# CS 288 Final Report

## Added Functionality

The additions that I implemented for my final project, diverged slightly from the previous deliverables that we were working on all semester. I created two parts to my ASL learner. The first part is the normal learning section that you can use to learn the first 10 digits of ASL and the second part is a game that I created to test the user's knowledge of the numbers, and hopefully use them in a way that is engaging and creative. When the user is first prompted to login, they may enter their username and click login. From here, the user is now prompted with two buttons from which they may choose. The first button is labeled "Learn" where the user will select the course of learning how to sign the first 10 digits of ASL. The user will go through all the steps that have been implemented in previous deliverables. If the user is finished learning, they may quit the program and they will be able to start over again. From here instead of choosing "Learn", the user may choose "Play." If the user will choose play, the program will initiate a revised version of the Deliverable 6 implementation, which was our most basic version while still utilizing the kNN Learner. When the user chooses play, they are agreeing to the fact that they know how to use and interact with the device, as well as having sufficient knowledge of the 10 digits. Since the user is an expert now, they decide to play. When the user clicks the play button, they will be prompted with only an open window of the wire frame hand. The user is now free to sign whichever gesture they please. However, their first gesture is going to be an argument for another program. I decided that I wanted to reuse my Final project from CS 206 (Evolutionary Robotics) and implement it in a way that the user was able to interact with the robot. In this project, I have designed a quadrupedal robot in the Physics Bullet Environment and give the robot and ANN. There are also 10 cylinders placed symmetrically, in a decagonal arrangement. The robot can "visualize" these cylinders and each cylinder has an id that will be associated with the 10 ASL digits. Currently to develop my ANNs, the program closes when there is a collision with an object and the run time and id are saved as a fitness. I restructured my fitness function to define each digit to have its own ANN by passing arguments of fastest run time and collision id of the cylinder and what I get is a set of weights that will move the robot to that specified object in the shortest time possible. Ideally what the user will be able to do is sign a digit, and that digit will then be translated to a python script that will invoke the simulation with the desired ANN. Thusly the user will have triggered the robot to attempt to seek out a cylinder in the simulation. Since the user is an expert at all 10 digits, they should be able to reach every single object in the environment. Currently the system is set up that the user will sign one digit, the robot will reach that object and the program will terminate and the user will need to sign another digit. This process will repeat until they are able to reach all of the objects.

## A Look Inside the Code

There were quite a few changes to my existing code that I needed to make, as well as creating a few new programs to implement them in a desirable way. The only changes that I have made to my written report 3 script (del10.py) was adding the functionality to choose which path to take, whether it be “learn” or “play.” When the user chooses the learning path, they continue on in the direction of the del10.py layout which is the same functionality that I have implemented all semester with the additional changes after evaluating user testing. I used a Tkinter library to create a functional GUI that the user can interact with, to input their username, press the login button, as well as the previously stated buttons. If the user presses the play button, the current script is terminated, and a new script is opened. From here the user will now be exposed to the deliverable 6 implementation of the kNN learner. I rewrote this script such that the user is exposed to the wire frame hand, and they have no prompts as to which digit to sign. This will allow the user to choose freely, and if the learner predicts their gesture to be a valid gesture, the script will then open up the asl_ANN.py script. This script is similar to the Project_Evolve_ANN.py that was created in CS 206. It will gather the respective ANN for the digit signed, and then open the Simulator, where the robot will attempt to find the corresponding object in the environment. When an object is found and touched, it will disappear from the environment, and will not load again until there are no more objects. This is all done in the c++ code for the RagdollDemo. The only changes that were made to this file, were the collision detection, fitness function handling, and creating and deleting the appropriate objects. The main script for the game “asl.py” uses threads to ensure synchronous processing. The main thread handles the leap motion device and all its processing. Determining whether a digit was signed and then passing that digit to the next thread. The second thread is where the data is saved to a .dat file to be read by the asl_ANN.py process. The third thread imports the module asl_ANN.py and calls the main function. The main function of this script gets the corresponding ANN and handles running the simulator and making sure the appropriate data is being stored in the appropriate places. When the simulator is finished running, when it either makes a collision or times out, the thread is destroyed and processing returns to asl.py. The main function of this process consists of a while loop that runs as long as the user allows it. Unfortunately, I was not yet able to handle what happens if all the objects disappear. Ultimately, I attempted to thread the processes, and try to keep the simulator running and switch/lock processing between the two programs, but I was unable to do so. Instead I just invoke the simulator and handler as child processes of asl.py and handle them all accordingly. It turned out to be rather difficult to handle a generic input from the user without the classifier, reading an open hand always as 5. Since the user must hover their hand over the device in order to advance, I created a wait function to allow the user time to sign their desired digit. I also created a list of integers (0-9) and remove a digit from the list when it is signed, therefore the user must sign a new digit on the next loop to continue. If not the program will wait for a user input.

## Future Development

If I had the opportunity to continue development and user testing, there would be a few aspects of this project that I would like to delve further into. The first is unrelated to the ASL per say, but more related to the game aspect. Ideally my intentions for the game were to allow the user to continuously control the robot with different gestures. Each new gestures would implement a new ANN which would influence the robot to start moving somewhere else. The most basic implementation of this would be creating an ANN for every derivative of a sequence of 10 digits that result in a collision. This would take a lot of processing, and a long time to do, but allowing the user to have access to the most evolved ANN for a sequence would be what I would like to implement. This way if the robot responds instantly and efficiently to the user input, the user can manipulate the robot in a way that allows it to continuously reach every object in one session. Instead the user must open 10 different sessions, which seems inefficient and may detract the user from continuing. With regards to the game, I would have liked to implement instructions that gave the user some insight into what exactly they will be doing, but due to a lack of time and resources, I was not able to do so. Instead the user must have prior knowledge of this portion of the game. With regards to user testing, I wish that I could pursue my ASL learner further. And allow the users to test the scaffolding system that I implemented. Unfortunately, I did not have users tests that lasted long enough to do this. If I had a 100 user sample to test with, I would certainly choose a lot of variety. I think the most important audience to test with, would be users who are already familiar with ASL. These users will have the best feedback, because they can interpret subtleties in the language, but may not understand the Leap Motion Device. I think it would be beneficial to understand how fluent ASL signers would behave with a familiar language in an unfamiliar form. I think that it is possible that the frustration or anxiety of trying to use a new device may leave the flustered or incompetent. However, I did not experience these issues with my user testing, therefore I cannot conclude that there would be much difference in negative emotions between fluent ASL and non-fluent ASL users. I think that another aspect of users to test would be the diversity of generations. From my personal experience, it is fairly evident that millennials and younger users are more apt to quickly learn a new technology as opposed to baby boomers and many older people. I think that this application would be most useful to the millennials therefore it would be really important and possibly useful to attempt user testing in possibly elementary and middle schools. At this age, the brain is still developing and it is easiest to learn new languages. If users are exposed to ASL at a young age, and especially during testing, they may be more capable of providing relevant feedback as to how users are learning rather than whether or not they can use the program. It would be interesting to explore the aspects of different learning styles. Do users respond well to hearing the digit in spoken word, if of course they are abled. If users could choose a language and then simultaneously listen and sign, this may have a greater effect than visually seeing the numbers. I believe that I would break the users up into 3 groups. I would have 33 users in each group and one outlier. The outlier will be someone who does not fit into the category of young/old or fluent in ASL. This user may be from the ages of 18-25 and possibly well-educated. They will provide a baseline for comparison. I would then break each group up into 11 users each and have a control group that users a basic program, a group that users a more advanced program, and a group that uses both. I think that testing the different learning aspects in this way may provide the most efficient results.