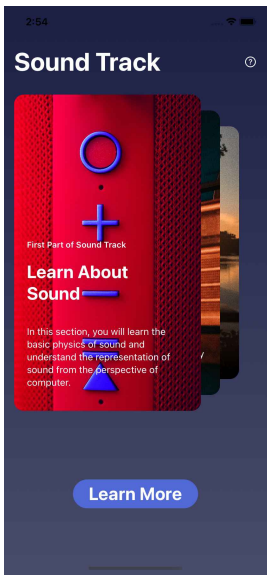
README

Hello, judges, I am a college student who is very interested in swift development, and I also like guitar very much, so this time I want to combine them to complete my WWDC student challenge work, which is called Sound Track.

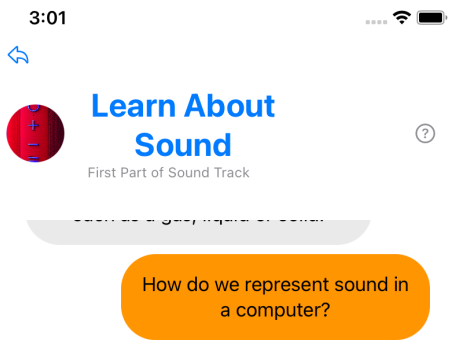
Introduction

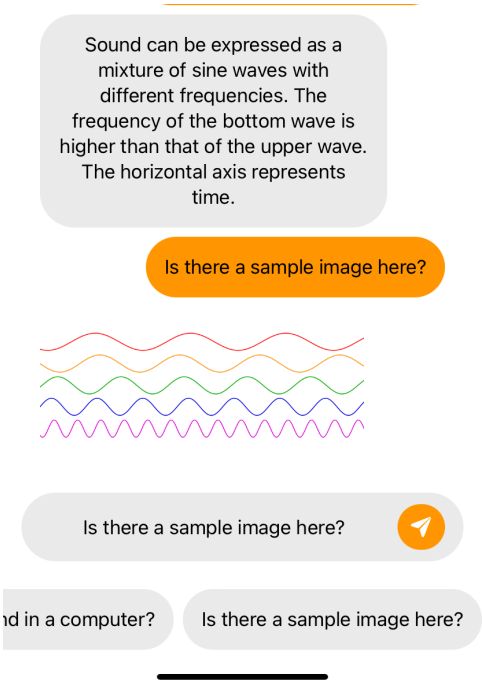
My work consists of four parts: Learn About Sound,Make Single Frequency Sound by Swift,Add Guitar Effect and Play Guitar by Swift. You can slide left and right to select the reading part, and click the button at the bottom to experience the part. I suggest you experience it in turn to get better results, and each part adopts the form of dialogue to make it more interactive and interesting. Of course, there may be some shortcomings, because I'm not very familiar with SwiftUI development. Thank you for your forgiveness.The following is the main interface



Learn About Sound

In this section, you will learn basic knowledge about sound. You can click the keyword under the text field to enter, and click the send button to get the relevant answer.





Make Single Frequency Sound by Swift

In this section, you will learn how to use swift to make a single frequency sound. I use AVFoundation and AudioUnit to control the device to make single frequency sound. I set the sample rate 44100 and the frequency 329.6Hz. The key codes are as follows

Swift

```swift
        auAudioUnit = try AUAudioUnit(componentDescription: audioComponentDescription)

        let bus0 = auAudioUnit.inputBusses[0]
        let audioFormat = AVAudioFormat(
          commonFormat: AVAudioCommonFormat.pcmFormatInt16,  // Int16
          sampleRate: Double(sampleRate),  // SampeRate
          channels:AVAudioChannelCount(1), //Single Channel
          interleaved: false )                  // interleaved stereo

        try bus0.setFormat(audioFormat ?? AVAudioFormat()) //   for speaker bus
        auAudioUnit.outputProvider = { (  // AURenderPullInputBlock?
          actionFlags,
          timestamp,
          frameCount,
          inputBusNumber,
          inputDataList ) -> AUAudioUnitStatus in
          self.fillSpeakerBuffer(inputDataList: inputDataList, frameCount: frameCount)
          return(0)
        }
```

Then I get the feature points by sampling function sin(). The key codes are as follows

Swift

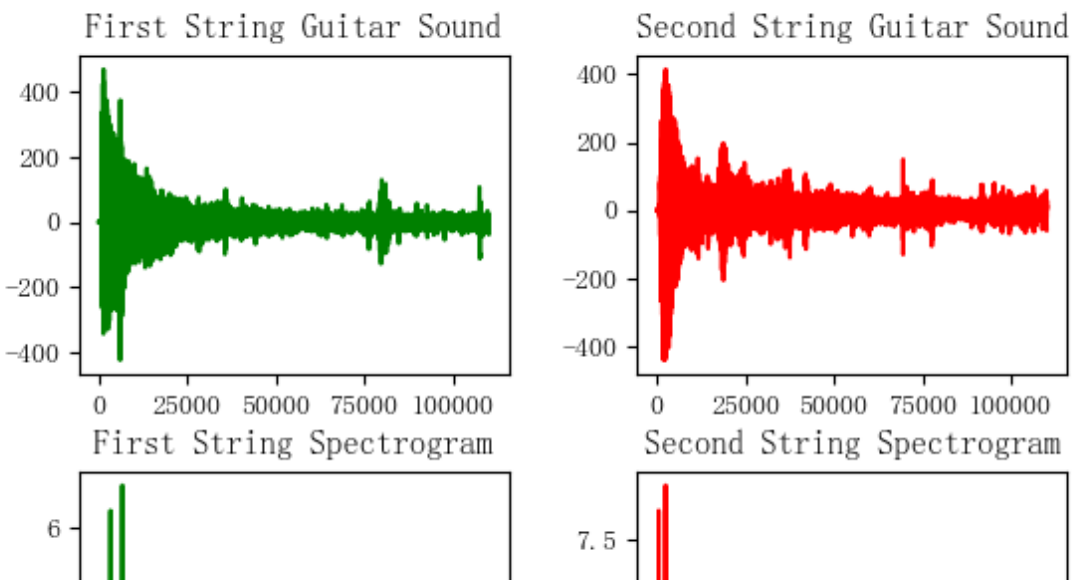
```swift
        var a = self.phY    // last phase
        let d = 2.0 * Double.pi * self.f0 / self.sampleRate   // delta

        let bufferPointer = UnsafeMutableRawPointer(mBuffers.mData)
        if var bptr = bufferPointer {
          for i in 0..<(count) {
            let u = sin(a)       // create a sinewave
            a += d ; if (a > 2.0 * Double.pi) { a -= 2.0 * Double.pi }
            let x = Int16(v * u + 0.5)    // scale & round

            if (i < (sz / 2)) {
              bptr.assumingMemoryBound(to: Int16.self).pointee = x
              bptr += 2  // increment by 2 bytes for next Int16 item
            }
          }
        }

        self.phY    =  a          // save sinewave phase
        self.toneCount -= Int32(frameCount)  // decrement time remaining
```

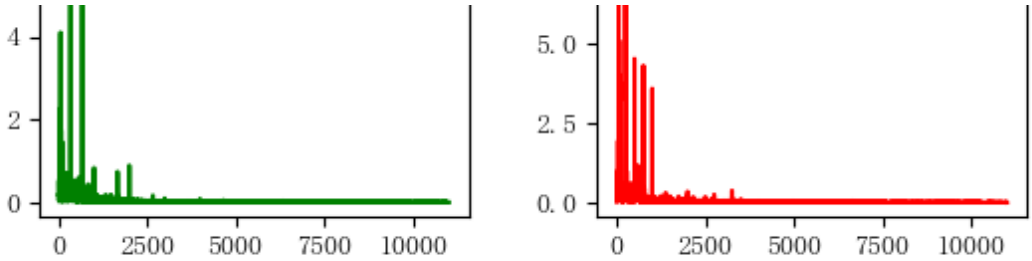
You can also get these things through dialogue. I want users to learn to manipulate iOS audio devices in this section.

Add Guitar Effect

In this section, we process the sound of the previous part to make it have the timbre of guitar, although it is not particularly similar in the end.

I use Fourier transform to sample the sound of the guitar, as shown in the figure below



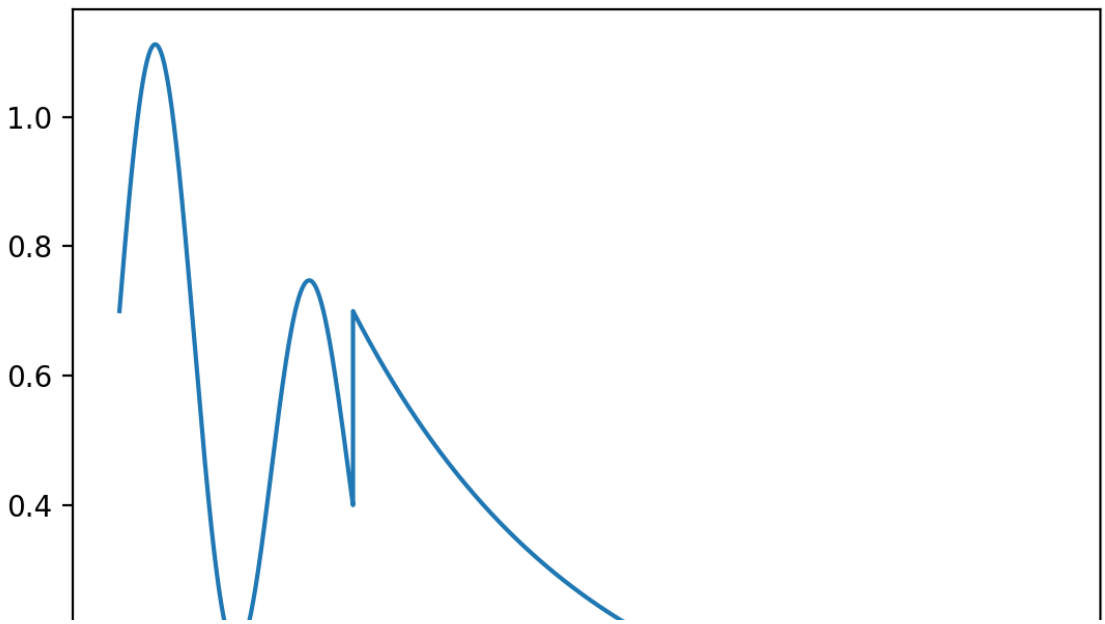


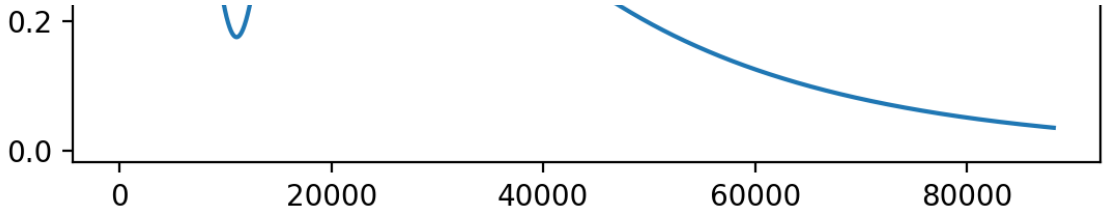
Due to 2019-nCov , I was closed in the dormitory, and there was inevitable noise interference.I use the following formula to fit the timbre curve of the guitar.

$$0.7 - \frac{x}{10\pi} + \frac{1}{2}(1 - \frac{x}{6\pi})\sin(x)$$

$$e^{(-x)}$$

The graph of the formula is shown below

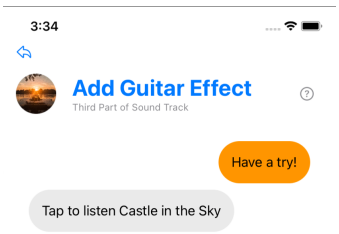


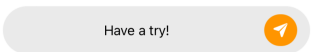


Use the sin sampling point in the previous section to multiply this function to obtain the simulated guitar timbre.

According to Equal Temperaments,I get the frequencies of the first to sixth strings of the guitar are 329.6, 246.9, 196.0, 146.8, 110.0 and 82.4 respectively. On the same string, the frequencies of all frets from low to high are an equal ratio sequence, with a ratio of $2^{1/12}$.

Using the above data, I can get a simulated guitar! I prepared the guitar music of the Castle in the Sky. You may have to wait for it to process for some time before you can hear it





Play Guitar by Swift

This is the final section of my work. You can enter play castle in the sky here to listen to Castle in the sky, or enter a string according to the rules to play the guitar.

The custom coding rules of music are as follows. Use ("12", 4) to represent one string, two frets and full notes of guitar, separated by | . 4, 1 and 0.25 represent full note, quarter note and sixteenth note respectively. Each input will be used as all the notes that the guitar string should dial in the current section. You can use the first section of the city of the sky as an example: ("13",1.5)|("12",0.5)|("13",1)|("13",0.5)|("17",0.5).Then you can input 'play' to play the music.

I hope you like my work! I know my English is poor. Thank you very much for seeing here.