HAW Hamburg Prof. Dr. Andreas Plaß

AVPRG Jakob Sudau

# VDrums Technical Documentation

| | | |
|---|---|---|
| Niklas Bartsch | niklas.bartsch@haw-hamburg.de | 2216285 |
| Simón Hoyos Cadavid | simon.hoyoscadavid@haw-hamburg.de | 2319104 |

# 1. Introduction.

During the class of Audio and Video programming in the University of applied sciences in Hamburg, the students were in charge to create a web application that utilized the JavaScript Audio API. The VDrums project was born with this preposition in mind. The core emphasis of VDrums is to manifest the feelings and emotion one feels while playing the drums, into something more than simple sounds, more than music: Image. The fact that one feels through more than one channel indicates that the human body is able to experience everything through something different apart from just hearing; in this case, the project is trying to bring the music to life in a way that it integrates the feeling of listening to your own creation, but also looking and being able to visually appreciate it.

VDrums is a web application that lets one play an electronic drum set, while seeing the in real-time visualization of it. Considering the functionality of VDrums it is obvious to which group of people it would appeal, and these are drummers who would like to enhance their playing experience to something more than just music. Nevertheless, this project does not have the intention of being used as a device to create music videos, nor something similar; it is planned for the person who uses it to have fun, which in turn also adds more people to the ideal group of usage: bringing in anyone who wants to get into playing drums but doesn't have the experience. Furthermore, the fact of images being involved means that more visually perceptive individuals might be more inclined to playing the drums, whereas they could've not done it otherwise. In conclusion, anyone with any type of interest in music would be inclined to test VDrums, even if they don't play instruments themselves.

This document will be in charge to explain the technicalities that were involve in the VDrums project, providing both a useful guide for any user trying to understand the usage of the program, as well as all the more technical bits and pieces that tie the program together.

# 2. Technical information.

## 2.1. Environment.

One of the most important aspects to be cleared up is to answer the question, of what the environment was, in which the project was developed. This section will be divided in three subsections: *JavaScript*, *three.js* and the JS[1] Audio API[2].

2.1.1. *JavaScript:* JavaScript, or JS for short, is an interpreted programming language that is part of the ECMAScript specification[3]. JS is compiled in

[1] JS: JavaScript.

[2] API: Application programming interface.

[3] *JavaScript. From Wikipedia the Free Encyclopedia. https://en.wikipedia.org/wiki/JavaScript [08.01.19]*

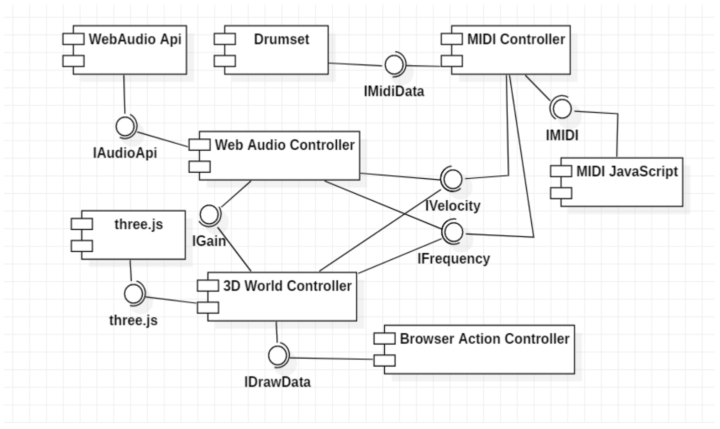
a browser, using a HTML5 document to be executed, meaning it can only be used in a web scenario (Unless there is external help of an environment such as Node.js, which will not be explained in this description). It has a wide variety of open source libraries and APIs that aid with the development of different kinds of programs. It was stated by the professor that JS had to be used in the creation of this project, being strongest reason for being chosen.

2.1.2. *Three.js:* This is an open source library created with the sole purpose of creating a 3D world in a 2D environment (Screen). It was used to help with the creation of all the visuals and representation.

2.1.3. Web Audio API: This interface helps with playing, as well as capturing, sounds in a JS application. It was used in order to connect the MIDI[4] information sent from the drum set to the browser, interpreting the different signals.

The aforementioned were used in order to create a cohesive project with the clear intention of displaying the drum set sounds in a purely web environment. It was tested primarily in Google Chrome.

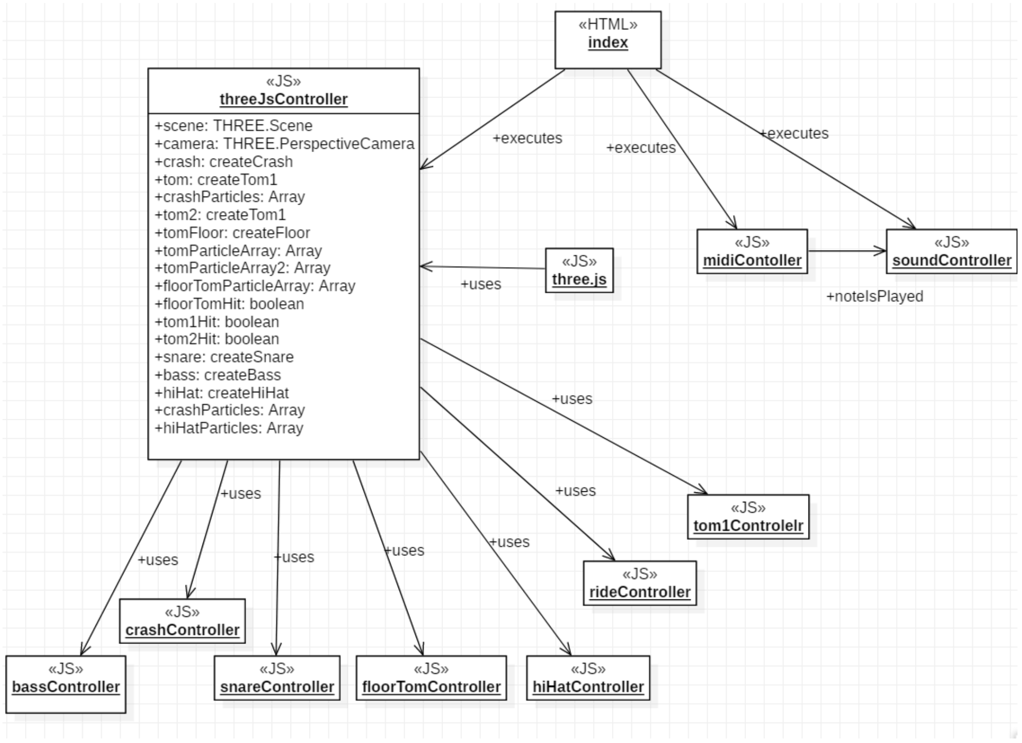
## 2.2. Software architecture.



While thinking about the structure and architecture that the system would have, the accompanying UML component diagram was developed. Furthermore, into the inner workings of each of the components, it is clear that there are three main software mechanisms: The MIDI Controller, the Web Audio Controller and the 3D World Controller. The first of the previously mentioned controllers implements the Audio API to read the MIDI signal sent by the drum set, this information is then sent to the other two to be processed. The Web Audio Controller is in charge of understanding these MIDI signals while playing the determined sound, parallel to it, the 3D World Controller is in charge of also interpreting these same signals but drawing and modifying a digital 3D world instead. The last-mentioned controller also implements the three.js JS library, which helps with the displaying of three-dimensional models in a two-dimensional space.

[4] MIDI: Musical Instrument Digital Interface.

The next stage of the development was making a cohesive, descriptive structure in addition to the generalized vision of a component diagram. From this premise the following object diagram was created:



This diagram does not have a complete description of each of the variables and functions of all of the objects, but it depicts mostly the threeJsController (Which is a part of the 3D World Controller Component). This controller was in charge of the core idea of the project: visualizing the whole 3D world. Inside of it, functions of each one of the underling objects were called in order to create the objects in the scene, which was declared, initialized and updated in the threeJsController[5]. As a result of this, the objects in charge of each one of the individual drum set pieces, such as bassController or crashController, had the whole purpose of returning or updating three.js objects, having a minimal impact on the scene persé.

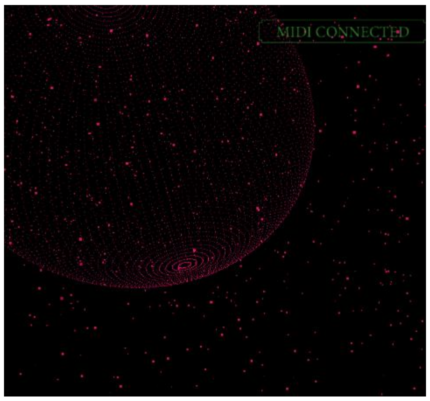
Last but not least, the midiController and soundController were the connection of the program to the outside world. In addition, they were deeply connected with one another, because the first was in charge of reading the notes, while the second one was in charge of playing them.

[5] With the functions:
- init(): Calls the functions in the part controllers to create objects.
- renderScene(): Is called 60 times per second and uses the functions in the part controllers to update the 3D objects.

### 2.3. Software description.

To begin with the software description, it is important to understand some of the problems that were presented during the development. First of all, how is the sound, that comes out of the computer, represented? This question does not answer itself in the most obvious manner, because the initial idea was to create the corresponding notes utilizing the oscillator node that JS has in its Audio API. Even though that was the way that the program worked at the beginning, the short amount of time and shear complexity of the premise of modifying the sound waves to better accommodate a drum or cymbal sound was too much to have a feasible finished project. Many professionals still work in the field of making digitally created waves sound like real-life instruments, which demonstrates the difficulty of the previous problem. Nonetheless, the program did work when playing the pure note (A hundred percent pure sine curve), even if the sound wasn't ideal, and traces of this are still to be seen in the program[6]. In order to solve this problem, the developers of VDrums decided to use prerecorded open source sounds that were played the instant each piece of the drum set was struck, taking in also all the MIDI information sent from the drum set[7], and recognizing each part of the drum set with a different note – for example 36 for the bass drum or 38 for the snare.

Secondly, the different controllers for all the parts of the drums set were then composed of several different functions that created either a mesh, or a collection of particles (or both in the case of the ride). Most particles would then occupy a space in a 3D object, acting as the object vertices, and since most of these meshes had only blank spaces between each point, what was left was an amalgamation of particles in the form of several different geometrical figures, that in itself helped as an abstract representation of the human psyche of the things we listen to while we are playing. Furthermore, some of the geometrical figures were also accompanied by several particle arrays that were generated when the object was struck, to distinguish the more explosive sounds to the contained ones. These digital representations of the parts of a drum set were then made to react to being struck, meaning they all react to how strong of a hit was made and how fast (Using the velocity described in the MIDI note).



## 3. Instructions of usage.

In order to make the use of our web interface as accessible as possible it was decided to just stay with the concept of "Plug and Play" making it easy for everyone to use it without having to install drivers or settings. Nevertheless, precautions have to be taken before one can start enjoying the visuals and playing the drums. The hardware has to be right first: All you need is an e-drum set

[6] For more information: The "soundController" file inside the folder "js" is the clear example of this.

[7] The Alesis Nitro set (Which was used for this project) sends all of the MIDI information through the channel 10 (Note on: MIDI[0] = 153; Note off: MIDI[1] = 0).

with either a USB – B, or a MIDI, connection and a cable with one of the previously mentioned ends with a normal USB – A on the other end. Once the drums are connected, having opened and loaded the web interface, one can start playing.

As soon as one opens the Interface it will be noticeable that the whole screen is black. It is supposed to be this way. Only when one starts playing the drums, one’ll see and hear the different visualizations and sounds of the drums. Make sure the sound of the laptop or computer is turned on and set to room volume.