Gustavo Costa - Technology Design

Project Summary

My Technology Design project is a multi-user music making web application called **TETRAD**. The project involves users navigating to the website, located at **tetradapp.com**, from any device, be it desktop or mobile device, and be able to play music in sync with all other users connected.

Stumbles, Subsequent Victories, Breakthroughs

The first major hurdle I encountered in building this project was was getting any devices that were currently connected to the website to play audio in sync with each other. To tackle this problem I researched Websockets and how they establish a stream of data between the client and the server in order to create a real-time connection between client and server. I determined that a typical HTTP request would not work to enable all users to stay in sync with each other. I realized that websockets were the key to getting my project to function as I envisioned.

However before arriving at websockets I tried using Meteor.js because it has a reactive UI built in, making it possible to build real-time applications. I believe that 'under the hood', so to speak, Meteor is not using websocket technology by default, rather I believe it works similar to React or Angular, it simulates changes on the client side before it gets a response from the server. The learning curve was a little higher for Meteor than Node/Socket.io so I determined that for the project to stay on track I would have to pick the technology I thought I could learn the fastest. Subsequently, this led to me studying node and understanding it's fundamentals.

Another difficulty with this project was the music itself. Initially I experimented with playing audio samples (buffered files) however I quickly discovered that it was challenging to make the devices playin sync. Often the devices were out of sync by mere milliseconds but in music that is a noticeable difference. I worked around this problem by using a JavaScript library that allows a developer to easily create synthesizers and effects in the browser. This library is called Tone.js. Once I changed from the process of playing an audio file to executing code, I mostly fixed my problems with synchronization.

From this stage onward I spent the majority of my time tinkering with the tones and notes of the music. This was a long process because certain timbres of synthesizer would sound excellent and clear on desktop clients but distorted and glitchy on mobile clients. For instance certain effects such as Reverb couldn't effectively be done on mobile due to the noise that was added as soon as those effects were added. Other effects such as Chorus or Delay were not distorted however they impacted performance much more heavily and affected synchronization. In addition to the effects and timbres of the music, the pitch was also an important factor. I quickly realized that certain frequencies, particularly low frequencies, just couldn't be reproduced through phone speakers. To get around this I researched some more concrete ways

to detect mobile devices other than screen dimensions. I discovered that in JavaScript you can detect if the client is a mobile browser with statements such as this:
`navigator.userAgent.match(/Android/i)`

Another stumble I had in completing this project was that I had never written server-side JavaScript before. As a result I spent about two weeks studying Node.js and how create a JS server before I could really understand any of the Websocket code I encountered online. Reading up on Node enabled to simply look at Socket.io's Chat example and easily adapt it to my needs for this project. Understanding the fundamentals of node made using technologies such as Express, Socket.io, and Meteor much easier.

Once I understood how to implement Socket.io into my Node Server it was a simple matter of emitting events on the client and the server and listening for those events on the client and server. For example:

```
if ( currentBeat == 1 ){
        io.emit('sync');
}
```

This is a barebones example but nonetheless it illustrates the core function of the server in this project. The server keeps the beat, and on the first beat of every measure it emits a signal to sync all the clients that are connected. If the client is not currently playing music the syn signal becomes a signal to begin playing music.

Fortunately, this project had minimal UI elements, as such they were a non-issue. I tried to incorporate a playful animation when the page loads as a hook to get people interested in what would happen next.

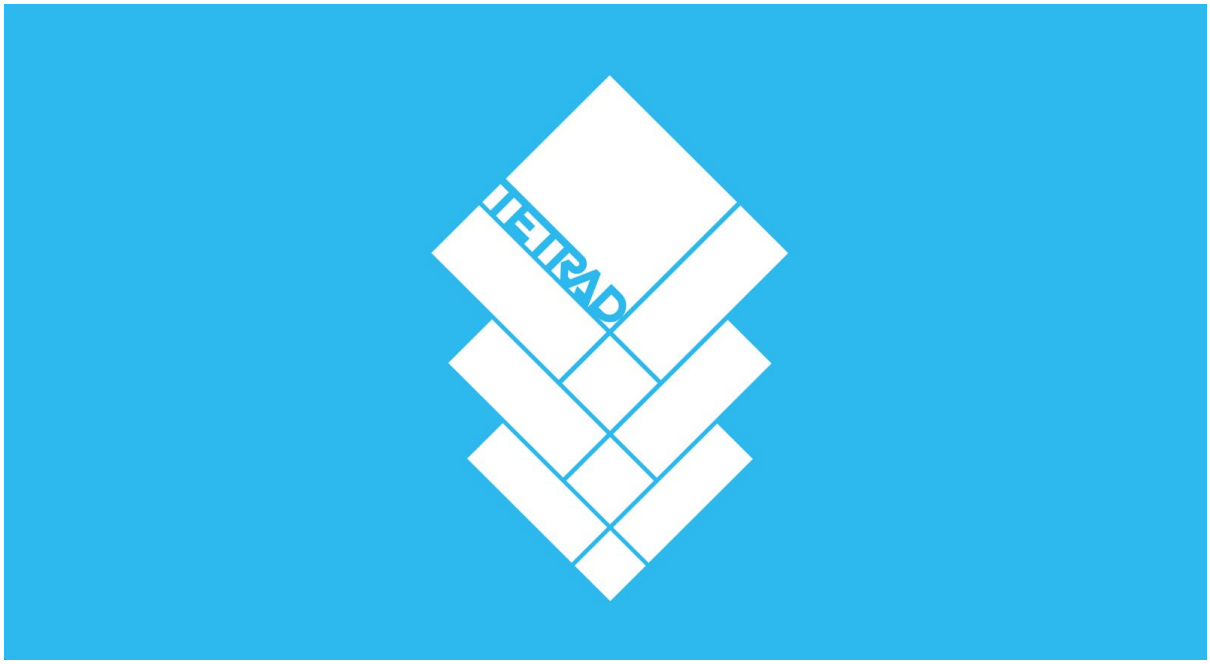
Technologies:

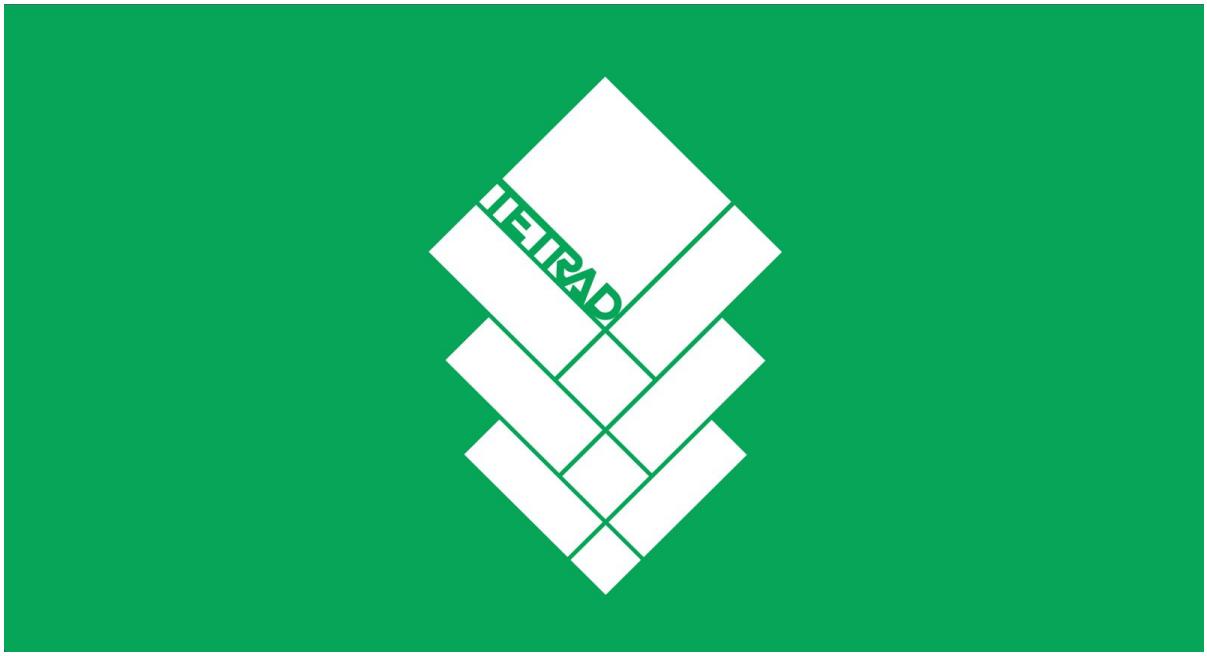
- HTML/CSS/JS
- Node.js
- Socket.io (Websockets)
- Express.js
- jQuery
    - RandomColour.js
    - Lettering.js
    - Tween.js
- Tone.js

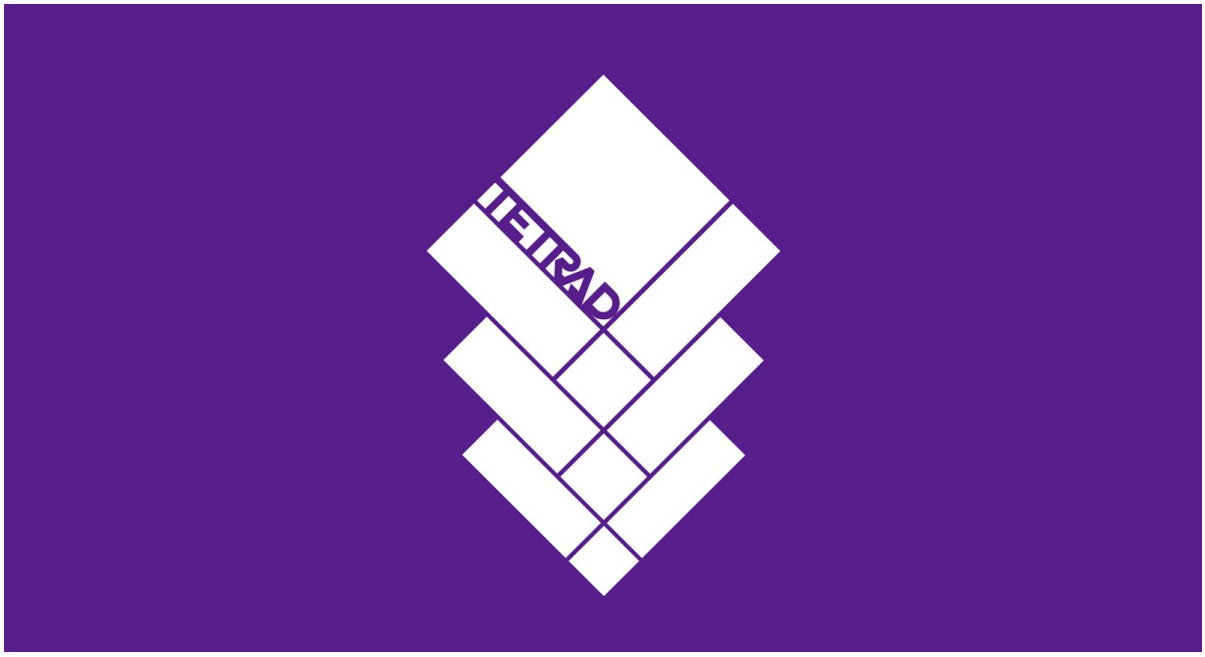
Links:

- Github
  - https://github.com/gustavo-costa/tetrad
- Vimeo
  - https://vimeo.com/191328518

Images







MAKE

MUSIC

TOGETHER

BEGIN