Boston University
Electrical & Computer Engineering
EC464 Senior Design Project II

Final Report

Speech Interactive Therapy Application

Submitted to

Andrey Vyshedskiy
(617) 817-1916
vysha@bu.edu

ImagiRation

Team 13 SITA

Team Members
Jenni Norell: janorell@bu.edu
Diana DeLeon: ddel@bu.edu
Zhe Deng: thezhe@bu.edu
Swathi Jaisankar: swathij@bu.edu
Cindhuja Sramasamy: cindhu25@bu.edu

Submitted: 10 April 2020

Table of Contents

1.0 Executive Summary

Our team is determined to make a speech interactive therapy application to help young, nonverbal children, typically between the ages of 2-5, with autism spectrum disorder (ASD). The game begins with a level scene. This scene contains a series of videos of a speech pathologist pronouncing phonemes. As the child's skills progress, the game will play more complex and advanced levels, potentially, in the future, up to sentence-level exercises. Parents or guardians will also have the choice to record their own video levels. For every video, the child will receive a visual reward based on an accuracy score. There are three distinct rewards scenes, each dedicated to a certain range of accuracy (low, average, high), which are interactive and colorful for the children. The voice detection algorithm controls the rewards logic. First, our client's own algorithm detects a child's voice. Then the envelope of the child voice is measured against a model envelope. Furthermore, additional metrics such as change in frequency are used to increase the result's accuracy. The goal is not to detect specific words, but rather encourage *vocalization*--the ability to produce and match sounds. Additionally, there is a recording setting, in which a parent or guardian can record their own videos for the child to practice with. All of SITA's components will be built in Unity 3D and deployed to iOS and Android. Our goal is to provide a very simple, yet dynamic therapy environment that gradually builds necessary speech skills.

2.0 Introduction

Our Project is on a Speech Therapy App. Our team is developing a game that will assist in the speech therapy of kids with autism spectrum disorder. We are developing this application for non-verbal children between the ages of 2 to 5. It is a dynamic speech therapy game for language cognition and speech training.

Our App is called the SITA - Speech Interactive Therapy Application. It is mainly designed to provide an early therapy action to the ASD kids in the absence of a therapist in a game-like environment.

Working of SITA:

➔ Opens with the lock screen in which the parent answers a math question to ensure it is not the child logging in.
➔ Goes to the levels/phonemes page where the difficulty level/phoneme type for the game can be chosen by the parent for their kids.
➔ From the levels page control can be transferred to the settings page where the parent can track the progress of their child and perform additional settings.
➔ After the level is chosen, the app will switch to the kids mode and a video of a speech pathologist pronouncing a word will be played to the child and the child will be asked to repeat it.
➔ Once the child repeats the word, the kid will be rewarded with some visual 2D effects with an interactive screen where as the kid touches the screen the particle effects will pop up.

Design of SITA:

❏ Developed in Unity 3D as proposed by the client.
❏ The User Interface is colorful and interactive in an appealing way to the kids.
❏ Created with only 2D effects to avoid longer loading time and lagging when in kids mode to keep them engaged.
❏ Access control provided to parents and kids only for privacy reasons and everything is operated offline.
❏ Database consists of a speech pathologist recorded model word videos organised based on phonemes and difficulty level.
❏ Customization for kids based on their improvement in speech in terms of difficulty.

Goals of SITA:

- To provide speech therapy access to all ASD kids at minimum costs.
- Make the speech training more fun filled in a gaming atmosphere.
- Ensure the training session is accessible to kids at their convenient times and places.

- Maintains privacy in terms of kids' voice, training and progress.
- Helps keep a regular check on their improvement and customize the training sessions.
- Developed on Unity 3D which can be made both Android and iOS compatible.

3.0 System Description

What we will provide as a solution is an application built in Unity3D. This application has various stages, or scenes. Each scene has its own functionality and connects to at least one other scene. Below is a flowchart of how these scenes interact with each other. The interrupt lock scene is the first page seen when opening the application. This is a simple math problem to verify that an adult is trying to access the application rather than a child, and it has two states. Its first state is its initial state, in which there are unlimited tries to access the application. The second state is when we enter the interrupt lock after exiting the video scene. From here, the user has three attempts to solve a math problem, and if they get it incorrect each time they are locked on the video scene for 60 seconds.

The next scene is the LevelMenu scene. This scene is, after getting passed the interrupt lock, the main menu. There are various buttons that connect to the video level scene and one button that allows us to navigate into the settings page (the parent setup scene). The Parent Setup scene currently has four panels, each of which will contain relevant information and a button leading back to the level scene. The video scene currently connects to a video and when clicked, plays it. After the video is played, there is a pause in which the application is running a voice recognition algorithm. Based on the child's reply, The voice algorithm recognises it and finds the accuracy percentage for the reply. Based on the accuracy percentage the rewards scene is loaded. If the accuracy is below 10% then the Almost there reward scene is loaded. Else if the accuracy is below 20% then the Great Job reward scene is loaded. Else, the Prefect Win reward scene is loaded.

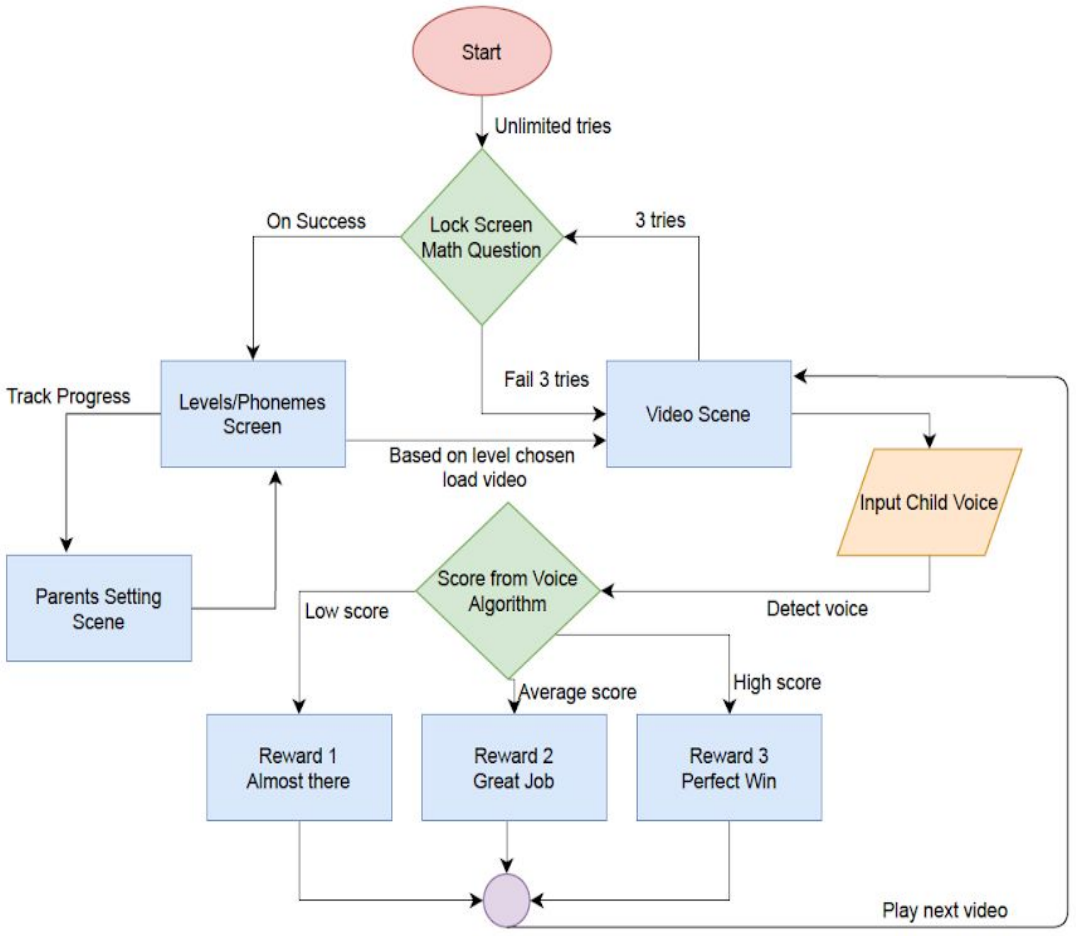
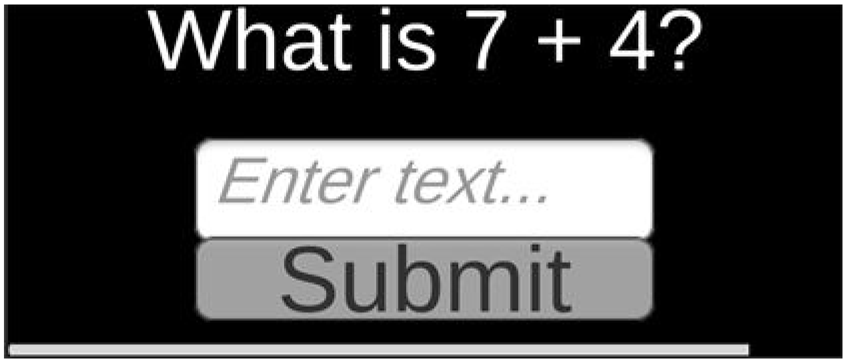
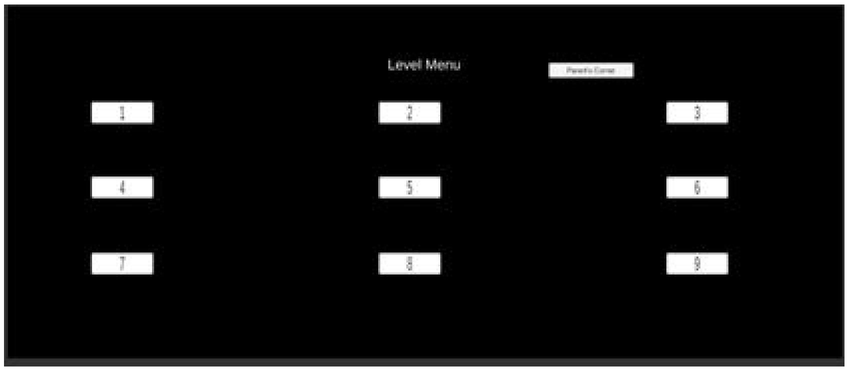


Figure 1. Block Diagram of Final Version

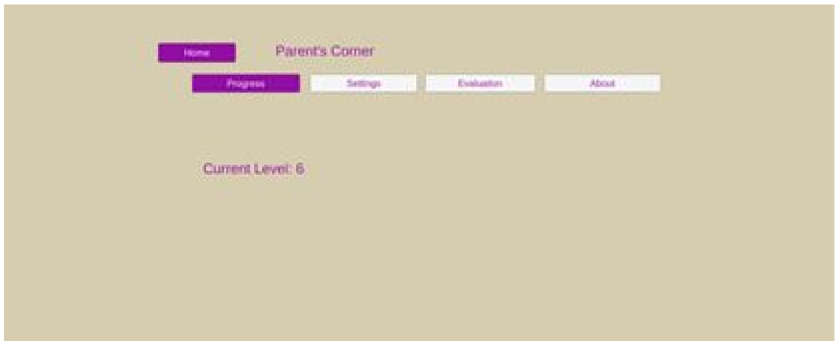
Lock Screen: Interrupt Lock, Answer the Math question within 10sec to ensure its the parent and enter the game.



Levels Screen: Choose the level to train your child according to the progress. Click on the Parents' Corner button to set the difficulty level and track the progress.

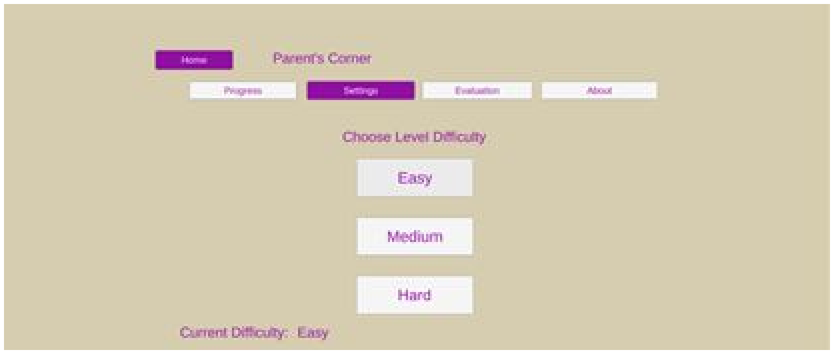


Parent's Corner: Progress - This displays the current level in which the kid is and shows the progress made until now.



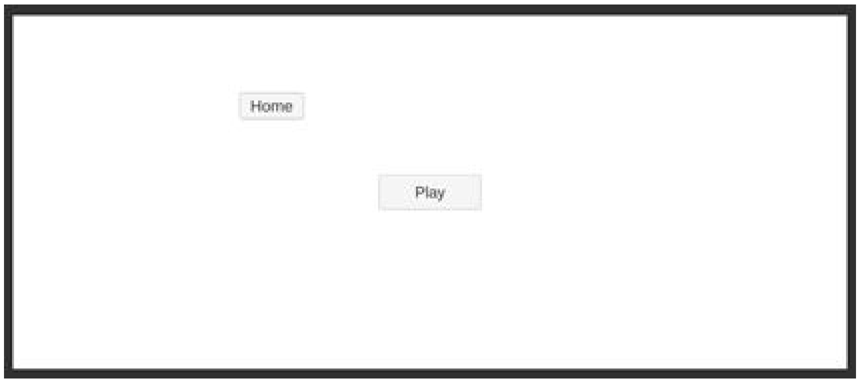
Current Level: 6

Parents' Corner: Settings - Choose the level of difficulty you would like to set for the kid as Easy / Medium / Hard.

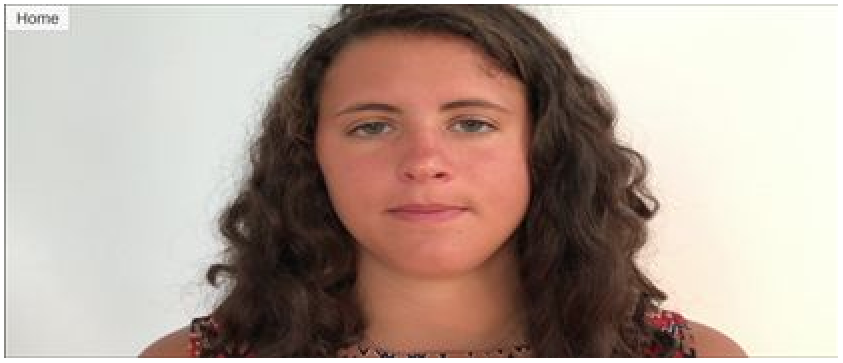


Current Difficulty: Easy

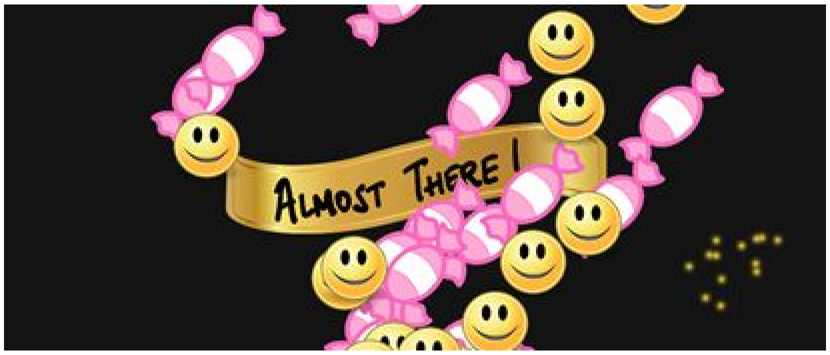
Video Scene: Click on the Play button to start the video of the speech trainer pronouncing the phoneme, word or sentence based on the difficulty set to.



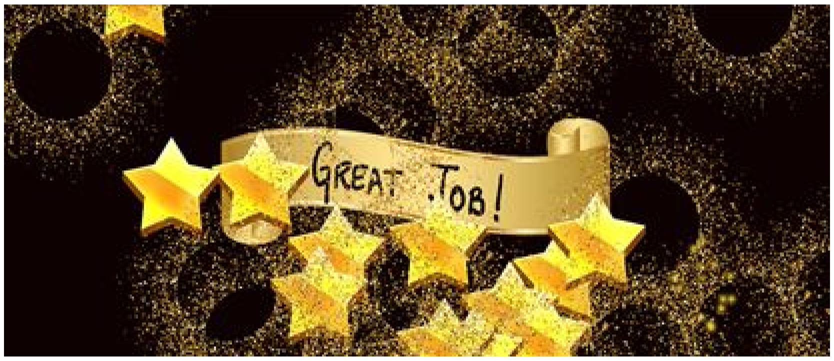
Video Scene: The Speech trainer pronounces the phoneme, word or sentence in a clear tone and waits for the reply of the child. Then the child's voice is recognised and scored based on the accuracy achieved in this try.



Reward Scene1: If the accuracy level of the child's voice is <= 10% then the "Almost There" reward with 2D effects is loaded. It has touch controls which produce golden particles wherever the child touches on the screen. The next video is loaded in 10 secs.



Reward Scene2: If the accuracy level of the child's voice is <= 20% and > 10% then the "Great Job" reward with 2D effects is loaded. It has touch controls which produce golden particles wherever the child touches on the screen. The next video is loaded in 10 secs.



Reward Scene3: If the accuracy level of the child's voice is > 20% then the "Perfect Win" reward with 2D effects is loaded. It has touch controls which produce golden particles wherever the child touches on the screen. The next video is loaded in 10 secs.

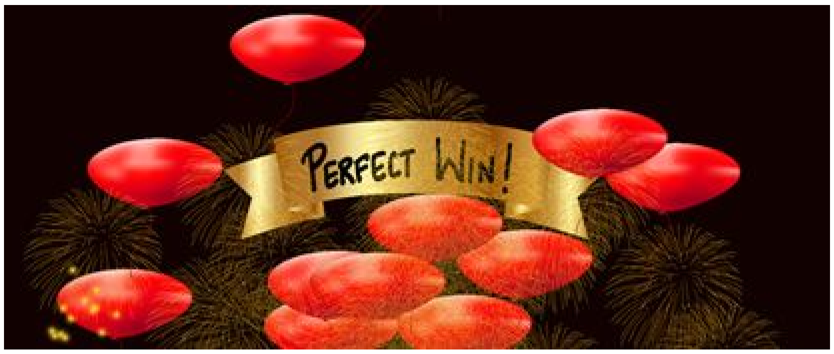


Figure 2. Screenshots from Current Prototype

4.0 Second Semester Progress

Zhe Deng

- Processor Interface (processors.cs) - C# Interface for audio buffer processing
    - **Envelope** - Captures RMS envelope history and envelope-aligned sample history in circular buffers.
    - **Spectrogram** - Stores Power Spectral Density history in circular buffers. For each PSD sample there is a sample for:
        - 1. high frequency energy (sum of upper half of each PSD)
        - 2. low frequency energy
        - 3. a pseudo-expected value used in frequency analysis ($\sum_{0}^{n} i * PSD[i]$ where n is length of PSD).
- Speech Class (Speech.cs) - A static class with general-purpose and experimental voice processing functions (only main functions described)
    - **public static void trim** - Trims an Envelope object using RMS envelope history either from the end or start of the word. All data before/after an envelope threshold ([0,1]) is hit--minus some extra samples for safety--are trimmed.
    - **public static void normalizeMinMax** - normalizes an analytical signal and removes any dc signal
    - **public static float[] errorNoRamp** - Takes model word and mic input envelope data and returns an array of sliding error results where the model word slides across inside the bounds of the mic input (model word length < mic input length). Overshoot error is multiplied by the 'overshoot' parameter.
    - **public static float[] DET -** Syllable detection based on "Differential Envelope Technology." Given an **Envelope,** filters through a fast and slow step response IIR filter and calculates the difference, thus extracting transient information.
- Envelope Matching
    - **Envelopes** are normalized with **normalizeMinMax**
    - The **error** function is called on the **Envelope** objects of the mic and model word. The MIN of the output array is equal to (100-accuracy)/100.
- Frequency Score
    - Done on highly trimmed (no trailing/leading zeros) model word and mic input data
    - The ARGMAX of the **Spectrogram**'s pseudo-expected value is calculated and normalized to the array's length (output range = [0,1]).
    - Sum of high and low frequency energy for both halves of the word and mic input are also captured, but not currently used.
- Syllable Detection
    - **DET** is called on mic and model word envelopes. The number of peaks past a threshold is the number of syllables detected.
- Video Player (ModifiedVideoPlayer.cs)

  - Refactored to act as driver code for voice detection.
  - Loads videos asynchronously and gets samples information for processing.
  - Calls **trim** on model word so that there are some trailing zeros/ low level noise (forces error metric to account for model word ending)
  - Sets mic input circular buffer as 0.5 seconds longer than model word
  - Blocks mic input evaluation when processing model word
- Mic Capture (Mic.cs)
  - Updates mic's **Envelope** and **Spectrogram** per buffer
  - Evaluates mic input when client's algorithm is triggered
- Debugging Features (ChildVoiceRecognition project, Speech.cs, Plotter.cs)
  - Different trims of mic and model word are captured and stored as wav files in Resources/WAV
  - All **Envelope, Spectrogram,** and **error** metrics are plotted on screen

Cindhuja SRamasamy:

- Rewards Scene
  - Scene 1: "Almost There" - modified textures of smilies and candies and added them as 2D particle effects.
  - Scene 2: "Great Job" - modified textures of glitter rings and golden stars and added them as 2D particle effects.
  - Scene 3: "Perfect Win" - modified textures of red balloons and golden firecrackers and added them as 2D particle effects.
- Touch Controls
  - Added golden particles that appear over the scenes.
  - Any touch screen device enables to move them throughout the screen.
  - The script collects the indexes of all the coordinate points wherever the child touches the screen and moves the 2D particle effects to those positions making it interactive.
- Scripting
  - Wrote the Touch control script - "Movebytouch."
  - Update() method - Inputs the coordinates through the gettouch method. Creates a vector with these positions by converting them to the world coordinates. Uses the transform function to move the particle effect to the respective location on the screen.
- Audio for rewards
  - Created sound files for rewards scenes to play in the background.
  - Looped the mp3 file component in the audio source to play until the video scene is loaded.
- Video Recorder
  - Swathi and I started working on the video recorder to capture the video of the Parent and store it and play it to the kid along with other pre-loaded videos. We got to the webcam displaying everything in front of the screen.
  - We began scripting to store the video recorded and play it back but due to the time restrictions and switching to remote classes we were unable to finish it.

  - As a result we have removed those components from the final deliverable since it is not finished.

Diana DeLeon:

- Video Scene
  - Implemented Load/Save method to videos to reduce memory usage using the Resources Class in Unity
  - Organized model words by difficulty using “Resources” file (“Database”)
    - Easy: Words with one syllable, Medium: Words with two or more syllables , Hard: Sentences
  - Edited ModifiedVideoPlayer.cs to choose a video depending on Level Difficulty (Easy, Medium and Hard) and Current Level (1, 2, etc.)
    - Used PlayPref’s to access the values that Jenni stored
- Settings Scene
  - Created the UI for the Settings scene by keeping the Parent_Setup scene’s background and upper buttons
    - Added 3 Buttons (Easy, Medium and Hard)
    - Added all the necessary text to the scene
    - Made color changes to the settings and progress buttons to indicate which scene the user is currently in
- Integration
  - Connected the Settings scene to the Parent_Setup and vice versa by creating click() event handlers for the progress button and the settings button

Jennifer Norell:

- Integration
  - Connected specific levels to specific video depending on which was selected
  - Minor Changes to Parent’s Corner
- Tracking Progress
  - Created two scripts that communicate with each other that keep track of the progress made in regards to levels.
  - One script is dedicated to the buttons, and passes the information along when a button is clicked (which level this was and if it is a further level than the one previously completed).
  - The second script is dedicated to receiving this data and displaying it on the Progress page of the Parent’s corner
- Level Difficulty Button on the Settings scene - Worked on this with Diana
  - In the Parent’s Corner / Settings page there is now a selection button in which the user can determine if they want the levels to be phonemes or words. Can later be adapted to include more variety (full sentences, parent recorded videos, etc)
  - Created two scripts to update the videos

        - One script is very similar to the buttons script for tracking progress, which passes the information about which button was clicked to the settings script
        - The settings script is used to receive the data and to display it on the settings page. It also stores which level of difficulty is chosen, which Diana used to select which videos will be displayed.

Swathi Jaisankar:

- Parents' Corner
    - Researched and figured out which aspects to incorporate into the Parents' Corner to make it best suitable for the child and the parent
    - Made it accessible after the math question is answered correctly
    - Created a canvas, scene manager, and event system
    - Canvas includes multiple buttons, panels, and text boxes
- Parent_Setup Scene and Integration
    - Buttons
        - Home Button: Connected it to the Home Page
        - Setup the UI for following buttons: Home Button, Progress Button, Settings Button, Evaluation and About
- Video Recorder
    - Took the lead on this portion which was meant for parents to be able to record videos of themselves saying specific words or sentences that they wished for their kid to repeat.
        - Wrote the CameraScript which allowed for easy access to the Webcam but was not able to connect the recording aspect in order for the game to store the recorded videos.
        - Integrated the CameraScript into Recorder Scene - plane object for a clear landscape view
    - Cindhu and I were not able to complete and implement this portion due to time restrictions and remote classes.

5.0 Technical Plan

Task 1: Video Player
12/16-12/30
The video recordings should be played for each level. Lead: Diana

Task 2: Rewards
12/15-1/15
After the video plays and the child repeats it, the rewards should show up. Lead: Cindhu

Task 3: Make Level Menu Phoneme-Oriented
12/23-12/30
Going off of what our client suggested, the levels will have to be sorted by phonemes rather than by numbers. The levels will increase in difficulty by phonemes. Lead: Jenni

Task 4: Sync Level Menu with Videos/Database
1/15-1/30
The level menu will have to be synced with all the videos which will have to be called from the database. The scenes will all have to flow into one another at real time and not be delayed especially as that could cause the child to get bored and lose focus on the game. Lead: Diana

Task 5: Syllable and Vowel Detection
Stretch Goal
Adding another two dimensions to the accuracy score helps distinguish words with similar envelopes. A more refined syllable detection algorithm as well as a vowel detection algorithm would make the voice detection more robust. Lead: Zhe

Task 6: Effects Assets and Sync with Sound
1/25-2/15
Particles will be reworked so that they leverage more components from the Unity particle system (i.e. more control on particle diffusion and scaling as well as better optimization). Audio assets will sync with triggers and then performance testing will be done to determine an appropriate number of triggers which does not negatively affect frame rate. Lead: Cindhu

Task 7: Progress Panel: Current Level
1/30-2/15
The Progress Panel will showcase exactly how far the child is getting by being connected to the Levels Menu so parents can easily keep track of their child’s progress. The Current Level will be displayed here. Lead: Jenni

Task 8: Parents Corner Sync Progress
2/2-3/15
For the parent's corner, the Settings Page is for parents to be able to set and customize the application in the best way that suits their child. This includes being able to pick the level of difficulty based on the child's progress. Lead: Diana, Jenni

Task 9: Parents Corner Sync Progress
Stretch Goal
For the Parents Corner, the Evaluation Page is going to be a form to help SITA with its ongoing research to develop the most effective interventions for children with Autism. The About Page will give parents detailed insights into the purpose of the app and gives them step by step on how to navigate through it. These aspects must be fully functional and integrated. Lead: Swathi

Task 10: Optimize and Test for Mobile
2/15-2/25
Our existing project will have to be tested to be the perfect fit for a mobile device. It should be working on both iOS and Android platforms so we will have to optimize it accordingly for both. Lead: Swathi, Jenni

Task 11: Test with ASD/nonverbal Children
Stretch Goal
We will be testing our application on actual children with ASD / nonverbal kids in order to test if it actually works. We will be able to measure the success of our application by keeping track of the progress that the child is making from the start of the use of the app to the end. Lead: Swathi, Cindhu

6.0 Budget Estimate

Our current budget estimate is zero dollars. Our Speech Interactive Therapy Application is a software development based project. We do not have any hardware components, and we do not have to purchase any software, platforms or tools in order to complete our project. All of the tools we are using to develop our application are free.

7.0 Attachments

7.1 Appendix 1 – Engineering Requirements

Team # 13 Team Name: Speech Therapy

Project Name: Speech Interactive Therapy Application (SITA)

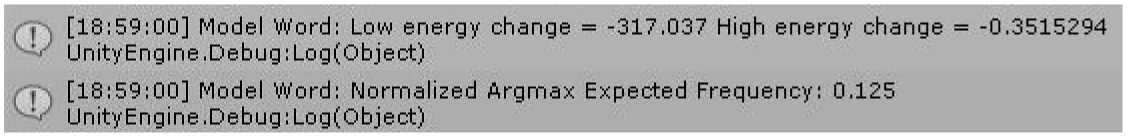
Requirement	Details	Status
Target Platforms (Task 7, 5.0)	iOS, Android	Android Compatibility Tested, Needs iOS Testing
Offline	Must not require internet and store parent recorded videos locally	Done
Tamper-proofing (SW diagram, 3.0)	Parent settings require simple math questions to access; video scene locks after 3 unlock tries	Done (see SW diagram, 3.0)
FPS (Task 7, 5.0)	App must run consistently at a minimum of 60 frames per second, without noticeable stuttering	Build is stable, but frame drops are still noticeable on mobile
Envelope Matching (Zhe, 4.0)	The envelopes must have a sample rate of 50 Hz. Accuracy scores from envelope matching must distinguish correct and incorrect input. Flat, high mic envelopes must score low.	Done and improved from sem 1, Best input scores >20% Ok input scores >10% Incorrect/Tampering scores ~0% (See Appendix 2)
Aspect Ratio (Task 7, 5.0)	Must support common mobile aspect ratios with minimal rescaling	Done, but could be improved
Touch Compatible	App must be compatible with mobile keyboards and support gestures	Done
Frequency Score (Zhe, 4.0)	See 4.0. Correct mic inputs have scores within 0.2 of model word score.	Added and passes tests, but not reviewed yet by client (See Appendix 2)

Syllable Detection (Zhe, 4.0)	See 4.0.	DET may need preprocessing for more accuracy. Results are dependent on the dynamic range of data. Detected syllables are often 1 or 2 off of ground truth.

7.2 Appendix 2 – Voice Algorithm Visualization

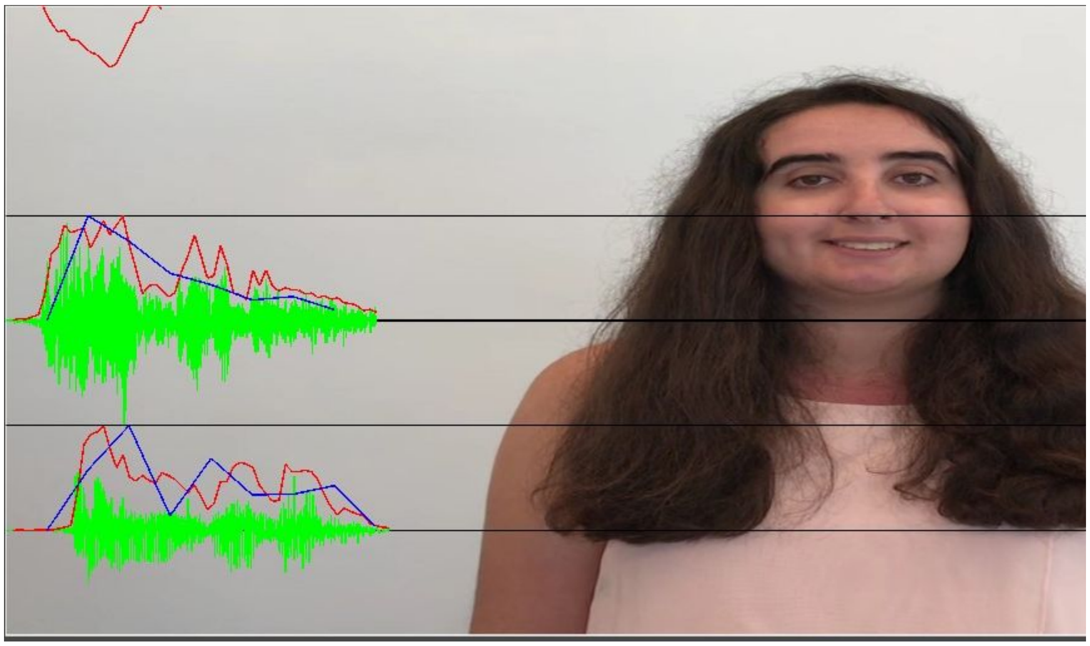
How to interpret the graphs:
Below are the debug graphs for input that cause Best (>20%), Ok (<=20, >10%), and Incorrect (<=10%) accuracy scores. There are three levels to the output plot. The first level (bottom) is the mic input, its RMS envelope samples (red), and pseudo-expected Spectrogram value samples (blue, normalized with normalizeMinMax, [see 4.0, Zhe]). The second level (middle) is the same, but for the model word. The third level is a plot of the error metric with an 'overshoot' value of 1.5 (see errorNoRamp, 4.0, Zhe).

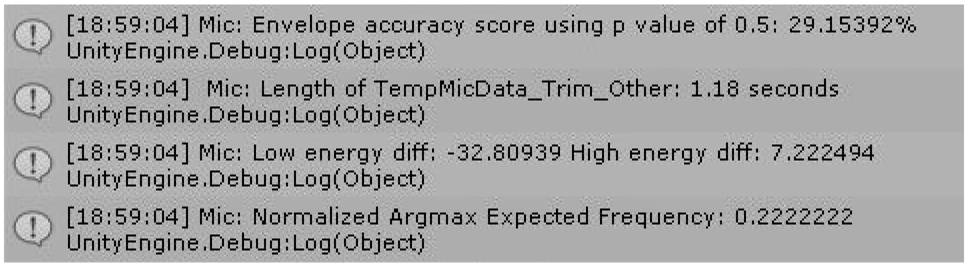
Test Environment:
The graphs are generated in the "ChildVoiceRecognition" debug project. The algorithm works well with both words and sentences. Here, the model sentence is "How was your day?" Below are the frequency metrics of the model word printed in console. "Normalized Argmax Expected Frequency" means the model word's "frequency score" is 0.125 and the peak of its pseudo-expected Spectrogram value samples should be around ⅛ of the way through the word.



```
[18:59:00] Model Word: Low energy change = -317.037 High energy change = -0.3515294
UnityEngine.Debug:Log(Object)
[18:59:00] Model Word: Normalized Argmax Expected Frequency: 0.125
UnityEngine.Debug:Log(Object)
```

Best:

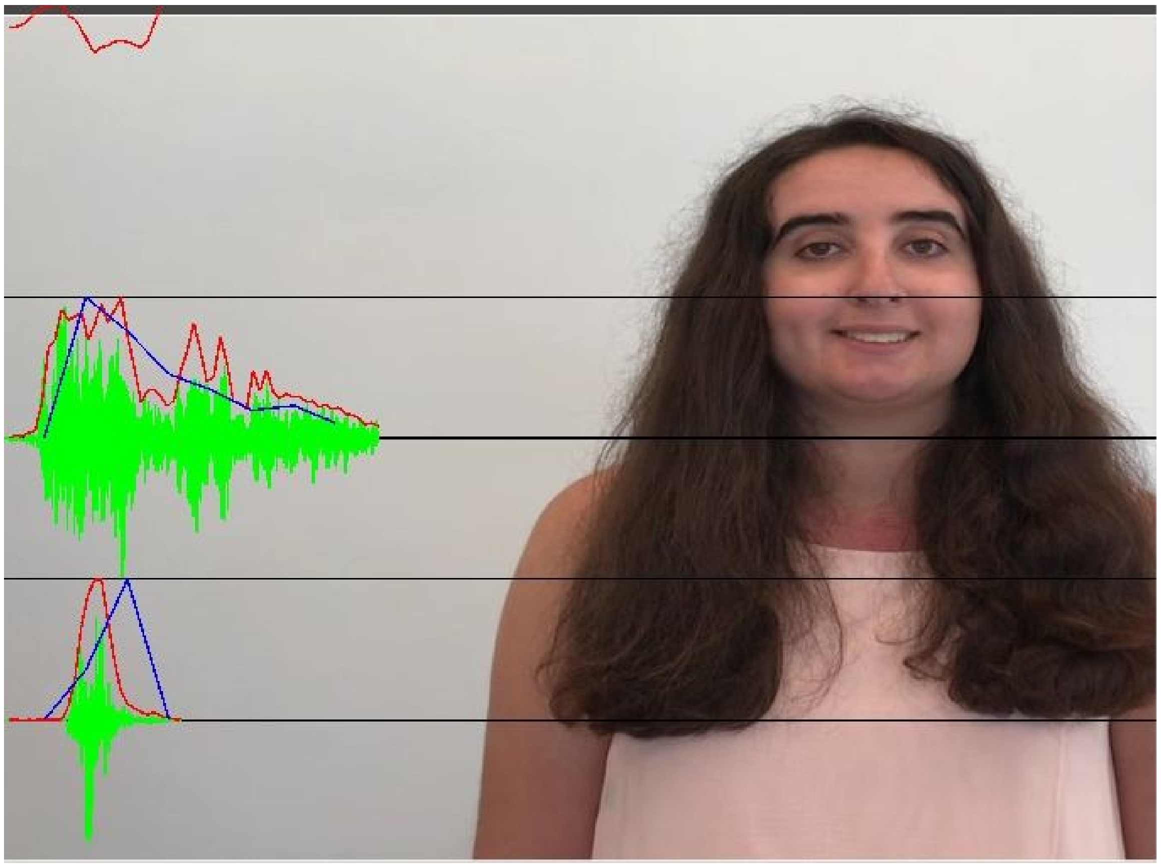


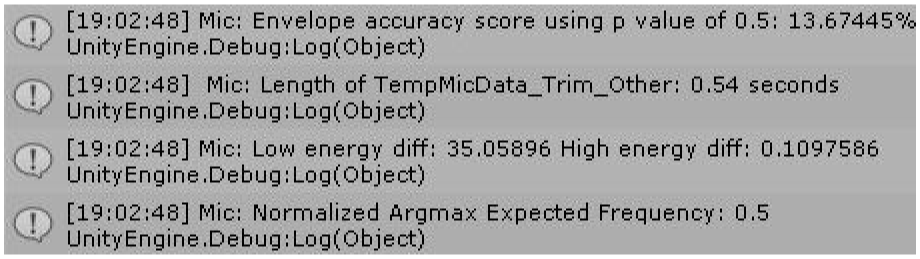


```
[18:59:04] Mic: Envelope accuracy score using p value of 0.5: 29.15392%
UnityEngine.Debug:Log(Object)
[18:59:04]  Mic: Length of TempMicData_Trim_Other: 1.18 seconds
UnityEngine.Debug:Log(Object)
[18:59:04] Mic: Low energy diff: -32.80939 High energy diff: 7.222494
UnityEngine.Debug:Log(Object)
[18:59:04] Mic: Normalized Argmax Expected Frequency: 0.2222222
UnityEngine.Debug:Log(Object)
```

The best match is generated by repeating the model sentence into the mic. Note the accuracy score is 29% (>20%) and the frequency score is 0.2222222 (within 0.2 of the model sentence's frequency score of 0.125). Furthermore, there is a clear dip in the error plot.

Ok:

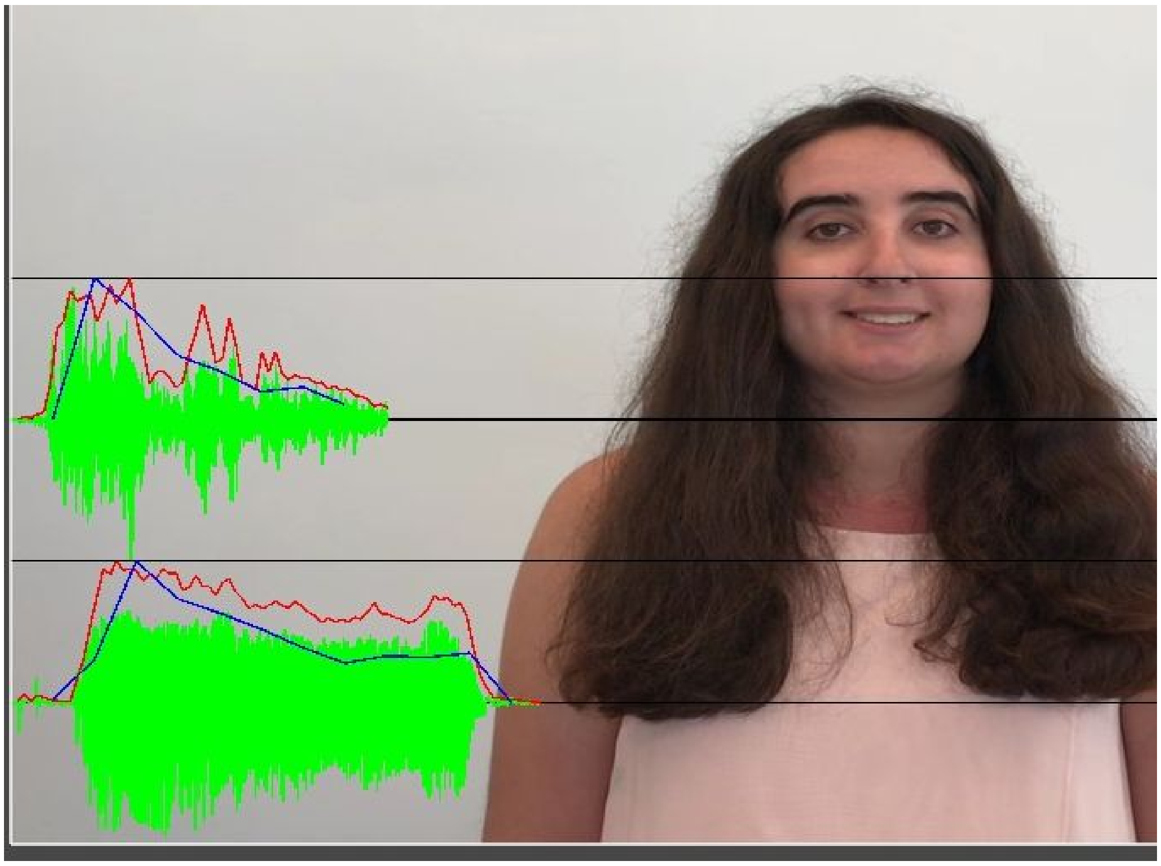


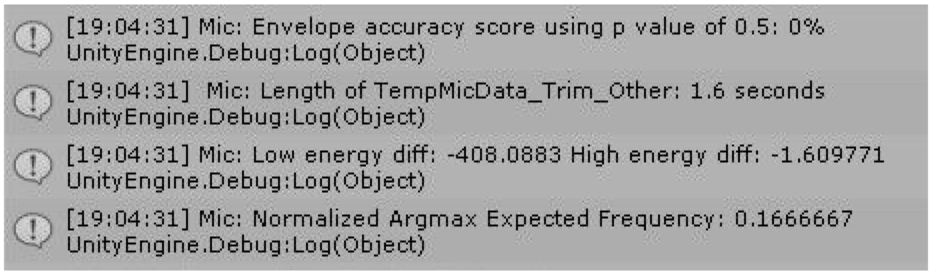


```
[19:02:48] Mic: Envelope accuracy score using p value of 0.5: 13.67445%
UnityEngine.Debug:Log(Object)
[19:02:48]  Mic: Length of TempMicData_Trim_Other: 0.54 seconds
UnityEngine.Debug:Log(Object)
[19:02:48] Mic: Low energy diff: 35.05896 High energy diff: 0.1097586
UnityEngine.Debug:Log(Object)
[19:02:48] Mic: Normalized Argmax Expected Frequency: 0.5
UnityEngine.Debug:Log(Object)
```

An “Ok” input is generated by saying “Hi” into the mic. Note the accuracy score is 13.67% (<=20, >10) and the frequency score is 0.5 (not within 0.2 of the model sentence’s frequency score).

Incorrect:





```
[19:04:31] Mic: Envelope accuracy score using p value of 0.5: 0%
UnityEngine.Debug:Log(Object)
[19:04:31]  Mic: Length of TempMicData_Trim_Other: 1.6 seconds
UnityEngine.Debug:Log(Object)
[19:04:31] Mic: Low energy diff: -408.0883 High energy diff: -1.609771
UnityEngine.Debug:Log(Object)
[19:04:31] Mic: Normalized Argmax Expected Frequency: 0.1666667
UnityEngine.Debug:Log(Object)
```

An incorrect input is generated by making a constant vowel sound (in this case, "UUUUU") directly into the mic. There are clearly many samples where the input envelope overshoots the model envelope because the input is a constant, loud sound. Thus, the overshoot penalty addresses this issue and decreases the score accordingly--in this case all the way down to a clamped 0%.