**Belegarbeit: Internettechnologien**

Erstellung einer Musikvisualisierungsanwendung in JavaScript unter Nutzung der Three.js Bibliothek

**Vorgelegt am:**

**Von:**  Josia Rudolph, Martin Görner

**Studiengang:** Technische Informatik

**Studienrichtung:** Technik

**Matrikelnummer:** s4004602,

**Gutachter:** Prof. Dr. Mathias Sporer (Staatliche Studienakademie Glauchau)

Inhaltsverzeichnis

[**Inhaltsverzeichnis II**](#_Toc150887892)

[**Abbildungsverzeichnis III**](#_Toc150887893)

[**Abkürzungsverzeichnis IV**](#_Toc150887894)

[**1. Zielsetzung 1**](#_Toc150887895)

[**2. Grundlagen 2**](#_Toc150887896)

[2.1. Programmierumgebung 2](#_Toc150887897)

[2.2. Das Three.js-Modul 2](#_Toc150887898)

[2.2.1. WebGL 2](#_Toc150887899)

[2.2.2. Three.js 2](#_Toc150887900)

[2.3. Programmierstil 2](#_Toc150887901)

[**3. Evaluierung 3**](#_Toc150887902)

[3.1. Visualisierungskonzept 3](#_Toc150887903)

[3.2. Umsetzung in Typescript 3](#_Toc150887904)

[3.3. Installationshinweis 6](#_Toc150887905)

[3.4. Musikvisualisierer 6](#_Toc150887906)

[**4. Fazit 7**](#_Toc150887907)

Abbildungsverzeichnis

[**Abbildung 1,** Verschachtelungs-Skizze der Programmcodedateien 4](#_Toc150887951)

[**Abbildung 2,** Klassendiagramm Typescript Quellcode 6](#_Toc150887952)

Abkürzungsverzeichnis

HTML Hyper-Text-Markup-Language

API Application Programming Interface

UML Unified Modelling Language

1. Zielsetzung

Musikvisualisierungsapp mit javascript

Musik abspielen

Coole Animationen auf Musik reagierend

Mit Three.js auseinandersetzten

Konzept Entwicklung

Plattformübergreifend und auf jedem Browser nutzbar

1. Grundlagen
   1. Programmierumgebung

Node.js: (npm Paketmanager und Liveserver/Runtime Environment) -> Laufzeitumgebung und LiveServer, Ausführung von JS ohne Browser auf Server oder Befehlszeile; kommt mit Paketmanager; verarbeitung großer datenmengen durch skalierbarkeit dank Asynchronem Verhalten -> für effiziente und Leistungsfähige Platformen, Nachteil: muss installiert werden, Laufzeitumgebung mit Compiler bzw. Transpier für typescript in JavaScript

Installation der benötigten Module und Bibliotheken per Befehlszeile

Ausführung der Kompilierung und Liveserver

* 1. Das Three.js-Modul
     1. WebGL

Effektiver grphik renderer

* + 1. Three.js

Modul für js, das unteranderem webgl nutzt/vereinfacht

Wie funktioniert Musik in Three.js: Laden (THREE.AudioLoader), abspielen (THREE.Audio), -> über THREE.AudioListener (wie Mikro) dann hörbar

Möglichkeiten Musik zu analysieren

Was ist gleich nochmal fft?

* 1. Programmierstil

Typescript, was das

Vorteil gegenüber normalen JS

Was bringt Typescript in Bezug auf OOP zusätzlich mit sich

1. Evaluierung
   1. Visualisierungskonzept

Die Visualisierung zielt darauf ab, die Laustärke der analysierten Frequenzbereiche durch die Bewegung und Verfärbung von Objekten darzustellen.

Die Darstellung der Musikvisualisierung ist in 4 Einzelbereiche unterteilt.

Der Hintergrund wird von einem Balkendiagramm ausgefüllt. Die Aussteuerung der Balken entspricht der Lautstärke der analysierten Frequenzbereiche.

Im Vordergrund ist die Visualisierung in drei Frequenzbereiche aufgeteilt für die Darstellung der Bässe, der Mitten und der Höhen.

In einem Kreis um den Mittelpunkt herum befinden sich Objekte, welche sich, nach der Lautstärke der Frequenzen im Bassbereich richtung Mitte bewegen. Diese sollen sich je lauter sie sind, umso röter verfärben.

Aus dem Nullpunkt der Szene heraus entspringen Objekte, welche nach den analysierten Lautstäken des mittleren Frequenzbereiches animiert sind. Sie verteilen sich in einem Kreis um den Mittelpunkt. Die Austeuerung nach außen soll proportional zur Laustärke sein. Die roten, grünen und blauen Farbanteile verändern sich nach Frequenz und Lautstärke.

Von dem äußeren Bildschirmrand soll Objekte in Spiralförmigen Mustern nach inne Pulsiert kommen. Diese Animation wird durch die Lautstärken der hohen Frequenzanteile gesteuert. Für diesen Bereich sollen hellere Farben verwendet werden.

Die Visualisierungen der drei Frequenzanteile sollen ebenfalls rotieren. Die Objekte, welche die Bässe und die Höhen darstellen im gegen den Uhrzeigersinn und die Objekte der Mitten im Uhrzeigersinn.

Für einen Intensiveren visuellen Eindruck bewegt dich perspektive je nach dem niedrigsten verfügbaren Frequenzbereich von den Objekten weg. Aus den Bässen lässt sich zumeist ein Rhythmus der Musik ableiten. Ist der Abgespielte Ton im Bassbereich Laut, was darauf schließen lässt, dass sich die Musik um ihren stimmungstechnischen Höhepunkt befindet, sind alle Objekte vollumfänglich zu sehen.

Die konzeptionellen Anforderungen an die eingeteilten Bereiche der Visualisierung sind in Typescript umgesetzt wurden.

* 1. Umsetzung in Typescript

Die Umsetzung der Evaluierung in Typescript erfolgt in drei Schritten. Zunächst entsteht die Oberfläche eines Musikplayers mit Steuerelementen in HTML, mit Anpassungen im Aussehen und Aufrufen von Funktionen in Javascript durch ausgelöste Events beim Betätigen der Steuerelemente.

Der nächste Schritt ist die Erstellung einer THREE.js Szene in Typescipt. Es soll daneben eine Klasse entstehen, welche über die Audio-API von THREE.js verfügt. Die ausgelösten Events sollen auf die Methoden dieser Klasse zugreifen.

Im letzten Schritt wird die Visualisierung in einer separaten Klasse für Szenarien implementiert. Diese Klasse enthält Methoden zum Erschaffen von Objekten, zum Animieren dieser Objekte, sowohl wenn Musik gespielt wird als auch bei Pausen und einer Methode zum Zurücksetzten.

Ein grober Überblick über die Einbettung der Implementierungen ist in Abbildung 1 zu sehen.



**Abbildung 1,** Verschachtelungs-Skizze der Programmcodedateien

Am oberen Rand wird eine Leiste mit Steuerelementen erzeugt. Dies geschieht in HTML mit Symbolen in einer Tabelle.

Der Typescript-Quellcode ist in einem HTML-DIV-Tag eingebettet. In dem Programmcode befindet sich die THREE.js Szene und weitere Szenendefinitionen zum Erstellen einer Grundszene. Die Klassen für die Events, die Audio-API und die einzelnen Szenarien werden aus weiteren Typescript-Dateien eingebunden.

Der strukturelle Aufbau des Typescriptcodes ist in einem UML-Klassendiagramm in Abbildung 2 beschrieben. Jede Klasse ist in einer separaten Datei ausgelagert.

Die Event\_Handler-Klasse listet im Konstruktor alle benötigten Events für das Betätigen der Steuerelemente und Tastatureingaben auf und verknüpft diese mit den entsprechenden Methoden. Die Methoden rufen Funktionen aus der Audio\_Processing Klasse auf.

Die Audio\_Processing-Klasse initialisiert die Elemente aus dem THREE.js Modul zum Verwalten des Audios, zum Laden des Audios und zum Analysieren des Audios. Die Methoden zum Laden, Abspielen, Anhalten und Zurücksetzten des Audios werden von den Events aufgerufen. Die Klasse verfügt außerdem über eine Methode zum Aufrufen der richtigen Animationsfunktion. Basierend auf dem Status der Musik, ob diese abgespielt wird oder gerade pausiert ist, wird entweder die Leerlauffunktion oder die Visualisierungsfunktion der erstellten Szenarien aufgerufen. Die Methode selbst wird von der Animationsschleife der THREE.js Szene ausgeführt. Beim Aufruf der Visualisierung wird im Voraus die Analyse des Audios durchgeführt. Die Analysierten Daten entsprechen einem Array aus der mittleren Lautstärke jedes Frequenzbereiches. Die Frequenzbereiche werden von der FFT-Größe bestimmt. In einem Spektrum von 20 Herz bis 20 Kilo Herz steht die FFT-Größe für die Anzahl der Unterteilungen. Mit einer FFT-Größe von 2048 wird den Visualisierungsfunktionen ein Array mit 2048 Elementen zum Darstellen übergeben.

Die Scenaries-Klasse ist ein Prototyp. Die erzeugten Kindklassen erben den Funktionsumfang und Membervariablen, um das Initialisieren verschiedener Visualisierungsmethoden, projiziert auf die vier einzelnen Bereiche unter 3.1 Visualisierungskonzept beschrieben, zu vereinfachen. Der Funktionsumfang enthält eine Methode zum Erstellen der Objekte, eine für die Animation der Visualisierung, eine Leerlaufanimation, wenn die Musik nicht spielt und eine zum Zurücksetzten der Objekte.

Für das Balkendiagramm im Hintergrund existiert die Kindklasse Freq\_Bar\_Scenary. Diese Klasse erzeugt aneinandergereihte Balken entsprechend der Menge der analysierten Daten, in einer bestimmten Tiefe. Die Daten werden der Reihe nach auf die Skalierung der Balken in vertikaler Richtung übertragen. Die Leerlauffunktion lässt die Balken stätisch in ihrer Skalierung auf den Wert Null zurück schrumpfen.

Die Visualisierung der drei eingeteilten Frequenzbereiche erfolgt durch dieselbe Kindklasse. Die Klasse Cube\_Scenary erschafft kleine Würfel zufällig über die Szene verteilt. Die Visualisierungsfunktion bestimmt die Positionen in der Horizontalen und Vertikalen entsprechend einer Berechnung mit den Analysierten Daten der Musik. Durch eine Switch-Case-Anweisung kann hier zwischen den drei verschiedenen nach Frequenzbereich eingeteilten Varianten unterscheiden. Im Leerlauf, wenn Musik nicht spielt Bewegen sich alle Objekte in Richtung Mittelpunkt der Szene. Ist dieser erreicht, wird das Objekt an eine Zufällige Position der Szene zurückgesetzt.

Die drei Varianten erzeugten Objekte unterscheiden sich lediglich in der aus der Analyse bestimmten Position. Über Sinus und Cosinus Berechnungen des Zählers für den Frequenzbereich Ordnen sich die Objekte Kreisförmig um den Mittelpunkt an. Je höher die Frequenz, desto höher ist der Zähler beim Iterieren durch das Analysierte Datenarray. Mittels eines Faktors, welcher Proportional der Lautstärke entspricht, wird die Position moduliert. Bei den Bässen wird die Lautstärke von einer absoluten Konstante abgezogen, sodass die Würfel von außen nach inne ausschlagen. Bei den Mitten wird die Laustärke addiert, sodass diese vom Mittelpunkt heraus weiter nach außen kommen. Die Analysedaten der Mitten werden zudem unter einer Hüllkurve zur Aufbereitung gelegt. Niedrige Laustärken werden niedriger und Laute werden Lauter. Die Positionsberechnung der Höhen erfolgt mit vielfachen von der Konstante PI und ist daher Sequenzieller aufgeteilt.

Die Einfärbung der Bassobjekte ist röter, je lauter die Frequenzen sind. Bei den Mitten und Höhen ist der Rotanteil ebenfalls direkt proportional zur Laustärke. Grünanteile sind indirekt proportional zur Lautstärke und Blauanteile sind indirekt proportional zur Frequenz.

In der THREE.js Grundszene laufen alle genannten Klassen zusammen. Hier werden neben der Szene, der Kameraperspektive und dem Renderer Objektinstanzen erzeugt und als Übergabeparameter weitergereicht. Zuerst werden Objekte einzelner Szenarien instanziert. Diese Objekte werden beim initialiseiren der Audio\_Processing-Klasse übergeben, damit diese die Animationsfunktionen aufrufen kann. Der Eventhandler bekommt beim initialisieren die Audio\_Processing-Klasse übergeben um Funktionen zum bedienen des Audios aufrufen zu können. Zuletzt wird nur noch die Animationsschleife gestartet. Für jeden verfügbaren Frame des Browsers wird diese einmal ausgeführt. Die Animationsschleife ruft die Visualisierungsfunktion der Audio\_Processing-Klasse auf, welche die Animation der Visualisierung aufgrund des Musikstatus auswählt.



**Abbildung 2,** Klassendiagramm Typescript Quellcode

* 1. Installationshinweis

Das Visualisierungsprogramm ist in zwei Versionen verfügbar. Zum einen als fertiger Build und zum anderen als einzelne Quelldateien.

Im fertigen Build werden alle Javascript und Typescript Quelldateien inclusive der THREE.js Bibliotheken zu einer bundle.js-Datei mittels node.js zusammengefasst. Das soll die Performance steigern und sorgt für einen deutlich kompakteren Quellcode ohne zusätzliche Bibliothek oder Modulordner. Zur Ausführung wird ein zusätzlicher HTTP-Server benötigt. Der Node.js Paket Manager (npm) ermöglicht eine bequeme Installation über die Befehlszeile.

Für die Ausführung der Originalen Quellcodedateien werden die THREE.js Module benötigt. Diese können vom npm installiert werden. Der Umfang der eigensverfassten Projektdateien betrifft eine HTML-Datei „index.html“, eine CSS-Datei „style\_sheet.css“ und den Typescript Quellcode unter „Audio.ts“, „Event\_Handling.ts“, „Scenaries.ts“, „Scene.ts“ und „Global\_Collection.ts“.

Ein Konkretes Benutzerhandbuch mit Installationsbefehlen ist unter der beigelegten Datei „ReadMe.txt“ zu finden.

* 1. Musikvisualisierer

Die Nachfolgenden Bilder sind Screenshots des Visualisierungsprogrammes.

Abbildung 3 zeigt das Programm nach dem Aufrufen der Website. Alle Objekte wurden erzeugt und befinden sich in der Leerlaufanimation. Die Steuerelemente werden eingeblendet, sobald man mit der Maus dem oberen Fensterrand nahekommt. Die Bedienung enthält auf der linken Seite ein Bedienungsfeld zum Laden einer beigestellten Musikdatei. Diese ist offiziell als No-Copyright-Sound verfügbar. Direkt rechts daneben befindet sich ein Knopf zum Laden einer eigenen Musikdatei des lokalen Rechners. In der Mitte sind Steuerelemente zum Bedienen des Musikplayers. Die Symbole stehen der Reihe nach für das Abspielen, Pausieren und Zurücksetzten der Musik. Auf der rechten Seite wird der Name der Musik zu erkennen gegeben.

Beim Laden einer Musikdatei startet der Audioplayer automatisch. Die Visualisierung startet. Unter Abbildung 4 ist ein Screenshot einer laufenden Visualisierung zu sehen. Der Hintergrund ist klar unterscheidbar vom Rest und zeigt das Balkendiagramm entsprechend der Musik. Um den Mittelpunkt herum sind Objekte im Kreis angeordnet. Die, die rot eingefärbt sind entsprechen den Bässen. Sie verschwimmen mit den Objekten der Mitten. Außenherum weiter aufgefächert befinden sich die Objekte der Höhen.

Ein Bild, das Screenshot, Farbigkeit, Universum, Raum enthält.

Automatisch generierte Beschreibung

**Abbildung 3,** Leerlaufanimation mit Steuerelementen

Ein Bild, das Screenshot, Feuerwerk enthält.

Automatisch generierte Beschreibung

**Abbildung 4,** Visualisierungsanimation

1. Fazit

Visualisierer Luft nach oben, aber funktioniert gut auch Plattformübergreifend

Musikplayer könnte noch ausgebaut werden -> Verlaufsbalken

Three.js einfache Handhabung zum Erstellen von Szenen

Abbildungsbezeichnungen überprüfen, immer dann wenn ich mich auf eine Abbildung beziehe