Memo

To: Professor Pisano, Professor Osama, Professor Hirsch

From: Krishan Eskew
Charles Mo
William Nilsen
Mia Hernandez

Team: 12

Date: 2/28/23

Subject: Prototype Test Report #2

Software:

- Unity Application
  - Game Assets
    - Rabbit
    - Unity UI objects (text, shapes, etc)
    - DreamScape environment
    - Fruit package
  - C# scripts
    - Main.cs, MoveWithTerrain.cs, buttonControl.cs, ChangeScene.cs, ClapDetector.cs, DragObject.cs, FilterButterworth.cs, Game1.cs, Game2.cs, Game3.cs, MovingAverage.cs, StreamingMic.cs, ThirdPersonCam.cs
- iPad
  - Unity app simulator

Set up:

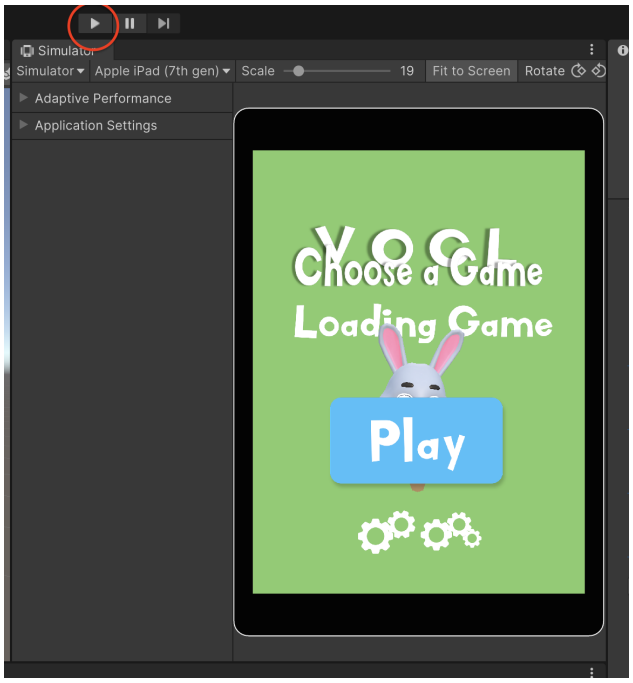
The only required hardware for this prototype is a computer capable of running Unity 2021.3.13f1 and an iPad with the Unity application simulator downloaded. Connect the iPad to the computer via a USB

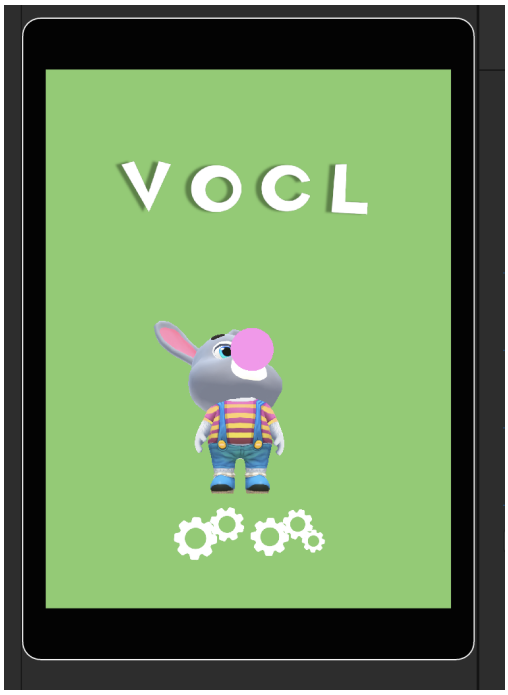
cable; a USB-c to USB convertor is required on Macbooks. Upon starting Unity, navigate to Edit on the toolbar, then go to Project Settings and then to the Editor tab. Once there, click on the iPad that should be listed on the drop down menu. Pressing the play button on the Unity IDE will start the game and position the game assets (i.e. rabbit, fruits, text) as defined graphically and programmatically via the Unity IDE and C# scripts.

Test Procedure:

Unity Application

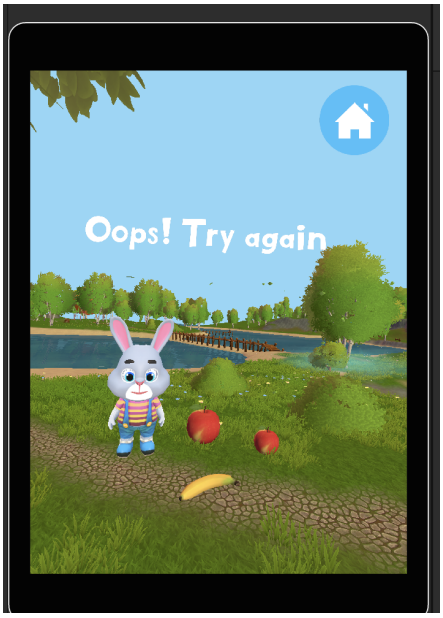
1. Press the Play button located on the Unity IDE. This will start with the loading screen on the simulator as well as the iPad.



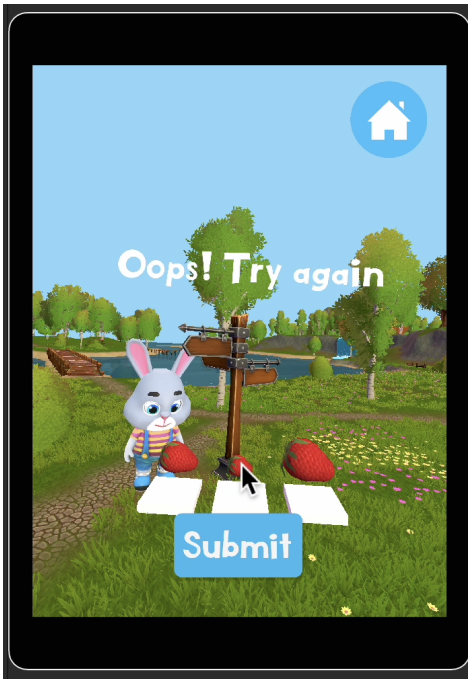


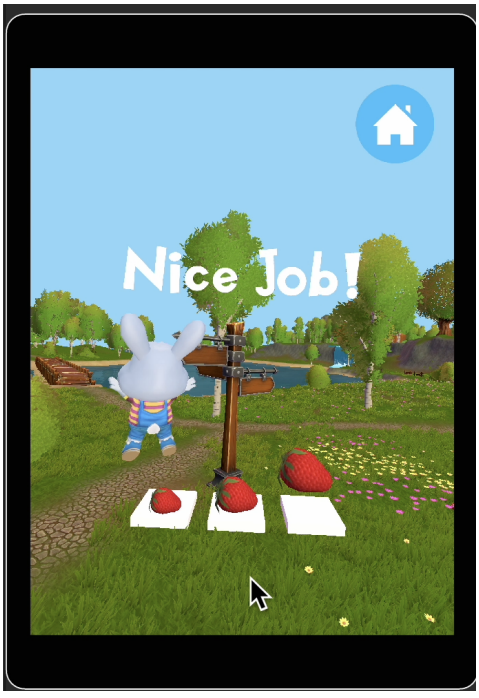
2. When prompted with the “Play” button, tap the Play button. This will start the game.

3. Tap on the largest apple, if incorrect the user is prompted to try again. If correct, the rabbit asset does a celebration animation and runs to the next destination.

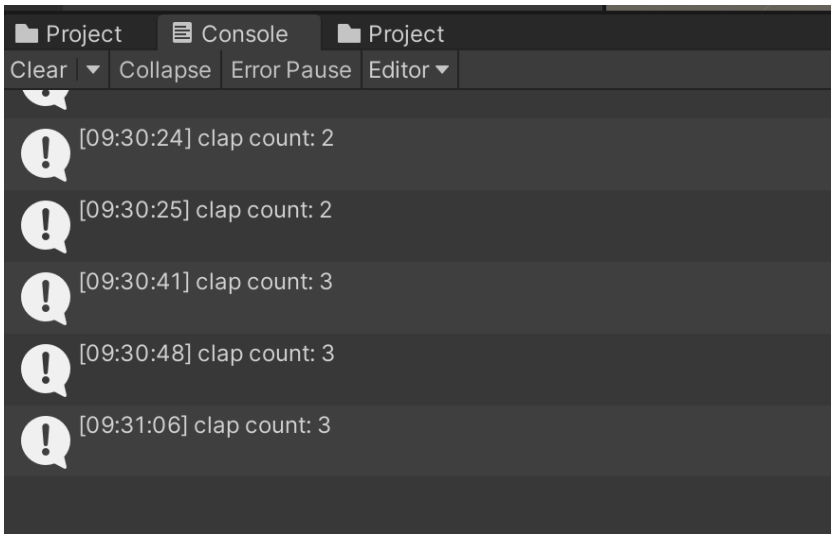


4. Order the strawberries by dragging them. Once done, tap the ‘Submit’ button. If incorrect, the user is prompted to try again. If correct, the rabbit asset does a celebration animation and runs to the next destination.

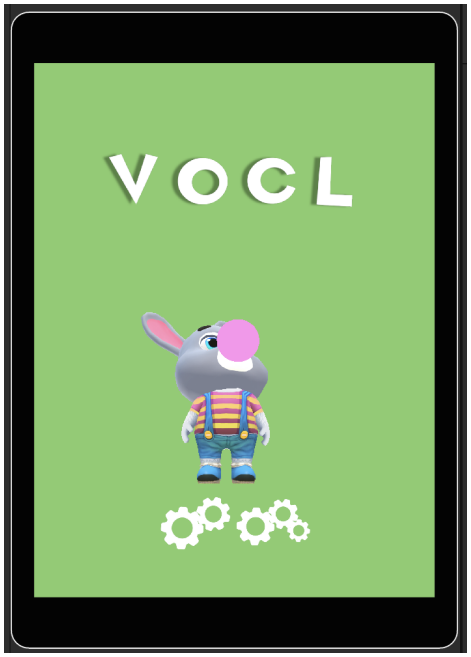




5. Clap three times. On the Unity console, you will see the amount of claps detected.



6. Tap on the ‘Home’ button to go back to the ‘Choose a Game’ screen.



Measured Criteria:

Unity Application

1. The Unity Application loading screen lasted for 10 seconds to account for the time lag on the iPad simulator.
2. The Play button successfully changed the active GameObject to allow the user to play the game.
3. In the first task, the Game1 script is able to detect if the correct fruit is tapped.
4. In the second task, the checkOrder function is able to distinguish if the fruits are in the correct order by comparing their x-coordinates.
5. In the third task, the ClapDetector script is able to detect different amounts of claps during intervals using the updated algorithm.

Conclusion:

Our second game prototype demonstrated the capability of our Unity application to animate our purchased assets as seen in our loading and in-game screens. Additionally, we implemented 3 separate game tasks within our Unity app. The first two games test reading comprehension and incorporate touch based input. The third is able to test the user's ability to engage by prompting the user to clap three times. Our audio peak-counting algorithm is able to count the number of claps. The rabbit Unity asset is also able to congratulate the user by jumping, prompt them to try again if needed, and run to the next location using animation controllers and our C# scripts. The first game makes use of Unity's tag feature which is how we are able to tell when the user chooses an incorrect object. The second game also has boundaries – in DragObject.cs – so that the user cannot drag objects out of the screen or below the white planes. With these two details in mind, game edge cases were successfully accounted for. The GUI and game were visually engaging to the professors and TAs, which shows that we made good use of our budget in purchasing interesting assets.

The next steps in our project will include making an end-of-game screen, designing an API for our machine learning model, and improving the working conventional audio algorithm. As for the API, the team has determined this would be the most straightforward way to get our machine learning model working with our current Unity application. A Flask application running in Python is most likely what will be used to host while the Unity application will be the client that sends audio .wav files. Lastly, the application will need to be built using the Apple developers key our client has provided.