Gesture Based UI Project

Video Demo Link

https://www.youtube.com/watch?v=D5G7TaQQ-0w&feature=youtu.be

Main Objective

Develop an application with a Natural User Interface. This should be a local implementation of the application using gestures to interact with it. For example, a voice controlled application fits the parameters of gesture based control. You can expand out to include real-world hardware and use this as an opportunity to prove a concept. You can reproduce a classic game or system using a gesture-based interface. For example, a platformer game or a navigation application using Kinect or voice control. Maybe Tetris using the Myo armbands to control the blocks, or Flappy Bird using the Kinect as the controller. Applications with multiple users are also acceptable.

My Idea

My original idea for this project was to create a game very similar to "Super Mario Bro's". However instead of using the classic arrow keys to move the player, I planned on using just voice commands. Unfortunately, I had to scrap this idea as I could not come up with a solution for allowing the player to move left or right while jumping which is a classic feature in the original "Super Mario Bro's" game.

I then came up with the idea of creating a basic 2D platformer, which involves the user navigating the character through different obstacles using only voice commands. The main objective of the game is to get the character to the chest which is at the bottom of each level. As the user progresses through the levels, the difficulty of them increase. For example in some of the later levels, platforms will begin to move and traps are also added which restart the level if the user comes into contact with them.

Purpose of Application

The main purpose of this application is to implement voice commands into a 2D platformer game where the user only uses voice commands to move the character. The main objectives I have for this project are:

- Allow the user control the character by using voice commands.
- Have a start menu which uses voice commands to navigate it.
- Allow the user to pause the game by simply saying "Pause".
- Allow the user to access a help menu to view the controls of the game and find of the main objective of the game.

Gestures Identified as appropriate for this Application

As I have mentioned above the gestures that I chose to implement were voice commands. I had to do a vast amount of research on this as I had never integrated voice commands with a game used by Unity. Firstly I had to open my Unity project, go into the build settings and download the Universal Windows Platform package for unity. Once this was downloaded I had to go into go into the build settings once again and ensure that the microphone on my laptop would be able to listen out for voice commands. I did this by:

- Opening the project in Unity
- Select File at the top left of the screen
- Select Build Settings
- Select PC, Mac and Linux Standalone
- Select Player Settings
- Select the Windows Icon
- Scroll down to Capabilities and check the checkbox associated with the Microphone.

How I implemented voice commands into the Application

First of all I created a C# Script called VoiceCommands.cs which is responsible for the majority of voice commands in this application. I implemented the voice commands by doing the following steps:

Import Unity's speech recognition package into the script:

```
using UnityEngine.Windows.Speech
```

Initialized the keyword recognizer variable which tells Unity to look out for keywords:

```
private KeywordRecognizer keywordRecognizer;
private Dictionary<string, Action> actions = new Dictionary<string, Action>();
```

Created actions associated with the different keywords:

```
actions.Add("right", Right);
actions.Add("left", Left);
actions.Add("up", Up);
actions.Add("down", Down);
actions.Add("play", Play);
actions.Add("help", Help);
```

Set up methods that are associated with the keywords that run if the certain keyword is heard

```
private void Right()
    {
        transform.Translate(1, 0, 0);
    }

    private void Left()
    {
        transform.Translate(-1, 0, 0);
    }

    private void Up()
    {
        transform.Translate(0, 1, 0);
    }

    private void Down()
    {
        transform.Translate(0, -1, 0);
    }
```

The purpose of the methods listed above allowed the player to move right, left, up and down on the X and Y axis. The 'Right' method moves the player +1 spaces on the X axis. The 'Left' method moves the player -1 spaces on the X axis, the 'Up' method moves the player +1 spaces on the Y axis and the 'Down' method moves the player -1 spaces on the Y axis.

Technologies used in creating the application

While creating this application I used many technologies such as:

Unity

I used Unity as the main game engine as I had worked with Unity previously in college and really liked how it worked and how easy it is to use.

Visual Studio 2017

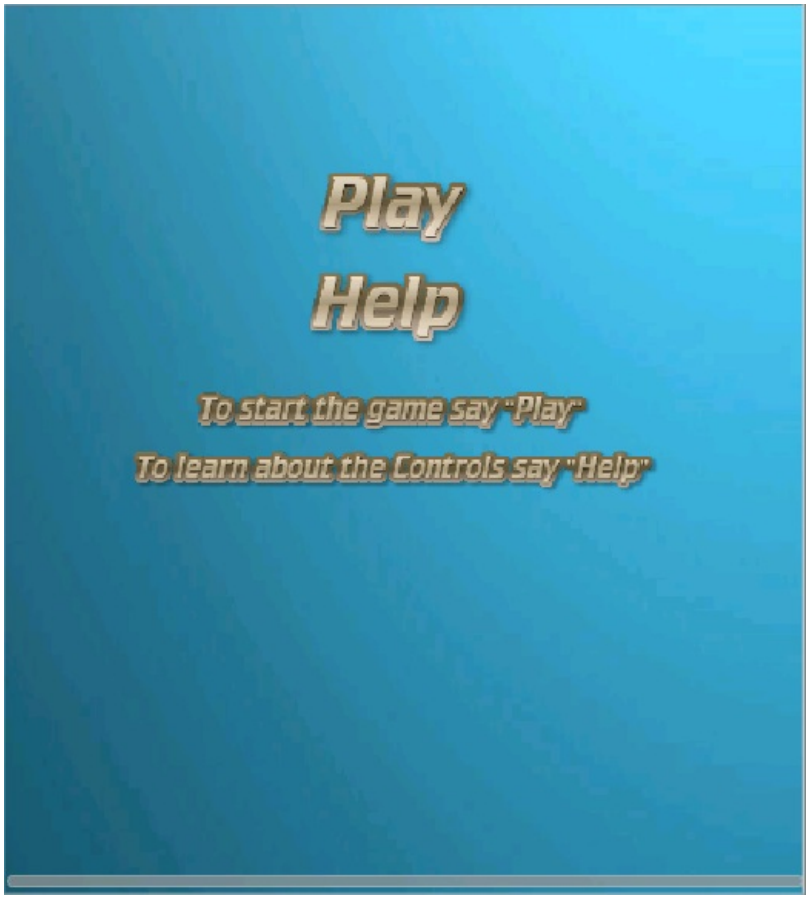
I used Visual Studio Code 2017 to write all of the code, which was all written in C#. The main reason for this is that I also had past

experience with it as I used it throughout my college years.

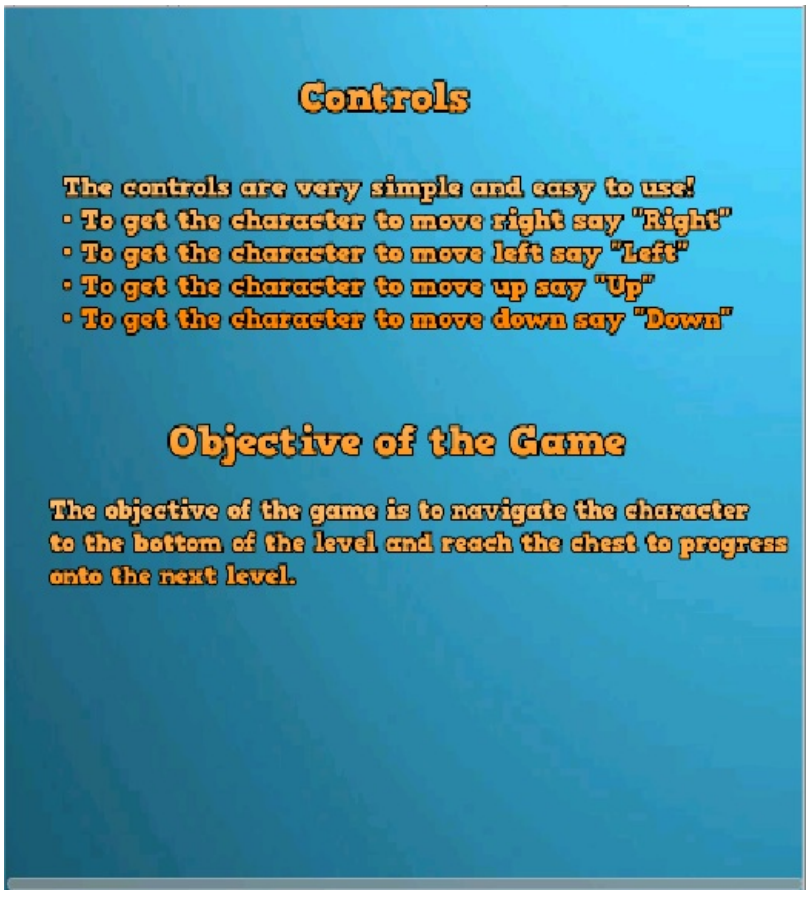
Architecture for the Solution

In game screens

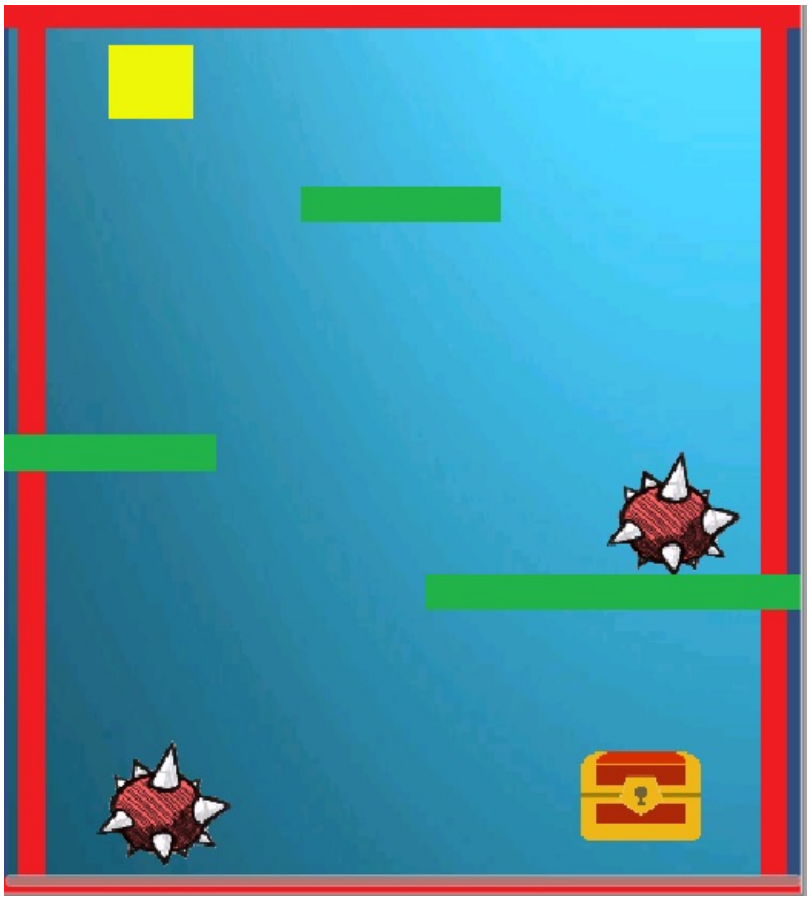
Start Screen



Help Screen



Game Screen



Conclusions

I really enjoyed developing this application as I had never used any sort of gestures to control an application before. I really enjoyed learning how exactly voice commands are integrated with games and the methods you use with them.