Python based Music Visualizer

Pratik Shrestha(08)

June 3, 2016

Contents

1 Introduction

Music visualization or music visualisation, a feature found in electronic music visualizers and media player software, generates animated imagery based on a piece of music. The imagery is usually generated and rendered in real time and in a way synchronized with the music as it is played.[1]

This project real-time python based music visualizer. Here, a music file is read and its information and frame rates are processed to generate a graphical animation. An amplitude graph is generated by drawing lines between amplitude read from the music file. The graph is scaled to the viewing co-ordinate at the center of the graphic interface and translated from both left to right and right to left according to the frame rate of the song creating a mirror effect of spike visualization. Furthermore, a circular effect with its radius and color depending upon the musics amplitude and frequency is also generated.

All of this is happening in real-time along with the music peice being played.

2 Libraries Used

2.1 Scipy

SciPy is a Python-based ecosystem of open-source software for mathematics, science, and engineering.[2] It is used to read amplitudes and frame rate of music files.

2.2 Numpy

NumPy is the fundamental package for scientific computing with Python.[3] It is used to generate frequency domain of the music peice.

2.3 Pygame

Pygame is a set of Python modules designed for writing games. Pygame adds functionality on top of the excellent SDL library.[4] Simple Direct Media(SDL) Layer is a cross-platform development library designed to provide low level access to audio, keyboard, mouse, joystick, and graphics hardware via OpenGL and Direct3D.[5] Pygame is used to generate the graphics interface. Geometric primitives for visualization effects such as lines, cricles and graph are also drawn using this library.

3 Source Code

```
from scipy.io.wavfile import read
from random import randint
from numpy import fft
import pygame, sys, time


def main():

    #graphic interface dimensions
    width, height = 420, 360
    center = [width/2, height/2]

    #read amplitude and frequency of music file with
       defined frame skips
    file_name = sys.argv[1]
    frame_rate, amplitude = read(file_name)
    frame_skip = 96
    amplitude = amplitude[:,0] + amplitude[:,1]
    amplitude = amplitude[::frame_skip]
    frequency = list(abs(fft.fft(amplitude)))

    #scale the amplitude to viewing co-ordinate of the
       frame height and translate it to height/2(central
       line)
    max_amplitude = max(amplitude)
    for i in range(len(amplitude)):
        amplitude[i] = float(amplitude[i])/max_amplitude*
           height/4 + height/2
    amplitude = [int(height/2)]*width + list(amplitude)

    #initiate graphic interface and play audio piece
    pygame.init()
    screen=pygame.display.set_mode([width, height])
    pygame.mixer.music.load(file_name)
    pygame.mixer.music.play()
    now = time.time()

    #visualizer animation starts here
    for i in range(len(amplitude[width:])):

        screen.fill([0, 0, 0])
        #circular animation: radius of circle depends on
           magnitude amplitude and color of circle depends
           on frequency
        try:
            pygame.draw.circle(screen, [(frequency[i]*2)
                %255, (frequency[i]*3)%255, (frequency[i]*5)
                %255], center, amplitude[i], 1)
        except ValueError:
            pass

        #the amplitude graph is being translated from both
           left and right creating a mirror effect
        prev_x, prev_y = 0, amplitude[i]
        for x, y in enumerate(amplitude[i+1:i+1+width
           ][::5]):
            pygame.draw.line(screen, [0, 255, 0], [prev_x
                *5, prev_y], [x*5, y], 1)
            pygame.draw.line(screen, [0, 255, 0], [(prev_x
                *5-width/2)*-1+width/2, prev_y], [(x*5-width
                /2)*-1+width/2, y], 1)
            prev_x, prev_y = x, y

        #time delay to control frame refresh rate
        while time.time()<now+ 1.0000000000/frame_rate*
           frame_skip:
            time.sleep(.00000000001)
        now = time.time()

        pygame.display.flip()


if __name__ == '__main__':
    main()
```

Review code in github at `https://github.com/prtx/Music-Visualizer-in-Python`**.**

4 Screenshots

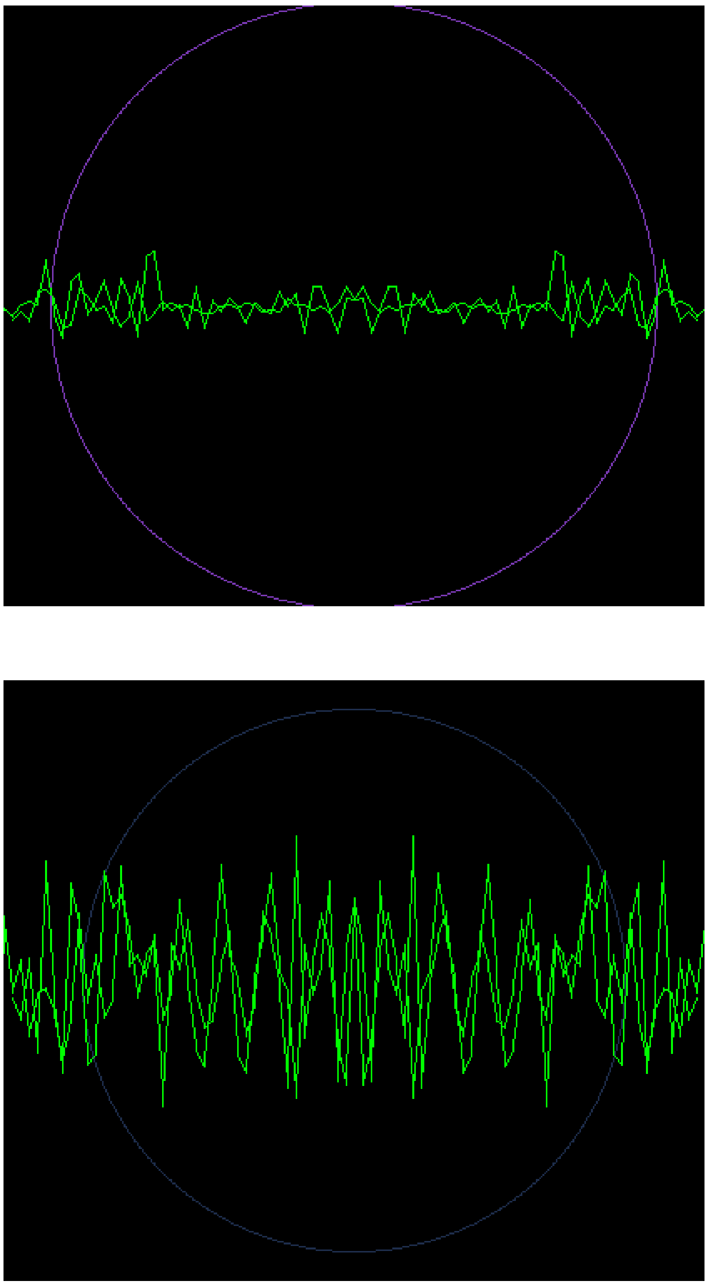

Figure 1: Program Screenshot

5 Conclusion

The project implemented basic shape drawing, scaling, tanslation, window-viewportco-ordinate transformations and frame refresh to generate a realtime animation of music visualization. The project exhibited the rules and essense of computer graphics.

6 References

1. https://en.wikipedia.org/wiki/Music_visualization[12/5/2016]
2. https://www.scipy.org/[28/5/2016]
3. http://www.numpy.org/[28/5/2016]
4. http://www.pygame.org/wiki/about[28/5/2016]
5. http://www.libsdl.org/[28/5/2016]