Daphne Yang

TP1: Project Proposal

Project Description [5 pts]:
The project is called Learn ASL/ASL Stack, and uses the Leap Motion to recognize letters of the ASL alphabet, test the user on their knowledge of the sign language alphabet, and implements a game that involves stacking blocks which are stopped by the user displaying a certain symbol.

Competitive Analysis [5 pts]:
Most of the project involving sign language recognition that I've seen online are solely used for recognizing the characters presented by the user to the Leap Motion. The learning game that I will be implementing will have multiple modes, including an option for the user to input gestures for words in ASL to expand the dictionary, a dictionary option for the user to make a sign and have the computer return it in English, and a game mode where the goal is stack blocks on top of one another and the blocks are stopped by making the sign for the letter (randomly selected) indicated on the block. The game will be an interactive and fun way for the user to actually be engaged and able to learn the ASL alphabet quickly.

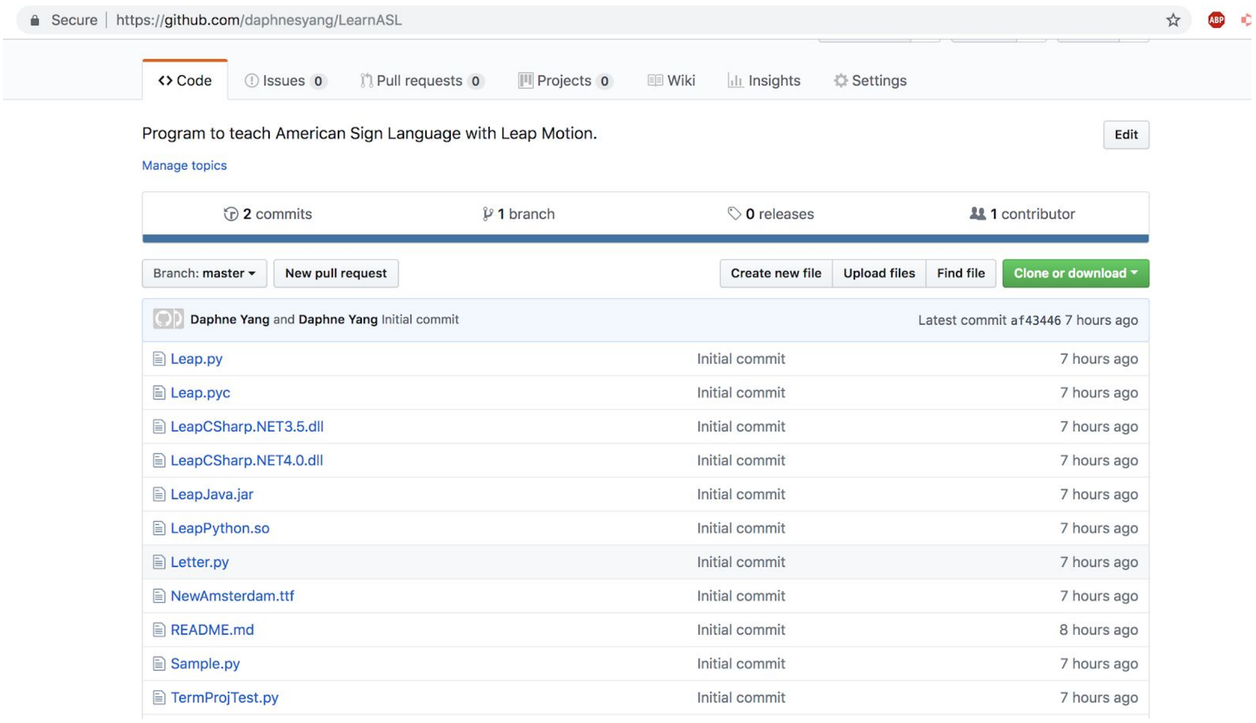
Structural Plan [5 pts]:
There will be five different functions for each of the modes and the start screen (startScreen(), helpScreen(), dictionaryScreen(), gameScreen(), and testScreen()). Additionally, there will be class objects for Buttons (storing their size, color, and text), Blocks (storing their width, length, letter, color), and Signs (storing the positions and directions of fingers relative to the hand). The classes will each have their own file.

Algorithmic Plan [5 pts]:
For creating the game with ASL words as well, I will have to implement a way that users can access settings and set motions and type in the phrase or word (which can then be included in the dictionary). The way I will approach this is use the gestures feature that is included with Leap Motion, and tracking the path of a hand for a set amount of time (one second). I will have to also take into account the rate of change of the position to be able to get recognition for moving letters or motioned words. With the 3D graphics, I will be able to achieve this by using geometry to calculate the way the blocks should look.

Timeline Plan [5 pts]:
I should have all the organization elements of the project, as well as the test, dictionary, and a simplistic version of the game mode using just the ASL alphabet done by TP2 to reach the MVP stage. Through Thanksgiving break, I will work on putting things into their appropriate classes, implementing an efficient way to recognize letters, and creating the basics of the game mode. After this has been done, I will add the option for the user to input words (which are using moving and might require two hands) for TP3.

Version Control Plan [3 pts]:

I am using a GitHub repository to keep my code backed up online. Through terminal, I can continuously add in the updated versions of my code with a description of what I changed in each version.


Secure | https://github.com/daphnesyang/LearnASL

Program to teach American Sign Language with Leap Motion.

2 commits | 1 branch | 0 releases | 1 contributor

Daphne Yang and Daphne Yang Initial commit — Latest commit af43446 7 hours ago

Leap.py	Initial commit	7 hours ago
Leap.pyc	Initial commit	7 hours ago
LeapCSharp.NET3.5.dll	Initial commit	7 hours ago
LeapCSharp.NET4.0.dll	Initial commit	7 hours ago
LeapJava.jar	Initial commit	7 hours ago
LeapPython.so	Initial commit	7 hours ago
Letter.py	Initial commit	7 hours ago
NewAmsterdam.ttf	Initial commit	7 hours ago
README.md	Initial commit	8 hours ago
Sample.py	Initial commit	7 hours ago
TermProjTest.py	Initial commit	7 hours ago

Module List [2 pts]:
Modules: Pygame
Hardware: Leap Motion

TP2 Update:
I added a inputScreen() for users to input data for recognition of new signs.

I switched the way I would store the signs and the data for the signs. Instead of using a Letter object, I simply have a dictionary with everything that has been input into the system (there is an option to input new signs) with the name of the sign as the key and a dictionary of the coordinates for position and direction as the value for that key. I am storing this in a text file in the same directory (this file is read then combined with a dictionary of the new signs from opening the program again), so that even after closing and reopening the program old signs that were inputted will be stored. If a user inputs something with the same name, it will update and override the old data stored.

TP3 Update:
I added in a screen with instructions that appears before the game begins. I also modified the way I am checking if a sign is a match; I take the difference between each coordinate of the sign inputted into the dictionary and square that value, then add that to a variable called

sumOfErrors. This variable accumulates all the error from the data points and if it is below a certain threshold, determines that the sign is a match. Additionally, I calculated the distance between each adjacent fingertip and added that as another value in the sign data to help with accuracy. Lastly, I added in more room for error by altering the inputSigns feature, where the user is prompted to display the sign and enter it in three times, so that there are actually three versions of the data for each sign. Therefore, the checking loop in dictionary, game, and test mode must loop through each version of each sign.

I also added the test mode, which diplays a