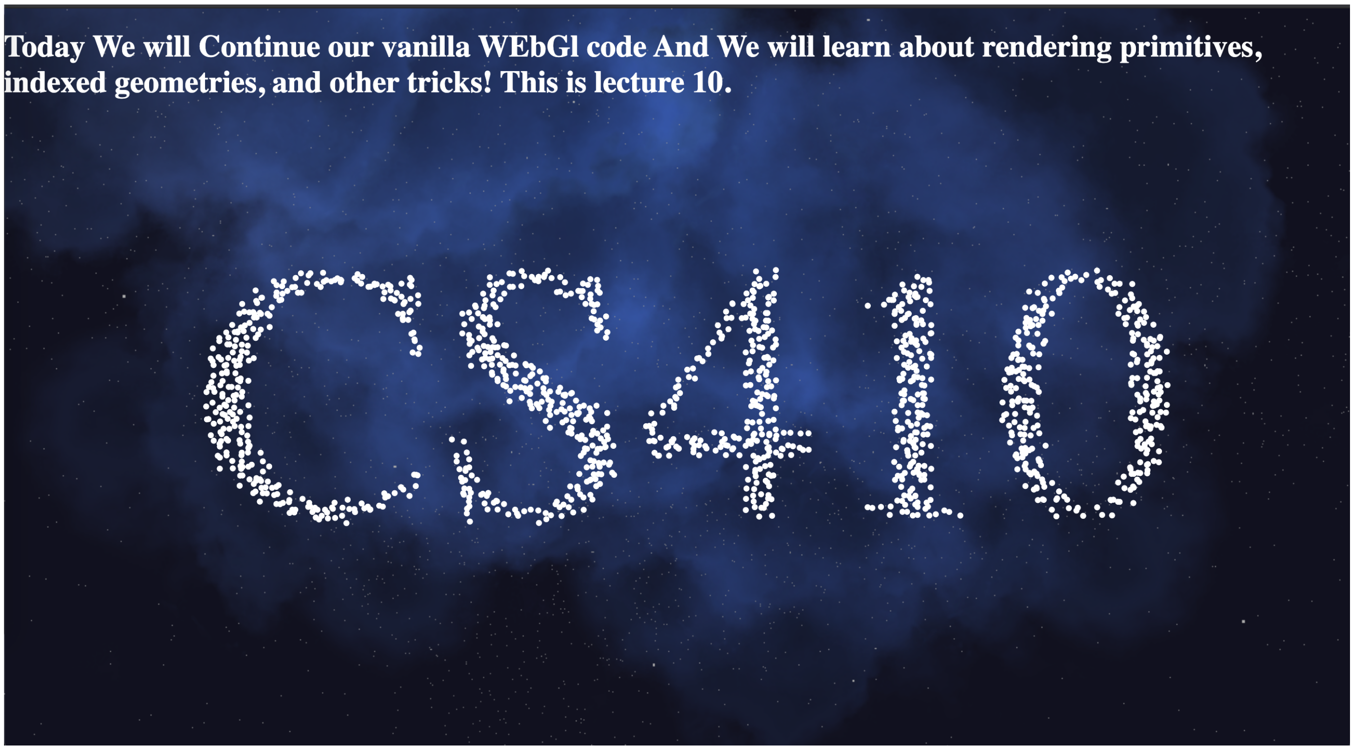
CS410 Intro

Saurav Paudyal
saurav.paudyal001@umb.edu
University of Massachusetts Boston



Today We will Continue our vanilla WEbGl code And We will learn about rendering primitives, indexed geometries, and other tricks! This is lecture 10.

Figure 1: CS410 Intro Page.

ABSTRACT

Professor Daniel Haehn needs an intro for the CS410 Software Engineering course (CS410.net). This project involved building a website that could detect silence on the voice over sound files, using Web Audio API. And also includes various webgl techniques to accomplish projects goals.

Despite all the difficulties, the project exemplifies the use of silence detection to demonstrate the potential of the program. And combined with other WebGl skills, we can produce prominent results.

KEYWORDS

WebGL, Visualization, Three.js, Blotter.js, CSS

ACM Reference Format:
Saurav Paudyal. 2019. CS410 Intro. In *CS460: Computer Graphics at UMass Boston, Fall 2019.* Boston, MA, USA, 2 pages. https://CS460.org

1 INTRODUCTION

Saurav Paudyal, Computer Science Major. https://cv14.github.io/cs460student/finals/index.html

This project is important me because :
1) Can be used in real life.
2) Its a real Contribution in the real world not just a homework assignment.
3) Another learning opportunity.
4) Can show-off what we learned so far.

2 RELATED WORK

1)https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API*
2)https://dzone.com/articles/exploring-html5-web-audio
3)https://blotter.js.org/
4)https://threejs.org/
5)https://www.youtube.com/watch?v=bYqW6fehI7s
6)https://www.youtube.com/watch?v=y-sqPKhMa8E
7)https://www.youtube.com/watch?v=1bkibGIG8i0
8)https://redstapler.co/three-js-realistic-rain-tutorial/
9)https://cv14.github.io/cs460student/05/index.html

3 METHOD

So I had to make a website for the class cs410 in the coming semester. The website needs to be able to run WebGL animations. Play the intro music and voice over. Display the text for the voice-over. Take the intro voice-over text as a form of Uniform Resource Identifier. Synchronize it with the voice-over sound file and display it. Add other WebGl elements.

3.1 Implementation

First I started working on the sound detecting part of the project. In order to do that I used the Web Audio Api. I created a new AudioContext. Then I created the method setupAudioNodes, which creates a BufferSource audio node and connects it to the destination. The loadSound function shows how we can load an audio file. The buffer which is passed into the playSound function contains decoded audio that can be used by the web audio API.

Once we turn the source node on the file plays. Then I created an analyzer node: With this node we get realtime information about the data that is processed. This data we use to determine the signal strength. Then I created a javascript node: We use this node as a detector to detect the volume with new information. Then I connected everything together. The audiosource creates a signal based on the buffered audio. This signal is sent to the splitter, who splits the signal into a left and right stream. Each of these two streams is processed by their own realtime analyser. From the javascript node, we now get the information from both analysers, providing us the current volume at each time in milliseconds.

Then I check if the average volume is below some threshold value, we print the voice-over text on the screen.

```
//threshold value
        if(average < 10){
          if(n > res.length-1){return;}
          //print out the average volume on console
          console.log("First Channel : " + average);
         //print out the time that volume went down.
          console.log(context.currentTime);
          st = st + res[n++] + " " ;
      document.getElementById("r3").innerHTML="<h1>"+st+"</h1>";
        }
```

To take input as a form of Uniform Resource Identifier. I did :

```
        //get input as web param.
     var param = window.location.search.substring(1);
        //console.log(param);
        var res = param.split("%20");
```

Here I got the input as URI and stored it in res array for displaying it later on.

And as for the WebGl part, I used Three.js and Blotter.js to do most of the work. I used Three.js to create a raining like effect background. Also tutoriols online and on https://redstapler.co/three-js-realistic-rain-tutorial/
helped me achieve it. I used Blotter.js to create the "CS410" text effect.

3.2 Milestones

How did you structure the development?

3.2.1 Milestone 1. Brainstormed different designs and ways to approach the problem.

3.2.2 Milestone 2. Confirmed the deisgns and approaches on how to start the backend side of the project.

3.2.3 Milestone 3. Detected Silence but was not able to figure out the threshold perfectly.

3.2.4 Milestone 4. Displayed text on every silence part.

3.2.5 Milestone 5. Made flying CS410 text effect using Blotter.js

3.2.6 Milestone 6. Made storm background using Three.js

3.2.7 Milestone 7. Combined everything together.

3.3 Challenges

Describe the challenges you faced.

- Challenge 1: Threshold value is still not right and if loaded different voice over file is not able to synchronize the text accordingly. Need to make new voice-over files with a bit longer silences.
- Challenge 2: Tried several other libraries but was not able to work it with JavaScript.
- Challenge 3: Lots of time consuming because it is my first time doing html and Webgl stuff took a lot of time. Had to watch lots of youtube and other webpages.

4 RESULTS

My final result was not something I expected, but I got close to what I wanted to achieve. Due to shortage in time I was not able to make it completely working.

5 CONCLUSIONS

Overall it was great project to work on and to learn from. Due to time shortage was not able to complete the report and some parts of the program. Below are the refrences that I used.

5.0.1 Milestone 6.