Software Design of Voice Control in 2D Platformer Project

2D Platformer is a sample open project provided by Unity which can be downloaded from here. It is a simple action RPG game which player can control character and destroy enemies. There are 4 simple actions that can be performed: LEFT, RIGHT, JUMP, and SHOOT. Because of the nature of voice control, player can only perform one action at a time. So, using only voice for all 4 actions may not be a good idea in this game which requires low latency and may have multiple actions in a second. In this pilot work, I implemented voice control for JUMP and SHOOT actions. I also provided calibration so player can calibrate his/her voice all the time.

It is obvious that the key part of this work is speech recognition, however, we need to trade off between speed (latency) and accuracy. In this case, speed is a must. The tricky thing is that "Do we really need high accuracy?". We obviously need accuracy for JUMP action. And, it should be better to make SHOOT action pretty loose since player may get excited and say something out of the tone which should be consider as SHOOT as well. So, my design concept will focus on speed, accuracy of JUMP action and vast acceptance of SHOOT action.

I added C# script (VoiceControl.cs), modified PlayerControl.cs and Gun.cs (shown in red bold text in the following 2 pictures), so, they can receive commands and perform actions.

```
// PlayerControl.cs
void Update()
{
    if((Input.GetButtonDown("Jump") || VoiceControl.jump==true) && grounded)
    {
        jump = true;
        VoiceControl.jump = false;
    }
}
```

```
// Gun.cs
void Update ()
{
    // If the shoot button is pressed...
    if(Input.GetButtonDown("Fire1") || VoiceControl.shoot==true)
    {
        VoiceControl.shoot = false;
    }
}
```

Now, let's look into the main actor VoiceControl.cs. The most popular approach to recognize speech is to look at the frequency spectrum which can be obtained by Fourier Transform. In each frame, we will analyze the spectrum of the voice. We can use AudioSource.GetSpectrumData method to obtain spectrum of the voice (I believe that it performs FFT internally.). However, one word contains many frames of voice and the number of frames varies. So, instead of keeping all frames, I use average spectrum to represent the whole word.

Player can press (and hold) 'J' and speak "Jump" to calibrate JUMP action and 'S' for SHOOT action. The system will collect player's voice, then, calculate average spectrum and save into *spectrum_jump* and *spectrum_shoot*. Then, player can speak "Jump" and "Shoot" to activate the actions. The magic behind the scene is that the average spectrum of voice input will be calculated and compared with *spectrum_jump* and *spectrum_shoot*, then, the action will be predicted.

In conclusion, I created the script to listen to microphone, calculate frequency spectrum, calibrate by recording player's sample voice, predict action with simple similarity algorithm and send command to execute actions. I know that there are many tools and algorithms for classification problem, e.g., k-nearest neighbors, decision tree, support vector machine and so on. But in this case, I want to show my understanding in basic concepts so I implemented everything by my hands and made it as simple as possible.

If we want to improve this project, we should focus on noise reduction. My algorithm does not include this part so it does not perform well in noisy situation. Another solution will be calibrating environment and using adjustable threshold to reduce noise. Moreover, we can also use pattern matching to remove music, fed back from speaker to microphone, created by this game. I do not think that we need to improve classification algorithm for this game unless we add more commands, e.g., multiple shoots or throwing a bomb.

Finally, this was my really first time using Unity. I thought that speech recognition in Unity should not be that hard because I believed that there would be a lot of libraries and open source tools like when I was doing research in classification and activity recognition from accelerometer data which is pretty similar in concepts. Unfortunately, the free edition of Unity does not support C/C++ and I cannot find a way to integrate System.Speech to Unity development environment. Above all, thanks for this challenge. This one kept me busy and I was having fun. I like Dolby products and I also like this blind audition, please keep working hard!