AR Vocals Visualizer: Augmented Reality Pitch Visualization

Vamsi Immanneni

May 2024

Abstract

This application is a novel Augmented Reality (AR) project that enhances the experience of vocal music by overlaying a real-time visual representations of pitch above the singer's heads. It uses various iOS features to create the text display and anchor it to a person's body or face.

1 Introduction

The main goal for this app, beyond visualizing pitch, is to provide a platform for AR-text displays above a human object. Future developments can be endless, such as interesting 3D models for fonts and textures, or a song-recognition system, or speech-to-text. For now, we accomplish these main goals:

- Be able to track a person's head and display AR text above their head.
- The AR text should display a meaningful understanding of pitch and volume.
- The text should be easy to modify with plenty of options, and the object should be easy to move around in case the user wants it in a different relative position.

1.1 Technology Stack

- **ARKit, RealityKit**: Used for integrating augmented reality overlays
- **SoundpipeAudioKit, AudioKitEX, and AVFoundation**: Used for managing audio input and real-time pitch detection
- **SwiftUI**: Uses SwiftUI techniques for crafting a user-friendly interface

2 Design and Development

Here is a basic overview of the app via the front-facing camera:

In the above images, the pitch (or the non-recognition of it) are displayed through the text, while the text's thickness represents how loud the note is.

2.1 Development process

The first part of the project was to develop the MVD, which was to integrate the audio libraries to get real-time pitch detection, while ARKit was left for later. This took a while because it was my first time coding an iOS app, and XCode and Swift can be unintuitive in the beginning.

The next step was to create the AR display. This came with many challenges. First, I needed to integrate an ARKit library to be able to create an anchor for the AR text to go. The main issue here is that the front and back-facing camera allow separate AR abilities, for the back-facing camera I used "ARBodyTracking-Configuration" and for the front I used "ARFaceTrackingConfiguration". Finding tutorials on these options was not easy, as coding for iOS is limited to MacOS users, and XCode updates tend to change significantly from previous versions.

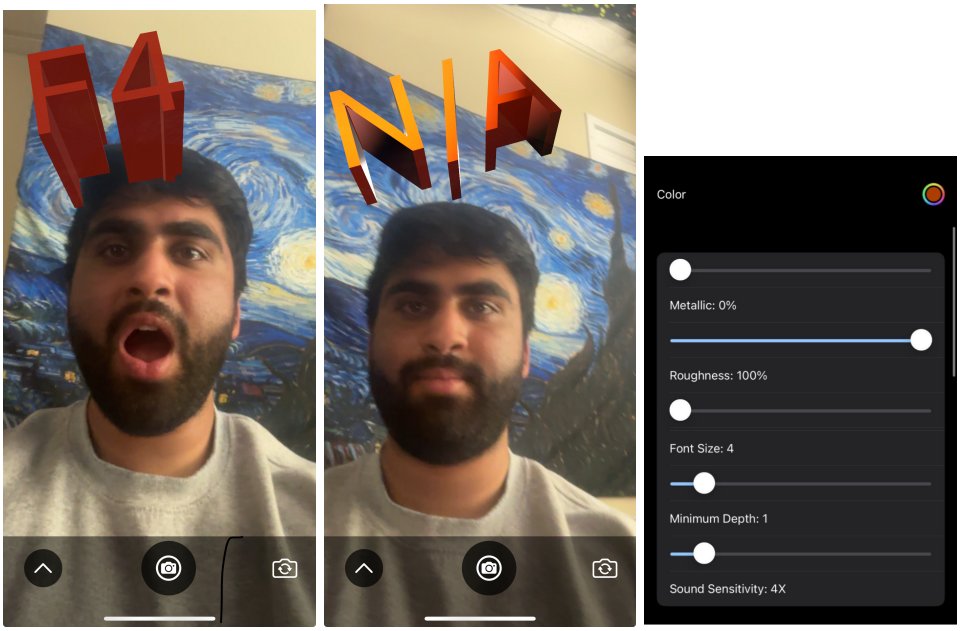

Figure 1: On the left is an image of me producing the note F in the fourth octave with a standard texture display. The middle is an image where it does not recognize the pitch, but also has some modifiers, like full shininess through the metallic property. On the bottom of the two images on the left is a control bar with three buttons: on the left is to show and minimize the menu, the middle is a button to take a picture (has not been implemented yet, should be replaced with a video button), and the button on the right is to toggle the front or back-facing cameras. The third image is the menu, with options for color (with opacity), Metallic (how shiny it is), Roughness (how matte it is), Font Size, the minimum depth of the text, and how sensitive it is to sound inputs.

After some time, I finished a bare-bones AR-pitch display, but the issue here was it was difficult to find the correct vector for the positioning of the text. After much tweaking, I realized that it is far better to allow a user input to move the display instead of fixing the offset values. Here, I learned how to handle SwiftUI gestures, and added the next two features: panning and pinching. Panning is done by tapping on the display with two fingers and dragging, which moves the text in the x and y directions (up and down and left and right). Pinching with two fingers, on the other hand, modifies the z distance from the anchor, making it closer or farther from the camera.

After that came the idea to add as many menu options as reasonable. Here, I thought it best to implement the SimpleMaterial structure from RealityKit, which gave the options shown above. Through this process, I also learned how to create a custom material to add custom textures and figures to the display if needed, but I left this idea for future work.

The final feature, which I got stuck on, was the video recording feature. It is only partially implemented, as I will need to learn more Swift to develop a workaround. The main issue here is that the app itself needs the mic to process the audio, while screen recording through ReplayKit (the most commonly used library for this) does not allow the mic to record simultaneously. That's why in the demo video I recorded the audio separately from a different device.

2.2 Limitations

The main limitation with this project is the jitter and lag of the AR display. To reduce the jitter, I implemented the popular lerp and slerp algorithms to smooth the movement. Via a large smoothing factor, the jitter slows down, but is still apparent. The other problem is the lag. Whenever a new anchor comes into view, the display freezes for a bit, and often the rear-camera AR display just freezes entirely. However, these

issues are periodic, and some days the app ran smoothly, which leads me to believe this performance issue is due to it being run on the development environment with the debugger as opposed to being downloaded from the App Store.

3 Novelty and Contributions

The main novelty is this is a new idea: displaying pitch with an AR display, and it works. Like mentioned in the introduction, this leads to a bunch of future work to make this app more creative and powerful. Overall, once the lag gets fixed, this could end up being a successful application for social media.

4 Instructions on How to Run and Demo

To run this and watch the demo, the instructions will be on the main ReadMe of the github.

5 Citations and References

I did not use any code apart from the standard libraries. I did research through a ton of YouTube videos to learn Swift programming, however.

- ARKit Documentation: `https://developer.apple.com/documentation/arkit`
- AudioKit Library: `https://audiokit.io/`
- SwiftUI Tutorials: `https://developer.apple.com/tutorials/swiftui`