Customized Gaming Music

Nikunj Rajput, Vaishnavi Nandedkar

***Abstract*— In recent years, manipulation of music using audio processing techniques is one of the major factors in leading technology. This paper presents a technique to provide customization for the background music played in games, using time scaling technique.**

***Index Terms*— Audio processing, PyGame, Python, Text to speech, Time scaling**

I. INTRODUCTION

In today's world, everyone wants customization in their technology. We can change the interface of a cellphone the way we want it to be. With the same thought in mind we have developed a python script to manipulate the background music in any game based on the user's choice. To manipulate the music file, we have used one of the most interesting audio processing techniques, time scaling. The script uses PyGame library from python to build a basic game and a user interface to select the music file and time scaling factor.

II. DESIGN OVERVIEW

The main design techniques used for this design are time scaling and PyGame in python.

A. PyGame

PyGame is a set of modules in Python used to design games and different user interfaces. In this project PyGame is used to design a simple slider-cube game and a user interface. The user interface is shown in figure (1). As seen from the user interface the user can choose from tree different music files. The number of music files can be increased by making small changes in the python script.

There are 3 different time scales to manipulate the music files, from which the user can choose. The slow button represents time scale of 3, medium button represents to time scale 4 and fast button represents time scale 5. The user can select any three of these different combinations and our script will append these three music files with their requested timescales and that appended music file will get played as a background music for the slider and cube game.

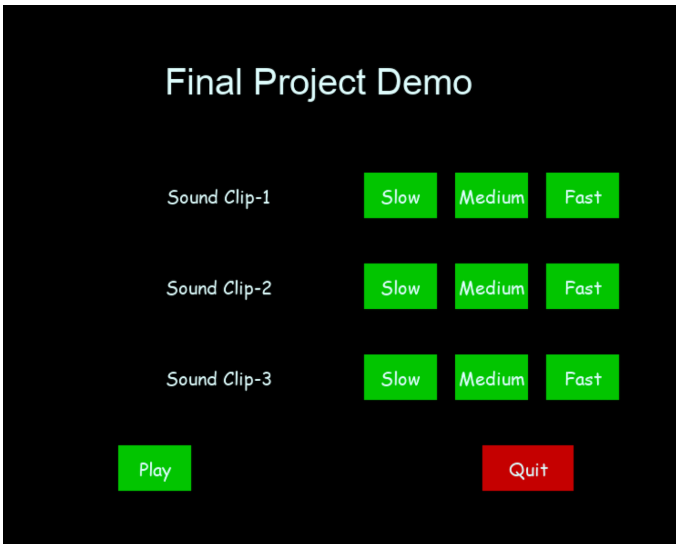

Figure 1. : User Interface

The game starts with minimum speed for the cube as well as the slider. Whenever the cube passes the slider, it is considered as a hit and after every two hits the level is changed (for demoing purpose the level was changed just after two hits). Figure(2) shows the interface of our game.

The PyGame works on event triggering and each time the game is called, each item is occurred with event. The functions for buttons in PyGame are defined in a way that each called function occurs only once and not every time in the event of game loop. There are functions for each button used to select the scale of the song and a check statement is given that it iterates only once in the event loop.

B. Time Scaling

Time scaling/ time stretching is a process in which the speed or duration of an audio signal is changed without affecting its pitch. For example time scaling should

sound like the music file is being played with higher speed or lower speed based on the time scale given, without any change in the tuning.

When a music signal is simply played fast, i.e. by lowering its sampling rate but by playing it at the same old sampling rate, the fundamental frequency of that

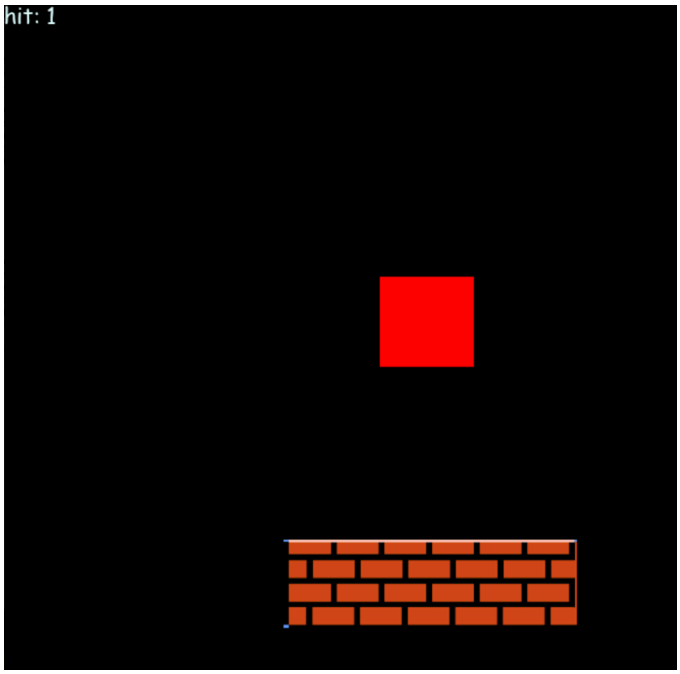

Figure 2. : Slider and Cube game

signal changes. This is against the concept of time scaling, which happens without changing the fundamental frequency (Pitch).

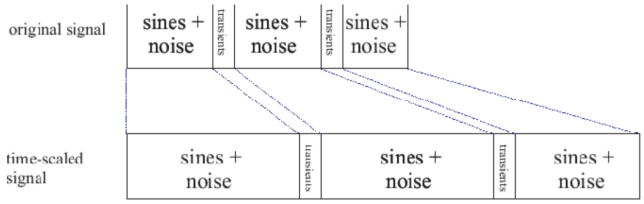

Figure 3.: Time scaling

Figure (3) shows the basic concept of time scaling. As shown in the figure the original signal is stretched in time scaling.

Implemented this time scaling technique in python using different libraries in python, such as Pylab, PyAudio, Struct, Scipy etc.

In time scaling, concept of STFT is used i.e. Short Time Fourier Transform which uses window size for overlapping of signals. In this, a window size is multiplied with each phase of the input signal of the block and an overlap is created between preceding and succeeding signal.

Short Time Fourier Transform Implementation:
The Short time Fourier Transform is given as:

$$\begin{aligned} X(\omega, m) &= \mathrm{STFT}\{x(n)\} \\ &:= \mathrm{DTFT}\{x(n+m)\, w(n)\} \\ &= \sum_{n=-\infty}^{\infty} x(n+m)\, w(n)\, e^{-j\omega n} \\ &= \sum_{n=0}^{R-1} x(n+m)\, w(n)\, e^{-j\omega n} \end{aligned}$$

In the above equation, the w(n) represents the window size which is used to multiply with input signal. Window function determines the block length of the signal. The window scaled input signal is passed through FFT(Fast Fourier transform) as in code snipped below. The phase difference of both the signals is considered, the one of initial block and the other for overlapping block. The phase difference is then integrated and passed through IIFT(Inverse Fourier Transform) to get the scaled output.

```
phase1 =  np.fft.fft(win*input_value[p1:p1+N])
phase2 =  np.fft.fft(win*input_value[p1+H:p1+N+H])
```

The output is saved as a wav file which is then loaded during then game event and is played at the background.

C. Libraries Used

Used IOS function “system” for text to speech conversion in the design. The used feature indicates whenever there is a hit or the level is changed.

Used Python time library to perform event driven PyGame functions.

Used Python random library to create a randomized position for the cube to start.

Used wave, PyAudio, Struct, numpy libraries to perform time scaling.

D. Conclusion

With the help of PyGame and Python created a game for which the background music and its time_scale is customized i.e. selected by the user.