hand = Any
       2. Player Busts - Player hand > 21, Dealer hand = Any
       3. Continue Playing - Player hand = Any, Dealer hand <= 16
       4. Player Wins - Player hand > Dealer hand, Dealer hand != 16 or Dealer hand > 21 and Dealer != 16
       5. Dealer Wins - Dealer hand > Play hand and Dealer hand != 22
       6. Push - Player hand = Dealer hand

Blackjack (probability)

* + Launch a Jupyter notebook
    - Open your terminal and navigate to the directory where you would like to save your notebook.
      1. Type the command jupyter notebook, the program will instantiate a local server at localhost: 8888 (or another specified port)
      2. A browser window should immediately pop up with the Jupyter Notebook interface, otherwise, you can use the address it gives you.
    - Create a new notebook
      1. Go to New and select the Notebook you’d like to use or Click Upload a Notebook you want to use.
      2. Notebooks currently running will have a green icon, while non-running ones will be gray. To find all currently running notebooks, click on the Running tab to see the list.
    - Run Cells
      1. Cells are how notebooks are structured and are the areas where you write your code.
      2. To run a piece of code, click on the cell to select it, then press SHIFT+ENTER or press the play button in the toolbar or you can utilize the Cell dropdown menu which has several options to run cells.
    - After a Cells are ran
      1. The output of the cell’s code will appear below it. To stop running a piece of code, press the stop button.
    - Example run
      1. In [1]: Entire blackjack class
      2. In [2]: game = blackjack()
      3. In [3]: game.newhand(shuffleafterhand=True)
      4. In [4]: game.cardplayed((9,1)) # Dealer gets 9

game.cardplayed((12,0)) # User gets Q

game.cardplayed((9,1)) # Dealer gets 9

Output:

Stand Probability = 22.0%

Hit Probability = 44.72%

Split? None

* + - 1. In [6]: game.cardplayed((12,0)) # User gets Q
      2. In [7]: game.getmoves()

game.printprobs()

Output:

Stand Probability = 87.95%

Hit Probability = 33.25%

Split? False

* + - 1. In [8]: game.newhand(shuffleafterhand=True)
      2. In [9]: game.cardplayed((9,1)) # Dealer gets 9

game.cardplayed((14, 0)) # User gets A

game.cardplayed((6,1)) # Dealer gets 6

game.cardplayed((14,0)) # User gets A

* + - 1. In [10]: game.getmoves()

game.printprobs()

Output:

Stand Probability = 80.72%

Hit Probability = 37.68%

Split? False

* + - 1. In [11]: for j in range(2,10):

for i in range(3):

game.cardplayed((j,2)) # extra tests

* + - 1. In[12]: game.getmoves()

game.printprobs()

Output:

Stand Probability = 41.83%

Hit Probability = 61.05%

Split? True

Object Detection

Currently, the detection works via a webcam & the built-in darknet detector demo [2]. This demo runs interference on the input webcam frames, and then draws bounding boxes on the output video feed frames. Prerequisites for the demo:

* Darknet [2] must be compiled.
* A trained weights file must be generated.
* A webcam is preferred but the detection does work on static images.

To run the demo:

1. Plug in the webcam into the host computer.
2. Position the webcam vertically above the cards to be detected and make sure the scene has plenty of light.
3. Navigate to the darknet folder containing the build files.
4. Run the following command from windows PowerShell:

**darknet** detector demo .\data\obj.data .\cfg\yolov4-tiny-obj3.cfg .\backup\yolov4-tiny-obj3\_20000.weights -c 0

**.\data\obj.data** → location of the custom object data file.

**.\cfg\yolov4-tiny-obj3.cfg** → location of the custom object configuration file.

**.\backup\yolov4-tiny-obj3\_20000.weights** → location of the trained WEIGHTS file.

**-c 0** → Use system webcam with ID=0

1. Observe the output in the video feed window with the bounding boxes drawn on. The identified cards will also appear in the command line window.

Testing Unity Project

In this section we will be testing all the classes together within Unity without the Object Detection input.

1. Within the Start function of the GameLogic class contained in the Unity project, change the values that are added to the argumentList. Use values that will give you each possible outcome.

Note: The values added after “p” will be added together to get the Player’s hand and every value after “d” will be added together to get the Dealer’s hand.

Settings

* + - Surrender - Player hand = 0, Dealer hand = Any
    - Player Busts - Player hand > 21, Dealer hand = Any
    - Continue Playing - Player hand = Any, Dealer hand <= 16
    - Player Wins - Player hand > Dealer hand, Dealer hand != 16 or Dealer hand > 21 and Dealer != 16
    - Dealer Wins - Dealer hand > Play hand and Dealer hand != 22
    - Push - Player hand = Dealer hand
  + After the changes have been made and saved in the GameLogic class. Run the Unity project by pressing the play button at the top middle of the screen.

Full System

**Setting up HoloLens to use Windows Device Portal**

1. Power on your HoloLens and on the device.
2. Use the Bloom on HoloLens to launch the main menu.
3. Gaze at the Settings tile and do an air-tap gesture on HoloLens.
4. Select the Update menu item.
5. Select the For developers menu item.
6. Enable Developer Mode.
7. Scroll down and enable Device Portal.
8. Set up the Windows Device Portal so you can deploy apps to this HoloLens over Wi-Fi, select Pair to generate a pairing PIN. Leave the Settings app at the PIN popup until you enter the PIN into Visual Studio during your first deployment.

**Connecting over Wi-Fi**

1. Connect your HoloLens to Wi-Fi.
2. Look up your device’s IP address by either:
   1. Going to Settings > Network & Internet > Wi-Fi > Advanced Options.
   2. Going to Settings > Network & Internet and selecting Hardware properties.
   3. Using the “What’s my IP address?” voice command.
3. From a web browser on you PC, go to https://<YOUR\_HOLOLENS\_IP\_ADDRESS>
   1. The browser will display the following message: “There’s a problem with the website’s security certificate” because the certificate, which is issued to the Device Portal, is a test certificate. You can ignore this certificate error for now and continue.

**Installing a certificate**

1. In Windows Device Portal, navigate to the Apps manager page.
2. In the Deploy apps section, select Install Certificate.
3. Under Select certificate file (.cer) used to sign an app package, select Choose File and browse to the certificate associated with the app package that you want to sideload
4. Select Install to start the installation.

**Installing the benchmark/non-remote rendering application**

1. When you’ve created an app package from Visual Studio, you can remotely install it onto your device from the generated files.
2. In Windows Device Portal, navigate to the Apps manager page.
3. In the Deploy apps section, select Local Storage.
4. Under Select the application package, select Choose File and browse to the app package that you want to sideload.
5. Check the respective boxes if you want to install optional or framework packages along with the app installation and select Next.
6. Select Install to start the installation.
7. Once the installation is complete, go back to the All apps page on your HoloLens and launch your newly installed application.

**Instantaneous System Performance Readings**

1. From the Windows Device Portal home page on the Microsoft HoloLens, Select the Views from the menu on the left.
2. Then select Apps.
   1. Ensure that the only app that is currently running is the one being tested.
   2. End any unnecessary processes that are running.
3. From the Windows Device Portal home page on the Microsoft HoloLens, Select the Performance from the menu on the left.
4. To see the instantaneous system performance, select System Performance.
   1. Log the following metrics periodically during the test:
      1. **SoC power:** Instantaneous system-on-chip power usage, averaged over one minute
      2. **System power:** Instantaneous system power usage, averaged over one minute
      3. **Frame rate:** Frames per second, missed VBlanks per second, and consecutive missed VBlanks
      4. **GPU:** GPU engine usage, percent of total available
      5. **CPU:** percent of total available
      6. **Memory:** Total, in use, committed, paged, and non-paged

**Performing a trace**

1. On the left, navigate to Performance > Performance Tracing
2. Choose an available profile or select Custom profiles > Browse then use this profile to analyze CPU performance:

<https://aka.ms/CPUProfileforDevicePortal>

1. Click Start Trace.
2. The HoloLens is now recording a trace. Make sure to trigger the performance issues that you want to investigate, and then select Stop Trace.
3. The trace will be listed at the bottom of the webpage. Select the disk icon at the right-hand side to download the ETL file.

You now have an ETL file that you can either open directly in WPA or send to someone else.

**Results and Analysis**

**Data collected**

1. SoC power: System-on-chip power usage
2. System power: System power usage
3. Frame rate: Frames per second, missed VBlanks per second, and consecutive missed VBlanks
4. GPU: GPU engine usage, percent of total available
5. CPU: percent of total available
6. Memory: Total, in use, committed, paged, and non-paged

**Reason data this collected**

1. SoC power and System power - collected to view how much power is being used while each app is being used
2. Frame rate - collected to view how fast the communication is between each app and the hololens
3. GPU, CPU, Memory usage - collected to see how efficiency of the application

**Analyzing the trace with WPA**

Windows Performance Analyzer is the standard tool to visualize traces as graphs and tables to allow you to analyze the system and application performance.

For a step-by-step guide for WPA: <https://docs.microsoft.com/en-us/windows-hardware/test/wpt/wpa-step-by-step-guide>

There are several files you’ll need to analyze your trace file in WPA:

* HoloLens trace file (\*.etl)
* Symbol file (\*.pdb). Make sure it’s extracted (not zipped)
* WPA profile (\*.wpaProfile)

1. Set up your files in a folder for WPA to access them.
2. Launch the Windows Performance Analyzer (WPA).
3. Open the ETL trace file by selecting File > Open > select the .etl in your file browser.
4. Load symbol file into WPA by selecting Trace > Load Symbols.
5. Apply WPA profile to generate graphs for analysis. In your WPA folder, select Profiles > Apply > Browse > choose the WPA profile file > Apply.
6. Once the graphs are selected, WPA will display them in the Analysis tab.

**Analyze and compare remote/non-remote rendering applications**

Use WPA graphs and tables to discuss the results.

TEST REPORT

**Introduction**

The test plan is a verification process that demonstrates the software and hardware features of the AR/VR device functioning correctly. The test plan is written to outline the steps that are necessary to produce a high-quality product that meets all requirements listed on the specification sheet. The AR/VR device testing will accomplish verification of successful connection to the Wi-fi router. Testing a solid connection to the router will allow the device to connect to the host computer for processing with the app. All these tests will allow observations on the battery life of the device. Our projections are that having a host computer process will allow battery life improvement.

Another aspect of testing is also attributed to the user and their interaction with the final product. Once the hardware and software have been fully assessed, user interaction is something that will need to be tested to fulfill the project's overall mission. This test plan will allow the team to study and gather data on the operations of the system in individual components and as a whole.

**Methodology and Procedure**

**Methodology**

This AR remote rendering application will be tested utilizing the Hololens 1, Wi-Fi, a computer, and a bench-mark app (non-remote rendering).

As a team we were unable to achieve full integration with the object detection and unity application. Therefore, we were only able to conduct a test for each class separately, as well as testing all the classes combined within Unity minus the Object Detection input.

**Procedure**

**Testing Classes individually**

GameLogic

1. Create two variables, one for the total card count for the player and one for the dealer.
2. Change the variables so that you can test each possible outcome.
   * + Settings
       1. Surrender - Player hand = 0, Dealer hand = Any
       2. Player Busts - Player hand > 21, Dealer hand = Any
       3. Continue Playing - Player hand = Any, Dealer hand <= 16
       4. Player Wins - Player hand > Dealer hand, Dealer hand != 16 or Dealer hand > 21 and Dealer != 16
       5. Dealer Wins - Dealer hand > Play hand and Dealer hand != 22
       6. Push - Player hand = Dealer hand

Blackjack (probability)

* + Launch a Jupyter notebook
    - Open your terminal and navigate to the directory where you would like to save your notebook.
      1. Type the command jupyter notebook, the program will instantiate a local server at localhost: 8888 (or another specified port)
      2. A browser window should immediately pop up with the Jupyter Notebook interface, otherwise, you can use the address it gives you.
    - Create a new notebook
      1. Go to New and select the Notebook you’d like to use or Click Upload a Notebook you want to use.
      2. Notebooks currently running will have a green icon, while non-running ones will be gray. To find all currently running notebooks, click on the Running tab to see the list.
    - Run Cells
      1. Cells are how notebooks are structured and are the areas where you write your code.
      2. To run a piece of code, click on the cell to select it, then press SHIFT+ENTER or press the play button in the toolbar or you can utilize the Cell dropdown menu which has several options to run cells.
    - After a Cells are ran
      1. The output of the cell’s code will appear below it. To stop running a piece of code, press the stop button.
    - Example run
      1. In [1]: Entire blackjack class
      2. In [2]: game = blackjack()
      3. In [3]: game.newhand(shuffleafterhand=True)
      4. In [4]: game.cardplayed((9,1)) # Dealer gets 9

game.cardplayed((12,0)) # User gets Q

game.cardplayed((9,1)) # Dealer gets 9

Output:

Stand Probability = 22.0%

Hit Probability = 44.72%

Split? None

* + - 1. In [6]: game.cardplayed((12,0)) # User gets Q
      2. In [7]: game.getmoves()

game.printprobs()

Output:

Stand Probability = 87.95%

Hit Probability = 33.25%

Split? False

* + - 1. In [8]: game.newhand(shuffleafterhand=True)
      2. In [9]: game.cardplayed((9,1)) # Dealer gets 9

game.cardplayed((14, 0)) # User gets A

game.cardplayed((6,1)) # Dealer gets 6

game.cardplayed((14,0)) # User gets A

* + - 1. In [10]: game.getmoves()

game.printprobs()

Output:

Stand Probability = 80.72%

Hit Probability = 37.68%

Split? False

* + - 1. In [11]: for j in range(2,10):

for i in range(3):

game.cardplayed((j,2)) # extra tests

* + - 1. In[12]: game.getmoves()

game.printprobs()

Output:

Stand Probability = 41.83%

Hit Probability = 61.05%

Split? True

Object Detection

Currently, the detection works via a webcam & the built-in darknet detector demo [2]. This demo runs interference on the input webcam frames, and then draws bounding boxes on the output video feed frames. Prerequisites for the demo:

* Darknet [2] must be compiled.
* A trained weights file must be generated.
* A webcam is preferred but the detection does work on static images.

To run the demo:

1. Plug in the webcam into the host computer.
2. Position the webcam vertically above the cards to be detected and make sure the scene has plenty of light.
3. Navigate to the darknet folder containing the build files.
4. Run the following command from windows PowerShell:

**darknet** detector demo .\data\obj.data .\cfg\yolov4-tiny-obj3.cfg .\backup\yolov4-tiny-obj3\_20000.weights -c 0

**.\data\obj.data** → location of the custom object data file.

**.\cfg\yolov4-tiny-obj3.cfg** → location of the custom object configuration file.

**.\backup\yolov4-tiny-obj3\_20000.weights** → location of the trained WEIGHTS file.

**-c 0** → Use system webcam with ID=0

1. Observe the output in the video feed window with the bounding boxes drawn on. The identified cards will also appear in the command line window.

Testing Unity Project

In this section we will be testing all the classes together within Unity without the Object Detection input.

1. Within the Start function of the GameLogic class contained in the Unity project, change the values that are added to the argumentList. Use values that will give you each possible outcome.

Note: The values added after “p” will be added together to get the Player’s hand and every value after “d” will be added together to get the Dealer’s hand.

Settings

* + - Surrender - Player hand = 0, Dealer hand = Any
    - Player Busts - Player hand > 21, Dealer hand = Any
    - Continue Playing - Player hand = Any, Dealer hand <= 16
    - Player Wins - Player hand > Dealer hand, Dealer hand != 16 or Dealer hand > 21 and Dealer != 16
    - Dealer Wins - Dealer hand > Play hand and Dealer hand != 22
    - Push - Player hand = Dealer hand
  + After the changes have been made and saved in the GameLogic class. Run the Unity project by pressing the play button at the top middle of the screen.

Full System

**Setting up HoloLens to use Windows Device Portal**

1. Power on your HoloLens and on the device.
2. Use the Bloom on HoloLens to launch the main menu.
3. Gaze at the Settings tile and do an air-tap gesture on HoloLens.
4. Select the Update menu item.
5. Select the For developers menu item.
6. Enable Developer Mode.
7. Scroll down and enable Device Portal.
8. Set up the Windows Device Portal so you can deploy apps to this HoloLens over Wi-Fi, select Pair to generate a pairing PIN. Leave the Settings app at the PIN popup until you enter the PIN into Visual Studio during your first deployment.

**Connecting over Wi-Fi**

1. Connect your HoloLens to Wi-Fi.
2. Look up your device’s IP address by either:
   1. Going to Settings > Network & Internet > Wi-Fi > Advanced Options.
   2. Going to Settings > Network & Internet and selecting Hardware properties.
   3. Using the “What’s my IP address?” voice command.
3. From a web browser on you PC, go to https://<YOUR\_HOLOLENS\_IP\_ADDRESS>
   1. The browser will display the following message: “There’s a problem with the website’s security certificate” because the certificate, which is issued to the Device Portal, is a test certificate. You can ignore this certificate error for now and continue.

**Installing a certificate**

1. In Windows Device Portal, navigate to the Apps manager page.
2. In the Deploy apps section, select Install Certificate.
3. Under Select certificate file (.cer) used to sign an app package, select Choose File and browse to the certificate associated with the app package that you want to sideload
4. Select Install to start the installation.

**Installing the benchmark/non-remote rendering application**

1. When you’ve created an app package from Visual Studio, you can remotely install it onto your device from the generated files.
2. In Windows Device Portal, navigate to the Apps manager page.
3. In the Deploy apps section, select Local Storage.
4. Under Select the application package, select Choose File and browse to the app package that you want to sideload.
5. Check the respective boxes if you want to install optional or framework packages along with the app installation and select Next.
6. Select Install to start the installation.
7. Once the installation is complete, go back to the All apps page on your HoloLens and launch your newly installed application.

**Instantaneous System Performance Readings**

1. From the Windows Device Portal home page on the Microsoft HoloLens, Select the Views from the menu on the left.
2. Then select Apps.
   1. Ensure that the only app that is currently running is the one being tested.
   2. End any unnecessary processes that are running.
3. From the Windows Device Portal home page on the Microsoft HoloLens, Select the Performance from the menu on the left.
4. To see the instantaneous system performance, select System Performance.
   1. Log the following metrics periodically during the test:
      1. **SoC power:** Instantaneous system-on-chip power usage, averaged over one minute
      2. **System power:** Instantaneous system power usage, averaged over one minute
      3. **Frame rate:** Frames per second, missed VBlanks per second, and consecutive missed VBlanks
      4. **GPU:** GPU engine usage, percent of total available
      5. **CPU:** percent of total available
      6. **Memory:** Total, in use, committed, paged, and non-paged

**Performing a trace**

1. On the left, navigate to Performance > Performance Tracing
2. Choose an available profile or select Custom profiles > Browse then use this profile to analyze CPU performance:

<https://aka.ms/CPUProfileforDevicePortal>

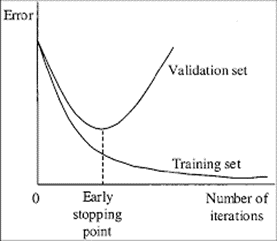
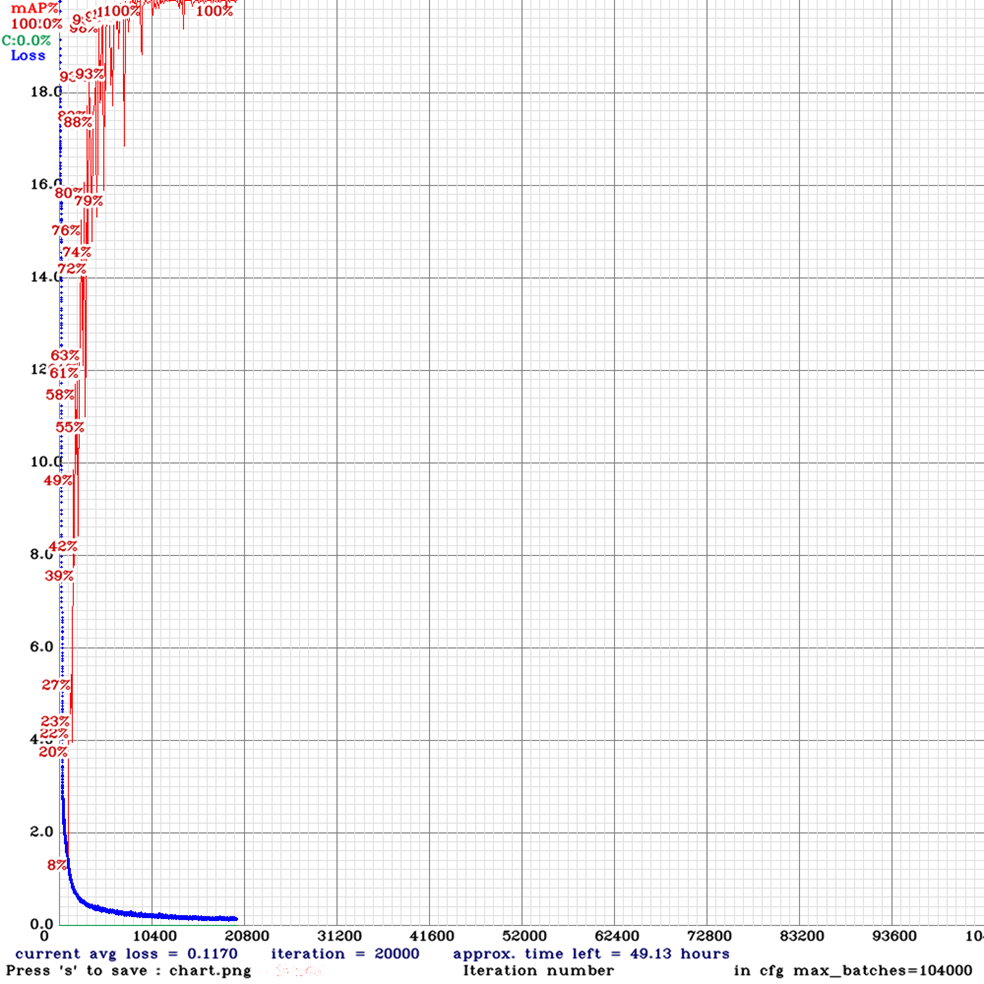
1. Click Start Trace.
2. The HoloLens is now recording a trace. Make sure to trigger the performance issues that you want to investigate, and then select Stop Trace.
3. The trace will be listed at the bottom of the webpage. Select the disk icon at the right-hand side to download the ETL file.

You now have an ETL file that you can either open directly in WPA or send to someone else.

**Results and Analysis**

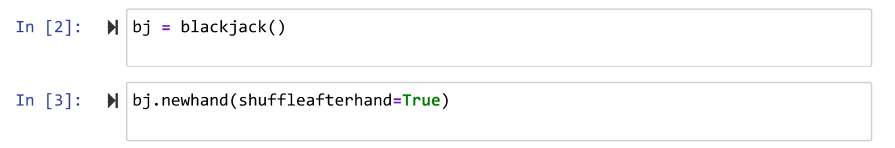
Object Detection

For testing of object detection, we can use metrics from training, as when training the current dataset, we use mAP (mean Average Precision) which describes the perceived accuracy of training by comparing the training dataset to the validation dataset. We also use average loss or error. See the plot below. Ideally, we want a high mAP (accuracy) and a low error. For around the 20,000 iterations we trained for, the below graph is shown of the mAP and avg loss. From the graph, you can see the avg loss ended up being 0.117 & mAP = 100%. We believe that the stopping point chosen, defined as “early stopping point”, can be more accurate than training for 40,000+ iterations.

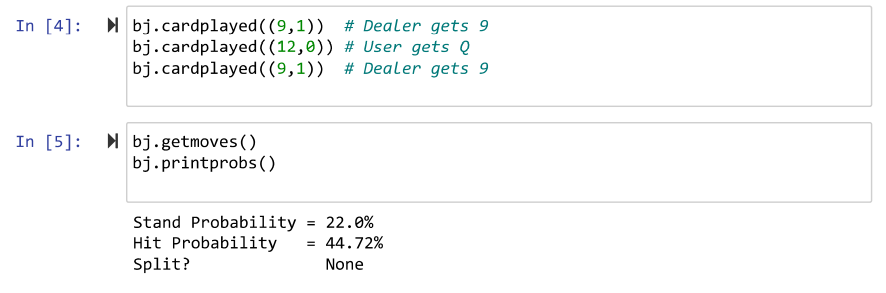


Probability Algorithm

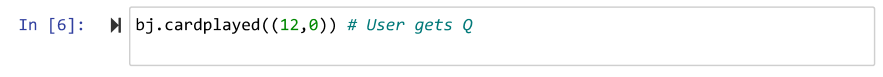
While designing the probability algorithm, a jupyter notebook was created to both demonstrate how to use the class, and test that it works as intended. A jupyter notebook is a feature of python that allows you to run code in ‘cells’, while a kernel runs in the background. The first cell is very long, so it will not be detailed here, but it defines the blackjack class that is used to perform all the necessary calculations.

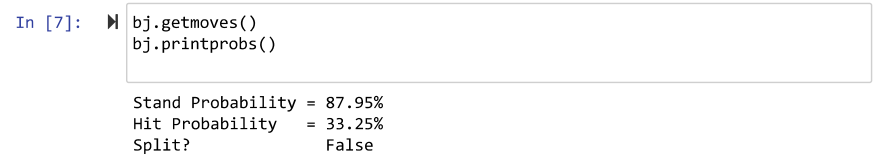


In cells 2 and 3, the class is instantiated, and a game is begun by setting up a new hand.

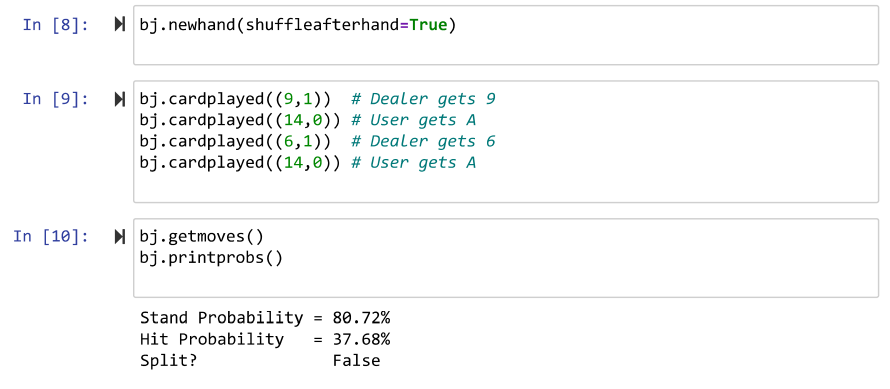


In cell 4, three cards are played, two for the dealer, and one for the user. In cell 5, getmoves is called to do the calculations, and printprobs is called to print them out, as shown.

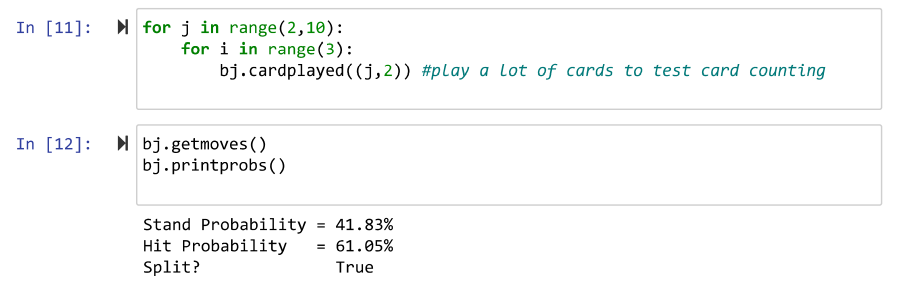




In cell 6, another card is played, and in cell 7, the probabilities are reprinted to show that they have changed as a result of a card being played.



Cells 8, 9, and 10 are similar, but the cards played demonstrate the program deciding whether or not the user should split.



Cells 11 and 12 continue on to show that by changing the cards in the deck, by playing cards to neither the dealer or user, the probabilities are changed so much that splitting becomes the correct choice, where it was not before.

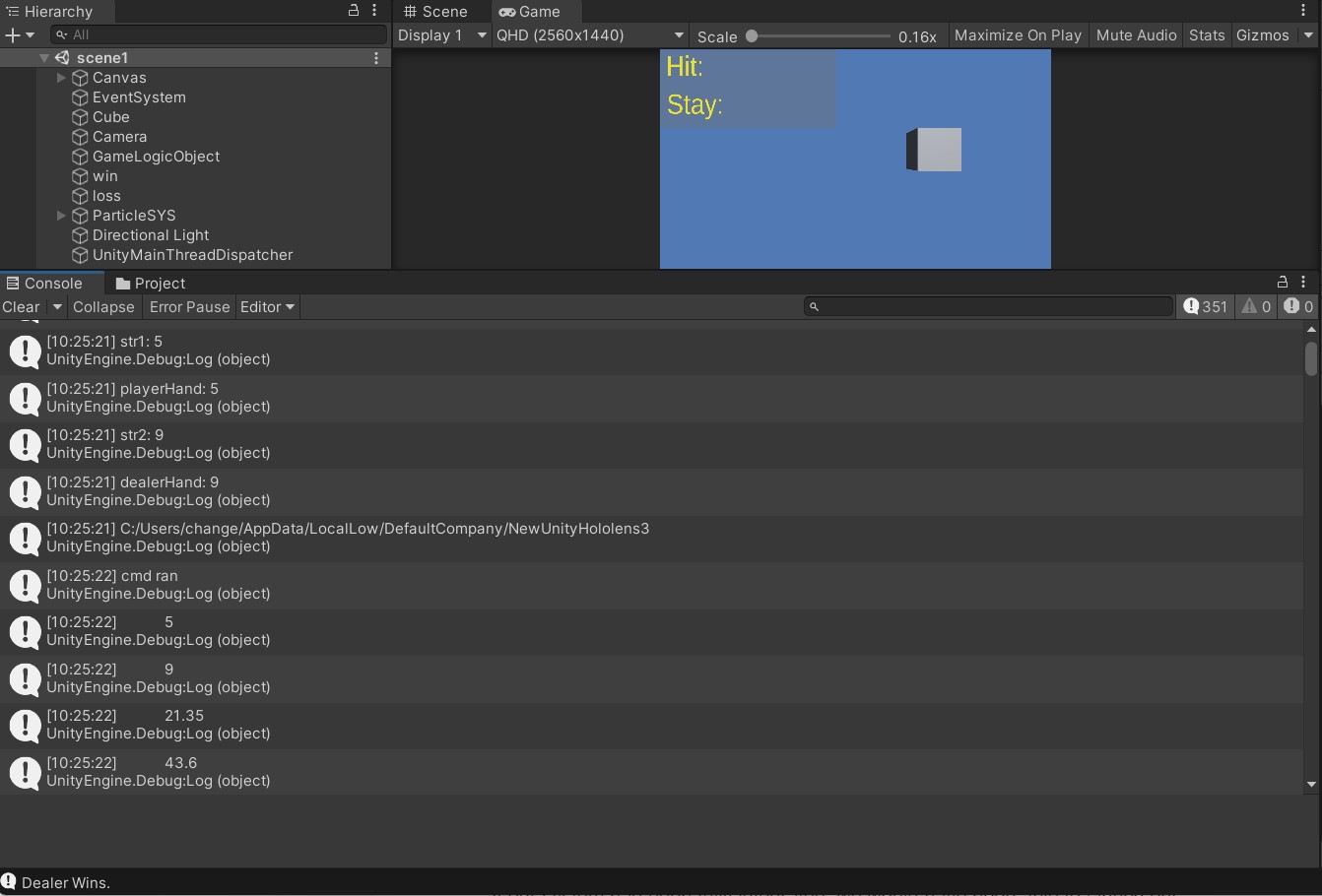
User Interface

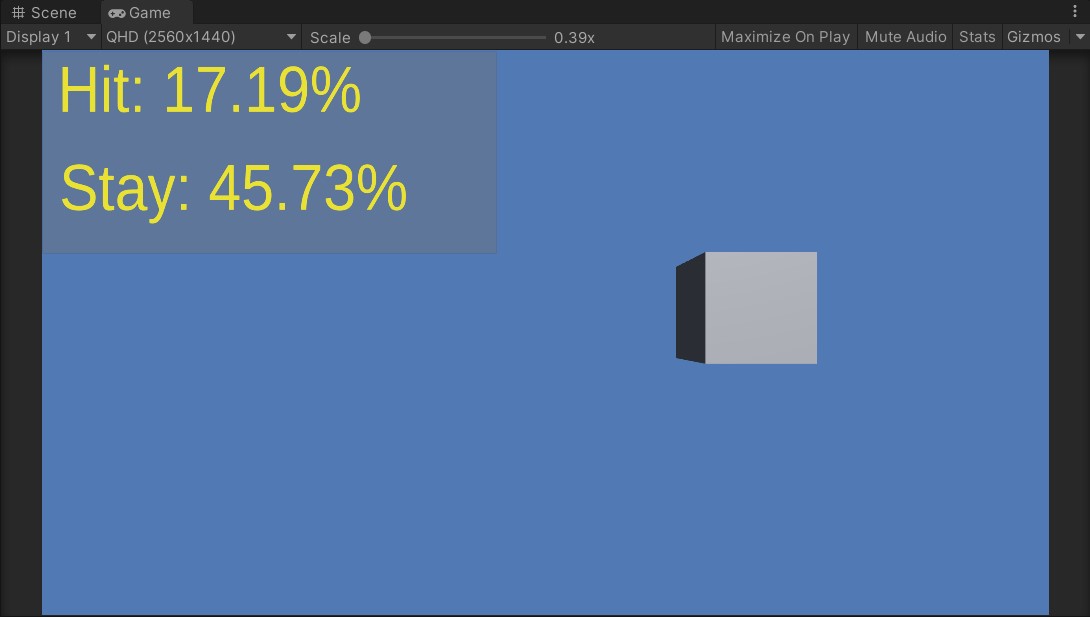
To test the UI, after building the UI solution in the “GameLogic.cs” C# script, the Unity Engine Debug Console was used to print out the probability values to determine that the UI is working correctly. Once it was built correctly, it would print out the values in the UI in yellow text, the “Stay” probability, and the “Hit” probability for the player. This testing using the debug console helped find a solution to a bug that was experienced in the design process. This bug was that it would not display the first probability on the UI text: “TextMeshPro - Text” object, even though it recognized the correct probability when printing it to the debug console. Ultimately the fix to the bug has been found after looking at a forum: "<https://forum.unity.com/threads/textmesh-pro-is-not-updating-when-changing-text-programatically.490125/page-2>”, it mentioned that running the needed code in the main thread of Unity Engine would fix it. To run the Unity Main thread, we used this Github repository which was linked to the forum found: “<https://github.com/PimDeWitte/UnityMainThreadDispatcher>”.

Overall, the debug console has been crucial for performing constant tests during the development of the UI for the blackjack assist app. It was constantly used to verify that variables and data structures are correct, as well as the logic for the app. For example, the global list: “argumentList” was iterated and printed to show it has the correct values. It has single character strings inside the list which represent the hands for the player and for the dealer. It has “p”, and “d” to denote dealer and player, and then the following card values are specified for player and dealer. This list is used to build the command line command with command line inputs when executing the blackjack.py python script with command prompt.

The following photo shows the debug console, it can be seen it prints the probabilities obtained from the “blackjack.py” file, and as well as single numbers which are from the global list, “argumentList” values. The probabilities seen in the photo are 21.35%, and 43.6%, they don’t have the percentage sign in the debug console, but they are the probabilities for that round of blackjack. Additionally, a notification from the debug console can be seen from the picture, it mentions who is the winner for that round of blackjack.

As a result, the testing performed on the user interface (UI) showed the correct behavior, and displayed the correct probability values obtained from the probability algorithm in the python blackjack script “blackjack.py”. The correct probabilities obtained from “blackjack.py” are reflected in the actual UI when starting the app, as well as on the debug console. Other tests performed, specifically executing the python script on command prompt with command line arguments, a text file was written to show the correct command line argument values has been received successfully from the “GameLogic.cs” C# script. Other successful tests performed include printing to the debug console who is the winner for that round of blackjack, the player, or the dealer. Testing to check that the global lists had the correct values from the set values for the player and dealer hands has been tested using the debug console, and is working correctly.





Special Effects

Blender is the free and open source 3D creation suite. It supports the entirety of the 3D pipeline—modeling, rigging, animation, simulation, rendering, compositing and motion tracking, even video editing and game creation. The coin was created manually using this software and otherwise would not be available as a special effect. The coin design was used following an online tutorial because Blender use was unknown.

For a raining coin effect in Unity the created 3D graphic had to become an object with components that included a particle system which would allow the new 3D import to have moving capabilities such as rotation and dispersion. Providing a Rigidbody component In Unity allowed these 3D coins to have physics qualities like the effect of gravity on falling objects which was a desired effect for this graphic.

Testing the raining coins effect included exporting the FBX file into Unity. This file contained the lighting, 3D object dimensions, and the customized gold color created on Blender. Once the 3D graphic was in Unity there were many adjustments and settings that were changed to view the coin design in a new platform such as Unity. The color did not transfer well from software to software therefore a new custom gold color was created in Unity and added to the 3D graphic. The dimensions of the coin also did not automatically stranger over and adjustments in Unity allowed the coin to become 3D once in Unity.

To have more control of this new particle system in Unity the team wanted to include a script that would allow the particle system to just be called and seen when necessary. Having too many objects in every frame of the game takes up a lot of memory and can cause delays or even GPU crashes. Aside from that the team just wanted the coins to be shown whenever the player won the game. Without implementing the particle system to the other parts of the game to include the game logic to include this control aspect a switch was created using the keyboard as an input for turning the switch ON and OFF. This will change as testing progresses for this system.

Unity is a powerful and diverse platform that gives creators and designers the power to be creative by providing many resources to allow special effects to take place like sound and custom moves. Sound effects can be made possible as imports while customizing the moves of an object in the game can be made possible through a script file. For the raining coins to have a sound effect associated with them there was a WAV file uploaded with a jackpot sound from a casino. Attaching the file as an audio clip in an audio source, a component of the object, allowed this effect to take place.

The combination of these was not terrible to test because they worked as they were intended to. Once implementing them into the game logic is where sound failure was experienced. There was sound so it wasn't an absolute failure it just no longer sounded like hitting a jackpot. This caused the elimination of this sound effect from the winning special effects and only the raining of coins stayed. The losing sound effect stayed because it worked correctly.

Full System

If our system had been fully integrated, we would have been able to launch our application on the Microsoft HoloLens. Once the application was launched, we would have been able to perform a trace from the Windows Device Portal. After the trace finished, an ETL file would have been available to download. An ETL file can be opened directly in the Windows Performance Analyzer, which is a tool that would have allowed us to visualize the traces as graphs and tables that show the system and application performance. The following data could have been collected from the WPA software:

1. SoC power: System-on-chip power usage
2. System power: System power usage
3. Frame rate: Frames per second, missed VBlanks per second, and consecutive missed VBlanks
4. GPU: GPU engine usage, percent of total available
5. CPU: percent of total available
6. Memory: Total, in use, committed, paged, and non-paged

This information would have been used to compare the application running on both the headset and on the host computer. By doing this our team would be able to prove that by utilizing a host computer would indeed increase the battery life and performance of the HoloLens.

**Discussion**

Object Detection

For Object Detection, Ideally, we would like to test the results in the real world with the object detection algorithm attached to the hololens & camera, but since we were unable to achieve that goal, we ended up testing each component separately. The results are of course with a camera at a 90° angle above the cards. If the tests were to be done on the lens, the angle would likely be around 135°, which produced a much lower mean average accuracy.

Probability Algorithm

As for the testing of the probability algorithm, the testing was not as rigorous as it perhaps should have been, but served the purpose of making sure the code would operate as expected with other parts of the project. In the test notebook, provided are a few examples of using the algorithm, during which a few of the features of the algorithm are showcased. This does seem to show that the algorithm is working as expected, as the results make sense given the inputs. However, If we were to do these tests with more depth, it would be reasonable to demonstrate each characteristic individually, then mix a few together, so as to test each piece more completely.

User Interface

The user interface (UI) test results were promising. After building the solution to the game logic using “GameLogic.cs” C# script, the script was able to successfully communicate with the “blackjack.py” python probability script. This was seen after the python script writes to a text file the probabilities obtained from “GameLogic.cs” via command line input arguments. In addition, it correctly displays who is the winner for that round of blackjack in the debug console.

Moreover, using the debug console, and researching online, bugs have been fixed. For example, the bug where it skipped printing the first probability to the UI, the Unity Main Thread Dispatcher github repository fixed that bug. Other bugs like when building the global list having incorrect string values have been fixed thanks to the testing performed using the debug console. One more bug that was fixed with testing using the debug console, was when the python script was not being executed after writing the built command string to the command prompt. It turns out that it had to be in the directory where “blackjack.py” was first, and then running “python blackjack.py” works for executing the python script. This meant the code was changed so that two commands were written to the command prompt for these purposes.

Full System

The following data would have been collected and analyzed if our project had been fully integrated:

**Data collected**

1. SoC power: System-on-chip power usage
2. System power: System power usage
3. Frame rate: Frames per second, missed VBlanks per second, and consecutive missed VBlanks
4. GPU: GPU engine usage, percent of total available
5. CPU: percent of total available
6. Memory: Total, in use, committed, paged, and non-paged

**Reason data this collected**

1. SoC power and System power - collected to view how much power is being used while each app is being used
2. Frame rate - collected to view how fast the communication is between each app and the hololens
3. GPU, CPU, Memory usage - collected to see how efficiency of the application

This information would have been used to compare the application running on both the headset and on the host computer. By doing this our team would be able to prove that by utilizing a host computer would indeed increase the battery life and performance of the HoloLens.

**Conclusions**

From our testing, we've learned many lessons about designing a product, mostly on the developmental side. Most of this involves finding bugs in code, as have been the bulk of our problems.

Testing for us was a means of seeing if our product was working at all, and as such, we did not rigorously test multiple aspects of our product. An improvement on this process would be a more thoughtful approach to testing, so that more erroneous errors could be caught earlier.

Testing for the actual Microsoft Hololens battery performance, and the latency of the app with respect to the Hololens was not implemented due to not being able to successfully integrate the hololens to the final product. Other than that, the testing for the written code has been performed and helped fix bugs and made sure the correct results have been obtained.

Bugs that have been fixed thanks to testing include the bug where it skipped the first probability when printing to the UI. Moreover, it helped fix a bug where the correct values for player and dealer’s hand was not being obtained in the global list “argumentList”, which was important for writing the correct command to the command prompt process to obtain the probabilities from the python probability algorithm “blackjack.py”. The bug where it wasn’t executing the python script via the command prompt was fixed.

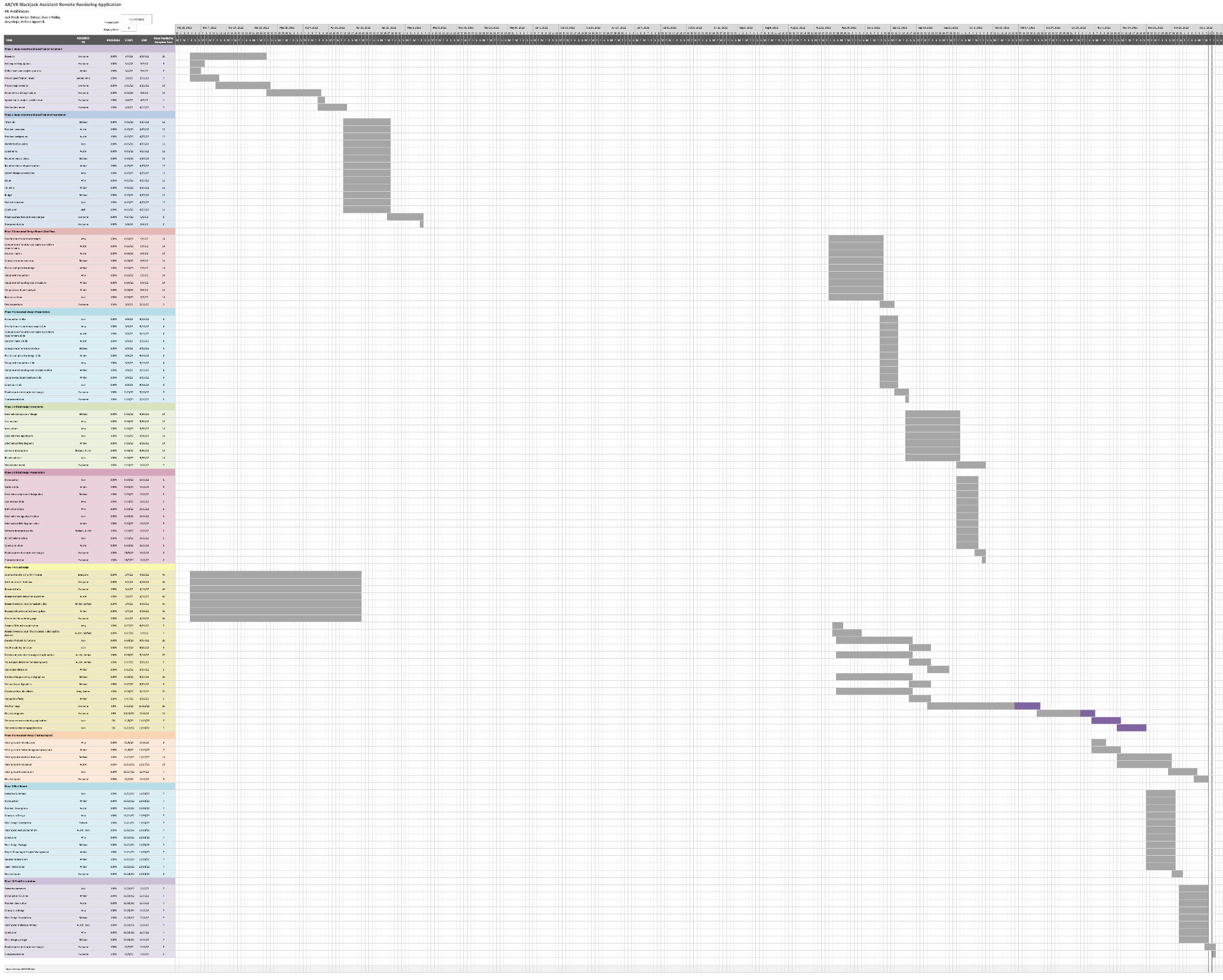
All-in-all I’d say there are a few things we could have done differently next time had we known all the knowledge gained from the project. We would: (1) take pictures at the ideal 135° angle, (2) train the dataset on the ideal 135° angle. This would provide much more accurate results since the basis of the data will be at the angle the hololens sees the project at. However, since we did not get this running on the lens, the angle of 90° worked well enough for our tests.

**Resources**

1. “Aka.ms.” [Online]. Available: https://aka.ms/CPUProfileforDevicePortal. [Accessed: 2022].
2. PimDeWitte, “Pimdewitte/UnityMainThreadDispatcher: A simple, thread-safe way of executing actions (such as Ui Manipulations) on the Unity Main Thread,” *GitHub*. [Online]. Available: https://github.com/PimDeWitte/UnityMainThreadDispatcher. [Accessed: Dec-2022].
3. “Textmesh Pro - Textmesh Pro is not updating when changing text programatically,” *Unity Forum*. [Online]. Available: https://forum.unity.com/threads/textmesh-pro-is-not-updating-when-changing-text-programatically.490125/page-2. [Accessed: 2022].
4. Windows-Driver-Content, “Windows performance analyzer step-by-step guide,” *Microsoft Learn*. [Online]. Available: https://docs.microsoft.com/en-us/windows-hardware/test/wpt/wpa-step-by-step-guide. [Accessed: 2022].

**Appendix**

Figure 1: Gantt Chart

****