Austin Hobbs, Jack Brock

**Discussion**

Broader conclusions can be drawn about the results, and potential reasons for these results can be discussed.

Issues in consistency, the testing procedure, accuracy and uncertainty should be discussed in detail.

No tests were performed. No lessons were learned

**Each person discuss what they put in results**

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.

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.

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.

Trace results

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.