!!!! DISCLAIMER (Results of our project are included in video files with our final zip file)!!!!

Our team decided to create an interactive language tool to aid in learning a new language. We decided to use Spanish as our basis due to the similarities that many of the characters share with the English language.

Data Structures

For our data structure we decided to use a binary search tree. We needed the traversal of our input data to be fast; we decided against a hash table because the two languages (Spanish and English) exist within two text files. By have the English translations placed into a separate text file, different languages can be read into the program easily. We also need the words to be in alphabetical order when printing the entire repository of nodes and their encoded data.

Methodology

We initial envisioned our project to an app that existed on the App Store. We wanted to create something lightweight with multi-functionality all integrated into a streamlined user-interface. Our project includes a direct translation tool that allow the user to input a specific word to be translated; the user may want to use this function if the word contains similar characters to their main language.

To directly aid in learning a new language, we implemented a matching game that contains various words and their direct translation. This function focuses on the physical interactions to help cement a word within the user's vocabulary.

The last function allows the user to take a photo of a word to be translated. This function is most useful when the user does not have the capabilities to type in the characters; this situation may occur when the language is vastly different than the user's main language (i.e. Korean to English).

!!!! DISCLAIMER (Results of our project are included in video files with our final zip file)!!!!

Project Details

OpenCV was utilized to achieve the optical character recognition. OpenCV allowed us the process different image files to identify different characters within an image. The KNN algorithm was loaded with pre-constructed training data that allowed us the determine the different ASCII values imbedded within an image. A string is then constructed and passed into our Binary search tree to identify a possible translation.

Void trainProgram(); trains KNN algorithm

std::string getString(); Optical Char Recognition & string construction

The UI was built within the QT Creator and imported into Visual Studio. The matching game utilizes a randomization function to pull random nodes and their associated data from the binary search tree. The data is then passed onto a separate UI class which prompts the user to play the matching game.

std::string* mainwindow::on_pushButton_clicked(); calls the matching game

The UI also implements a direct translation function to it. The user can find translations for words in Spanish, or the user can translate English words into Spanish. The is accomplished with two different search functions:

void Search(string word); searches for string in Spanish (or another language)

void SearchEnglish (string word); searches for string in English