Audio Visualizer using WebGL

CSE 418 (E2) - Mini Project

Faculty: Prof. Aju D

Submitted by:

K Harish

13BCE0164

Contents

S.No	Name	Page No.
1	Abstract	3
2	Introduction	3
3	Literature Survey	4
4	Proposed Methodology	4
5	Results	6
6	Conclusion	8
7	References	8

Abstract:

Audio visualizers are quite common in any music player. The primary function of any audio visualizer is to divide the audio data based on its frequency patterns and animate certain graphical items according to the frequency and amplitude change. There are various kinds of audio visualizers available. Out of them bar visualizer is quite common in any player. In this mini project, I developed a web application which can take any local audio file using drag and drop input. Once we dropped any audio file, audio will start playing along with 3d bar visualization. Used WebGL technology to simulate a 3D world with boxes and animated them based on the audio frequency. 3D Space is completely adjustable i.e we can use mouse to change the camera position at any time.

Introduction:

WebGL makes it possible to display amazing real-time 3D graphics in your browser but what many people don't know is that WebGL is actually a 2D API, not a 3D API. WebGL comes with almost all the latest browsers. There is no need of any specific software to install to run these applications. This cross-platform flexibility is one of the main reason why WebGL is popular. Even latest Android and iOS phones support WebGL. So, when it comes to end user he doesn't need to configure some complex parameters or install some extra libraries to run some small application. All he need to do is to open browser and enter the link of the website. This is the primary reason why I chose to use this technology for my Mini project. WebGL at the lowest level it doesn't have several built-in functions to work on. WebGL only cares about 2 things. Clipspace coordinates in 2D and colors. Developer should provide two shaders to provide WebGL with those two things. A Vertex shader which provides the clipspace coordinates and a fragment shader that provides the color. That will be a lot of work to write shaders from scratch and manipulate them. So, to make tasks simple there are various high level frameworks available which are written on WebGL. Some of the popular frameworks are given here: Three.js, Babylon.js, Turbulenz, PlayCanvas.js etc. For this project Three.js was chosen.

Another technology that is needed for the project is to analyse audio file and split them based on frequency. Web has evolved so much these days that all these technologies comes with browser. HTML5 provides Web Audio API which can handle any audio file and manipulate them

So, to accomplish our goal of visualization all we need to do is to use Web Audio API to manipulate the audio to frequency patterns and bind these frequency patterns to 3D objects created using Three.js and scale them accordingly.

Literature Survey:

Out of all the available frameworks for WebGL choosing the best one that suits given requirements is difficult task. The reason that Three.js suits best for the given project is Three.js uses all the three technologies Canvas, WebGL and SVG. When browser doesn't support WebGL it falls back to the other technologies which implies greater browser compatibility. And the other advantage of this library is it is lightweight. It also has default cameras, lights, and material shaders which can be used for demo purposes.

Documentation available in https://threejs.org/docs/index.html is descriptive and provides various examples demonstrating each and every feature. Another important feature of the Three.js is it provides geometry features where we can draw almost any 3D shape easily. Here in our project we need to draw cubes which can be very simple with Three.js. For our project, we draw n cubes aligned in a straight line. And along one axis these cubes are scaled to get bar effect. Considering all the requirements and features Three.js suit best for the project.

Proposed Methodology:

In our project the final output will have 3D bars visualizing based on the audio. And the background will have animating colors to give more immersive experience. All the important components of the project can be listed as follows:

1. 3D Scene with bars aligned in a line
2. A Camera whose position can be controlled
3. Audio analyser
4. Drag and drop handler
5. Bars animation handler
6. Animating background canvas

1. 3D Scene with bars aligned in a line:

To create a 3D scene with all the 3D components we are going to use Three.js framework which uses WebGL like we discussed before. Three.js has helper functions to create a scene. This function generates an empty scene with no objects. We should add light to the scene so that shaders can display objects or else the result will be just black. There are various types of lights available in Three.js. In our case, we used Hemisphere light which will be apt for given situation. Any light can do the job; it is just a personal choice of choosing which light looks good for our given 3D scene. We have 3D scene with the light but

whole 3D scene is just empty. So, to create bars we use geometry functions which create cubes. And we used random color generator which generates random colors and aligned all 60 cubes in a straight line. Each cube will have its own color. In the later section, we will see how we animate these cubes.

2. **A Camera whose position can be controlled:**

   Even though we added all the 3D objects, scene and light we won't be able view the result until and unless we add a camera. Three.js provides various types of camera. Here we used Perspective camera which will give a realistic look to the 3D scene. But we will notice that even after adding the camera we can't move freely in the 3D world. We will be restricted to a particular position. In order to orbit control over the 3D world there is a small library called OrbitControls.js which binds mouse movement with the 3D world movement. After binding these two, now we can freely move in the 3D world using mouse cursor.

3. **Audio analyser:**

   After the first two steps, we have our entire 3D scene is working properly. So, our WebGL work is done but we need to animate them according to the audio frequency. In order to achieve this, we first need to modify our input audio into array of frequency patterns. We don't need any extra framework to do this. HTML5 supports Web Audio API which contain analysers. We load the audio into to audio context and create a analyser. This analyser will stay in between source and destination of audio context. We will create an event handler i.e. whenever the audio in the audio context is processed we will divide the particular instant frequency bin and divide them into an array which has size as of number of bars in the 3D scene i.e. 60. Now the array contains frequency pattern at any instant which is distributed in an array. Only thing that is left is to animate them

4. **Drag and drop handler:**

   We have our audio analyser working like expected but there is no way to input the audio into the application. We can either use file input handler but that is old technology. We got used with drag and drop. So, we create a drag and drop handler where we can drop the audio file anywhere on the page and that audio file will be automatically loaded and audio will start playing. This handler should pass the audio file to audio context which we created in the previous step.

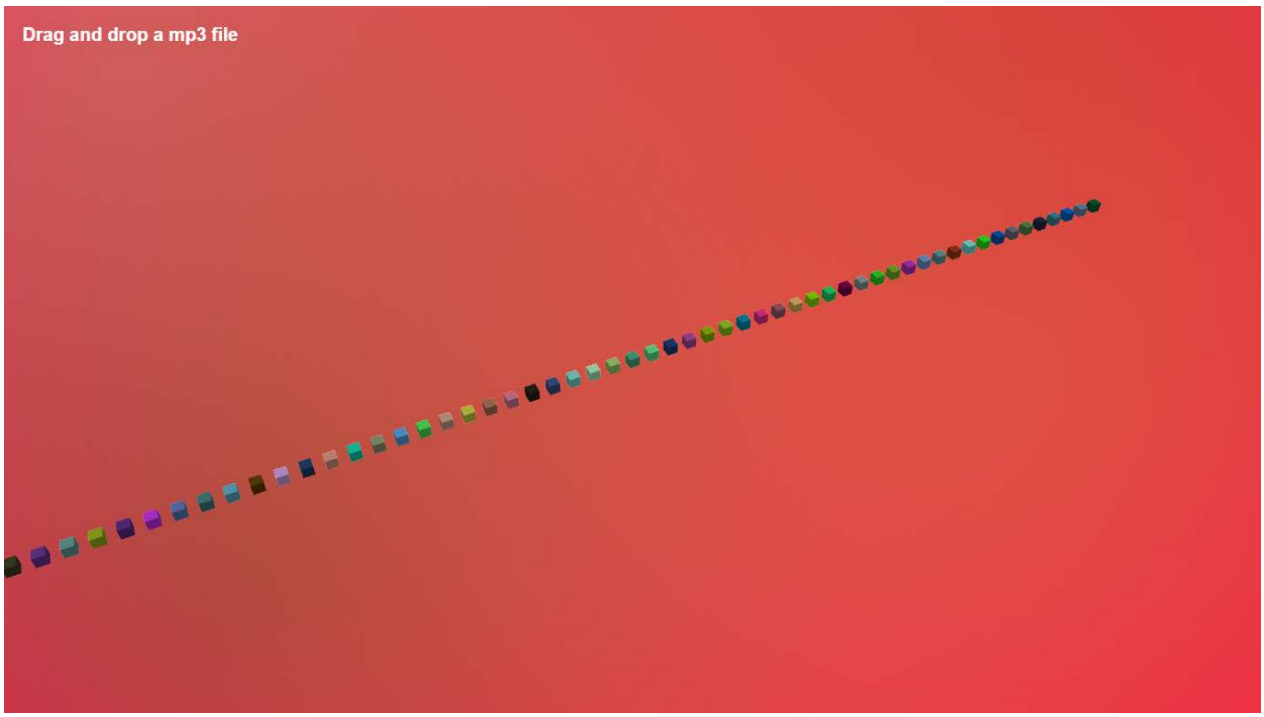
5. **Bars animation handler:**

   We need to map the cubes scaling with the frequency array which will be updated for every new audio processing. All we need to do is to scale our cubes in the event handler itself. So, whenever audio is processed it generates a new frequency bin and it will be distributed to an array. Array will contain all integers and we can use this as scaling factor and scale the cube on opposite sides generating animating bars. Once we add this animation handler dropping any audio file into the application will show us a visualization.

6. **Animating background canvas:**

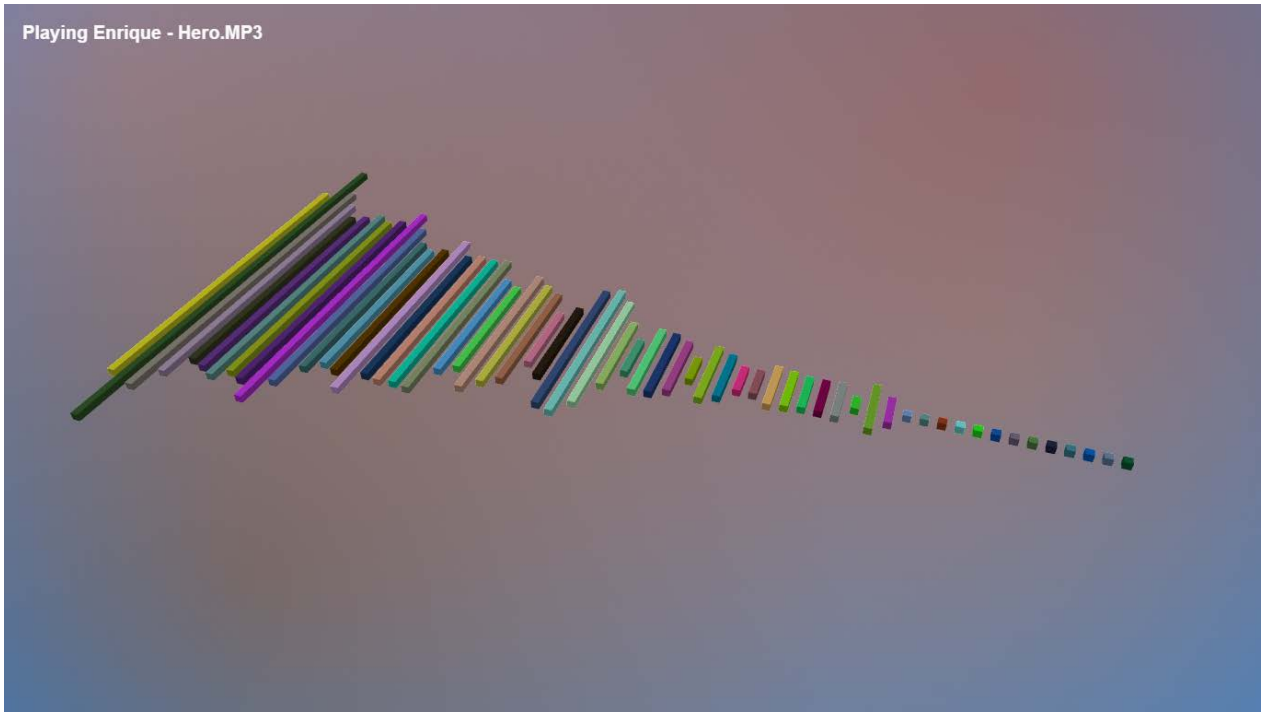
   Application works as expected after the above five steps. This section gives more immersive experience by creating animating background colors using canvas. We should make sure that our 3D scene is transparent to have our custom background. We create a canvas and using CSS animations we animate the background colors continuously making it look more elegant.

Result:



Before adding audio file (Bars at default size)

After adding audio (View -1)



After adding audio (View - 2)

Conclusion:

With the help of Three.js and WebGL technology we made a fully working audio visualizer completely in web browser. Demo can be checked by just opening the index.html in any latest browser and add an audio file. This application can also be extended by adding multiple visualization options and all player features can also be included to make it fully functional audio player.

References:

- https://www.html5rocks.com/en/tutorials/webgl/webgl_fundamentals/
- https://www.html5rocks.com/en/tutorials/webaudio/intro/
- https://threejs.org/docs/index.html#Manual/Introduction/Creating_a_scene
- http://srchea.com/experimenting-with-web-audio-api-three-js-webgl
- https://threejs.org/docs/index.html#Reference/Materials/MeshPhongMaterial
- https://www.chromeexperiments.com/
- https://github.com/mattdesl/three-orbit-controls