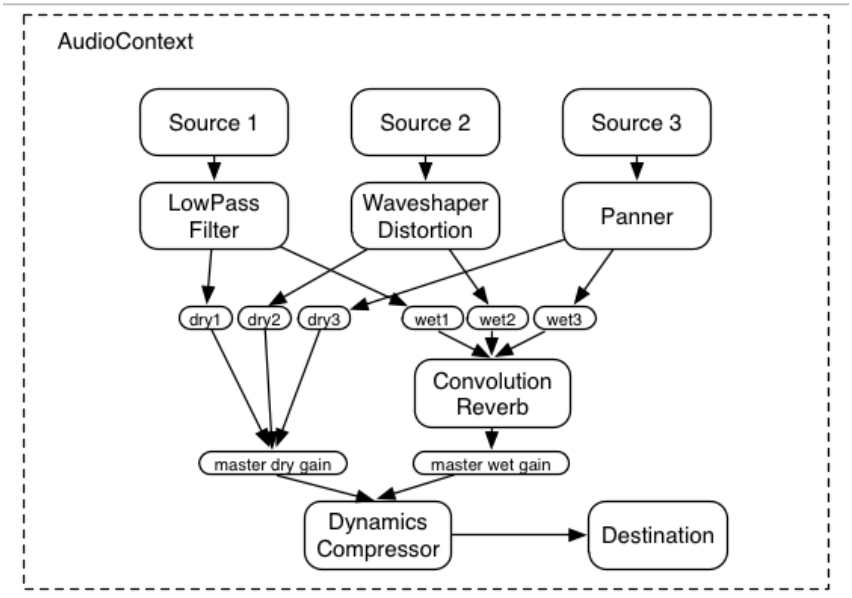
Technische Beschreibung:

Für mein QPT1 wollte ich mich mit der noch relativ neuen Web Audio API befasst.Um eine Struktur in meine Application zu bringen, habe ich mithilfe der Bibliother require.js zusammengehörigen Programmcode in einzelne Dateien aufgeteilt. Grundsätzlich kann man sagen, dass sich mein Projekt in drei unterschiedliche Teilgebiete aufteilen lässt (Audio, Visuals, AudioVisuals), welche ich gerne erklären möchte.

Audio:

Dieser Bereich befasst sich im allgemeinen mit dem Erstellen und Abspielen von Audio Samples. Grundsätzlich muss man sagen, dass die Web Audio API nach einem Node Graph aufgebaut ist. Das bedeutet, die unterschiedlichsten Knoten wie Oszillatoren, Effekte, Kompressoren,... beliebig miteinander verbunden werden können. Mithilfe dieser Funktion kann man auch sehr komplexe Sounds erstellen und auch unterschiedlichsten Audio Inhalte zusammen fügen. Das ist insbesondere praktisch, wenn man z.B. für alle derzeit abgespielten Sounds einen Master Volume Regler einbauen möchte. So muss man lediglich alle Sounds zu diesem Master verbinden. Die nachstehende Abbildung sollte den Aufbau verdeutlichen:



Die Webaudio API bietet neben dem abspielen von Audio Dateien auch die Möglichkeit Sounds

selbst zu erstellen und so direkt auf den Sound Buffer zuzugreifen. Für mein QPT habe ich mich deshalb dazu entschlossen alle Sounds Mathematisch zu berechnen und im Anschluss abzuspielen. Es hätte genauso die Möglichkeit gegeben, alle Sounds mithilfe eines Oszillators zu erstellen, jedoch ist man dort auf vier Wellenformen (Sinus, Quadrat, Sägezahn, Dreieck) beschränkt.
Um das erstellen des Audio Contexts und das abspielen der Sounds kümmert sich die Datei audio.js. In dieser befindet sich auch die Logik für das abspielen der unterschiedlichen Sounds. In den Dateien welche sich im Order audioEFX befinden sind alle Dateien die Sounds erstellen. Nachdem ein Audio Source Node nur einmal abgespielt werden kann, wird im ersten Schritt ein Buffer für alle Verfügbaren Noten einer Wellenform erstellt. Dieser Buffer repräsentiert, die Welle welche abgespielt werden sollte. Im Zweiten Schritt (function getSample()) wird aus diesem Buffer ein spielbarer Audio Source Node der den vorher erstellten Buffer zugewiesen wird. Das hat den Vorteil, dass der Buffer mit 48000 Samples/s nicht bei jedem Abspielen erneut erstellt werden muss.

Audio Visuals

Neben dem Abspielen von Normalen Samples bietet die Webaudio API auch die Möglichkeit auf Frequenz Daten des der gespielten Musik zuzugreifen. Diese Daten habe ich verwendet um ein Partikelsystem mithilfe des 3d Frameworks three.js zu implementieren. Die Partikel werden zufällig im Raum angeordnet und bewegen sich auf einer Zufällig berechneten Bahn im Raum.

Visuals

Dieser Bereich meines Projektes beschreibt die gesamte 2d Grafik meines Projekts. Zum einen wird das Grid, zum ein/ausschalten der unterschiedlichen Sounds gezeichnet und zum anderen der Maus Pointer. Weitere Grafiken können via Module einfach hinzugefügt und wieder gelöscht werden.

Verwendete Frameworks/Libaries

- Three.js (3d Framework)
- require.js (Libary zur Strukturierung von Dateien in Javascript Projekten)
- modernizr.js (HTML5 Featur Detection)
- spin.js (Loading Animationon written in Javascript)