Pitch Perfect

CS 148 Final Project - Summer 2017

George He
Stanford University
georgehe@stanford.edu

Abstract

Audio cues and interactions within the realm of gaming has led to the innovation audio-reliant systems of interaction. In this project, we propose a visual game system generated from and controlled with audio input to help improve pitch detection. Specifically, the proposed game will require users to respond to audio-generated events, sampled from input music as a regular interval.

1. Inspiration and Overview

The general game objective and visualization is inspired by lecture 6 on Concepts and Algorithms of Scientific and Visual Computing [2]. Using homework 3 and a base template for object loading and rendering, objects constructed could deform according the audio input. Specifically, torus objects [5] loaded into OpenGL were deformed to match the frequencies of the inputs, with angles corresponding the frequencies and the magnitude of deformation corresponding to the magnitude of the frequency. Scoring for the game was calculated by a heuristic using how much the user-provided input deviated from the target frequencies generated by a sampled points of an input song/wav file.

I do not claim credit for any of the audio used [4] or object files[5].

2. Libraries and Techniques

Libraries used to aid in audio collection and data manipulation included RTAudio[6], sndfile[1], and fftw[3].

2.1. Fast Fourier Transform

Transformation of the audio data into the frequency domain required use of the fast fourier transform to approximate an N-point fourier transform, where N corresponds to the number of sampled points. The equation for the fourier transform is modeled as follows:

$$F(\omega) = \int_{-\infty}^{\infty} f(x)e^{2\pi i\omega x}dx$$

In order to approximate the fourier transform, a fast fourier transform provided by fftw[3] was used. The fast fourier transform is returned with two components (the real and imaginary). The frequency is calculated using the magnitude of the fft, which is only an approximation of the true fourier transform.

$$Frequency[i] = |FFT[f(i)]|$$

$$f(i) = i * sr/frames$$

With a sample rate, sr, of 44100 and 1024 frames per discrete fourier transform buffered input, the allows an approximation of the frequencies to the nearest 43 hertz. The reasonable vocal range for humans lies somewhere between E2[82 hz] and C6 [1 kHz] [8].

2.2. Torus Manipulation

In order to visualize the frequency, a torus is deformed. There are two types of deformation present in this project. The first type of deformation visualizes the entire audible frequency spectrum. In this visualization, toruses were deformed such that the angles of the torus corresponded to the frequencies while the magnitude of deformation corresponded to the magnitude of the frequency.

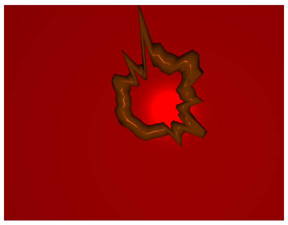

Figure 1. Visualization of full audio frequency spectrum

The second type of deformation attempts to summarize the information in frequency domain, by using either the magnitude of the largest single frequency bin or the as a parameter to tune the deformation. The extracted heuristic is used as an input to scale a portion of the torus:

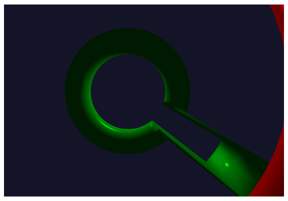

Figure 2. Visualization of single heuristic

2.3. Colorization

Similar to how the torus was deformed, the color of each torus was determined by a single heuristic of sum of all frequency magnitudes and then scaling it between 0 and 1. To cover the RGB spectrum with a smooth transition, a piecewise function to represent the color was created as follows:

$$C(n) = \begin{cases} [1-3d, 3d, 0] & n \in [0, 1/3) \\ [0, 1-3(d-1/3), 3(d-1/3)] & n \in [1/3, 2/3) \\ [3(d-2/3), 0, 1-3(d-2/3)] & n \in [2/3, 1) \end{cases}$$

Additionally, the color of the background is determined by the most recent score. Scoring points will push the background to a more green background while missing point values will cause the background to turn red.

2.4. Point Calculation

In order to determine how a user scored points, collision detection between the user-controlled torus and the "target" torus was determined by calculating the amount of overlap in angles of deformation between two singularly deformed toruses. This calculation is performed only when the two torus objects are very close to each other (distance less than 0.1 in rendered space). If the two toruses have similar/overlapping areas of deformation, they will not have significant collisions and thus the user will score points.

3. Challenges Encountered

3.1. Library Integration

Setting up and integrating the various libraries proved to be a major pain point. I ended up being stalled for roughly a due to issues linking and setting up the libraries. This issue was finally unblocked after going to a few office hours for help.

3.2. Frequency Detection and Fourier Transform

Understanding the fourier basis required a refresher on some basic physics/math, because the output of fftw didn't quite make sense initially. After understanding how sampling rate affected the frequency bins in the fast fourier transform, it was a straightforward process to find the frequency and incorporate it into the rendering of the world. Because audio collection is prone to noise, error, and background sound, it was often difficult to test how well the fourier transformation was actually detecting the frequencies. However, after using an online tone generator [7], it was much easier to test whether frequencies were correctly sampled using the fourier transform.

4. Conclusion and Relation to CS148

The initial proposal for the project is outlined as follows:

```
"We are going to create a game that is
controlled through sound.  The user can
control the deformation of 3D object
traveling through a path by changing the
frequency of their voice.  Users score
higher points by deforming the object
to better fit a pre-set mold.  This can
be tuned to help individuals train to
hit certain notes and sing to music!
As a stretch goal, users will be able
to input mp3/spectral audio files and a
playing field the for audio files will be
created by sampling points along certain
intervals."
```

Considering the initial goals, the project has successfully accomplished the initial goals in the project proposal. That is, the user is able to control the deformation of a 3D object while the environment around it changes, using their voice. Additionally, the points that a user scores is directly related to how well the object conforms to a mold. Even the stretch goal was realized by having the user Certain aspects of the development process, such as the modeling of the torus object using an object file as opposed to finding a mathematic model to generate the torus, offered different approaches depending on the use of outside tools/resources. With respect to cs148, this project heavily illustrates the applications of signal processing, physical simulation and animation, and concepts of 3D rendering in order to construct a virtual environment. Signal processing was applied to audio input and translated into a direct capability to extract the principal frequencies found in the audio. The physical simulation of the torus was used in the approximation of collision detection using stored attributes of the meshes, while the animation of the frequencies allowed users to interact with the audio using visual cues.

References

[1] E. de Castro Lopo. libsndfile. http://www.mega-nerd.com/libsndfile/, 2013.

[2] D. Hyde. Concepts and algorithms of scientific and visual computing. https://www.dropbox.com/s/rc8zxgxdvlqr0r1/L6b.pdf?dl=0, 2015.

[3] S. G. J. Matteo Frigo. The design and implementation of fftw3. http://www.fftw.org/fftw-paper-ieee.pdf, 2013.

[4] Onclassical. Onclassical demo. https://archive.org/details/onclassical-quality-wav-audio-files-of-classical-mu 2017.

[5] D. Palacios. Purpleprintkit. https://github.com/gamedevtech/PurpleprintKit, 2014.

[6] G. P. Scavone. Rtaudio. https://www.music.mcgill.ca/~gary/rtaudio/, 2017.

[7] T. P. Szynalski. Tone generator. http://www.szynalski.com/tone-generator/, 2017.

[8] Wikipedia. Vocal range. https://en.wikipedia.org/wiki/Vocal_range, 2017.

Supplementary Material

Code: For information about source code, please email georgehe@stanford.edu
Presentation:
`https://docs.google.com/presentation/d/1Eo0MYF3nxlOz5Si8muJ2GjHWg9_iE2_nJrn7uWR42yQ/edit?usp=sharing`
Video:
`https://youtu.be/OfhyBYgzo8Y`