**USER MANUAL**

**Hololens App:**

We have deployed it in the Microsoft Windows Store as an App.

Download the app

Run the App, Press start Begin the Game.

Boxes flash in an order, remember the order and tap the correct boxes.

If you are correct then your score increases by 5.

If you are wrong then you either play again or start over.

You also have the option to quit out if you feel like at any point of the game.

**Android App:**

We have deployed our game Beat That Android App in the Google Play store.

Download the app

Once downloaded simply open up the app and hit the start button to begin the game.

The game is simple. You need to remember which boxes flash and click on the order they flashed.

You have 3 replays per play.

Once a player loses they are shown their score and asked if they want to replay or quit the app.

If a player was to click on the boxes while they are flashing they automatically lose.

**Website Game:**

Open up the webpage <https://www.acsu.buffalo.edu/~satnamsi/>

When the web page is open pick which level of hardness you want Easy,Normal,Hard.

Once you pick the level you want just hit start button.

The game will begin and the color will flash just choose the color that blinks.

If you pick the wrong color you will lose and asked to pick a level of hardness you want again.

**DEVELOPER MANUAL**

**Hololens**

This was a Unity project with a visual studio backend in C# to be deployed on the hololens.

It is currently being certified on the Microsoft Windows Store. And Also is hosted on Sourceforge.

There are multiple Scripts being run in this project in conjunction to the unity Holographic displays.

Main0.cs:

This is the main part of the Game. Here we keep track of the Game Objects located on the canvas in unity.

* Start function finds gameobjects on the canvas and assigns them to a variable to be updated later.
* The four boxes each have a color variable.
* Two of which is set to false at beginning to keep hidden till user gets 15 points then its set to true in test correct.
* This is also where we add the initial random integer from range 0 to 2 to list called pattern.
* The *score* object’s text is set to 0
* The global variable *play\_back* is set to true
* Then call the atart() which starts the playback coroutine in flashanimation.
* After atart() finishes *play\_back is set to false.*
* The next function that runs is to testCorrect(). It takes in a click to test.
* Checks whether atart() has finished or if you hit the wrong box while tapping the sequence played. If you did any of that you lose the game.
* If you continuously match the pattern, the *score* increases and atart() is called again.
* As stated above you get more boxes added after reaching 15 points on the score.

Flashanimation.cs:

This is code that manages the cube flashes and time in between flash sequences. This is a Ienumerator function that takes in the list of ints and the iterates through it to flash the corresponding box to its number.

BlueScript.cs:

This script has one function that calls our Main0.cs Script with a button click to test. In the pickACube() function, there is a string passed in and we match that to our available boxes to check in that scene via Main0.

ClickAction.cs:

This Script takes in the user input given of button clicked on by either the GazeGestureManager.cs transmitting the object being clicked on or the SpeechManager0.cs transmitting the object targeted via voice command. The ClickAction script sends the cube clicked on over to BlueScript to analyze.

GazeGestureManager.cs:

Recognizes “tap” gesture and sends a message to clickAction as to which game object was clicked.

SpeechManager0.cs:

Takes in user voice commands, interprets them with keywords added and then sends them to CubeCommand.cs to call pickACube from BlueScript on corresponding game object or fulfill requests , such as “onSelect”, which transmit object being selected to ClickAction or “Quit”, which terminates the program.

CubeCommand.cs:

Obtains a string from SpeechManager0.cs to parse and determine which game object is meant to be selected and calls pickACube from BlueScript based on that game object.

**Developer Manual-Sound/Moving Boxes**

**Hololens**

To get sound on the boxes when they were clicked all i had to do was to add a new component to the unity boxes called audio listener. So just had to put it on the boxes and just write a script in which that would make the boxes make a sound when they were clicked upon. Just need to start sound on click on and shut sound of when it was done being clicked.

Getting the boxes to move was more of a challenge because either the box would move too fast or not move at all. So the way i went on to do this was by calling (GetComponent) and rendering the color of the material. Then calling transform and and rotating it 30 degrees so it just moves slightly enough so people can see the box tilt.

**DEVELOPER MANUAL**

**Andorid App**

MainActivity.java class:

Main menu scene that has a **start** and **quit button.** The start button starts the game and the quit closes the application.

Game.java class:

Game scene that has 4 square boxes, **Quit** and **Replay** button. When the scene starts a random number between 0 and 3 is chosen, this will represent on of the 4 boxes, and added to an int list called *\_pattern*. The function *startAnimation()* is called, which goes through *\_pattern* and makes one of the boxes flash corresponding to what number is at the position in the list; 0 == red, 1 == blue, 2 == yellow, 3 == green. Depending the score the current player has determines how long the flashes will last, ie. if the score is less than 5 the duration is 1000 and the delay would be 2000 while if the score was greater than 5 but less than 10 the duration would be 800 and the delay would be 1500. If the player was to click on the boxes while they are flashing the player automatically loses. Each box implements the *OnClick()* method, this checks if the box pressed is the correct one at that point of the sequence, if it is correct it increments the *Count* text by 1 and if the whole pattern is correct it resets the *Count* text back to 0 and increments the *Score* text by 5 and adds a random number from 0 - 3 to *\_pattern.* And starts the loop over again, flashes then checks for player input. If the player picks the incorrect box the score is put into the bundle class so it can be used in the *Score.java* class.

Score.java class:

New scene that display the score of the player that was transferred from the Game.java class using the Bundle class. It first checks to see if the text from the bundle class is an integer, if it is then we multiply the number by 5 and display the score, otherwise we just display the score. In this scene there is also a **replay** and **quit** button, the replay button takes you to the Game.java scene so the player can play again while the quit button exits the application.

**Developer Manual-Website**

I made in the game in javascript and a little CSS. In css i started off by making the padding so the boxes were centered and everything was in the right spot.I also made all 4 of the boxes and gave them all different colors. I then started to work on the flash animation in Javascript. I tried to work on the sound for the button’s but could not figure it out in time. Then it was time to randomize the color pad into different color when it blinks every time. Then i worked on the score and how to +1 every time you get a box right for each sequence. I even need to create a switch case to make it add. Then i worked on what to do to the game if you pick the wrong sequence them that you were told and how to make you start over again.