AVPRG Wintersemester 2018/19

„Projekt: Visual-Soundilizer“

Fakultät Design, Medien und Information
Media Systems

Prof. Dr. Andreas Plaß
Jacob Sudau

Von:

Peter Oetker 2240231
Joel Ehlen 2288230

Inhaltsverzeichnis

1. Vorwort

Unser Projekt entstand im Rahmen der Audio- und Videoprogrammierung Veranstaltung an der Hochschule für angewandte Wissenschaften Hamburg.
Das Ziel der Veranstaltung war es Erfahrung im Umgang mit C++, JavaScript und der WebAudio API zu sammeln und diese in einem Projekt umzusetzen und zu präsentieren.
Unser persönliches Ziel war es ein Projekt mit der WebAudio API zu erstellen, um 3D-Sounds für Umgebungen in einer HTML Anwendung simulieren zu können, die eine möglichst realistisch akustische 360 Grad Umgebung wiedergibt. In den folgenden Abschnitten wird das Projekt von der Planung bis hin zur Umsetzung erläutert.

2. Technische Umsetzung

2.1 Programmierung

Umgesetzt wurde das Projekt mit den beiden Programmiersprachen JavaScript und C++ in Visual Studio Code, die das Programm einen Live-Server in ein Browserfenster laden.
Die HTML-Anwendung besteht aus drei Teilen: Der „3D-Audio“, dem „3D-Test“ und dem „MP3-Player“, die über Buttons im obersten Rand(Header) abrufbar sind. Jede Anwendung für sich in einem eigenen JavaScript, während das Design in C++ festgelegt wird und die Zusammen- und Ausführung über die Index-Datei erfolgt.

2.1.1 3D-Audio

Die „3D-Audio“ besteht aus einer Bedienung von vier Pfeiltasten, einem Reset-Knopf der Position und dazugehörigen Audioelementen, für die Vorführung einer 360 Grad akustischen Umgebung in einer digitalen Anwendung. Abgespielt werden die Audiodateien über den „Play-Button“ und über den Volume-Slider wird die Lautstärke geregelt. Neun weitere Buttons sind für die Sounddateien wie beispielsweise Katzen, Regen- und Windgeräusche zuständig, während vier andere für Kompositionen stehen, die einen musikalischen Hintergrund erzeugen.

Die Anwendung ist aufgeteilt in fünf JavaScript Dateien. Unsere *audioFileList.js* ist für die Speicherung der Sounddateien in einem Array zuständig, die in der play.js über einen Button die jeweils ausgewählte Audio abspielt oder stoppt. Für die Audioverarbeitung ist die *resonanceAudioinit.js* zuständig. Sie erstellt den AudioContext und die legt die Größe und Eigenschaften des Raumes fest, während der Slider für die Anpassung des Volumens in der Volume.js festgelegt ist.
Die Bewegung geschieht in der *moveAudioSource.js* und kann per Pfeiltasten die Positionierung der Soundquelle in der Funktion *moveSource()* ändern, die über vier if/else-if-Schleifen die vier Richtungen der Pfeiltasten über die x- und y-Koordinaten aktualisiert und über die *resonanceAudio* neu positioniert, wobei zu sagen ist, dass die Z-Achse nicht im Raum liegt und somit nicht für die Neupositionierung von Bedeutung ist.

2.1.2 3D-Test

Die „3D-Test“ ist zusammengesetzt aus einen 2D-Canvas mit einer Soundquelle(Schritt-Geräusche), die über Bitmaps als eine Kugel dargestellt werden und in einer konstanten Geschwindigkeit um den Hörer in einem 360 Grad Kreis rotieren. Ein Button für Play und Pause, ist unter dem Header positioniert.

Die Funktion *testNode()* initialisiert hier die einzelnen Parameter: Position, Rotation und Rotationsgeschwindigkeit, Bild, AudioBuffer und die PannerNode. Die Rotationsachsenberechnung wird hier in einer Sinus- und Cosinus-Funktion beschrieben die über die *updatePosition()* Funktion später abgerufen werden kann. Des Weiteren initialisiert die Funktion *load()* das Bild für die spätere Abfrage der Neupositionierung. Bilder und Sounddatei wurden ebenso über entsprechende Funktionen hinzugefügt.

Die *Funktion update()* erstellt die Größe des Canvas, die Variablen der Koordinaten und der Sounddateilänge, die endlos in einer while-Schleife die bitmap des Bildes der Soundquelle aktualisieren
In der *whenLoaded()* Funktion wird die „3D-Test“ mit ihren dazugehörigen Funktionen ausgeführt, die Rotationsgeschwindigkeit und die Position der Node, also der Soundquelle festgelegt. Als *panningModel* wurde hierfür der „HRTF“(Head-related-transfer-function) gewählt. Dies ist ein mathematischer

Algorithmus der die Krümmung und Reflektion des Schaals im Raum berechnet, die ansonsten nicht gegeben wäre.

2.1.3 „MP3-Player“

Der MP3-Player war eine visuelle Komponente die wir ursprünglich mit in 3D-Test implementieren wollten, dieses aber aus Problemen mit der Funktionalität ließen. Elemente, wie der Button für Play und Pause, Soundvolumen und Zeitleiste, sind importierte Standartbestandteile der *Audio Controlls*.

Die Animation der Bars wurde über eine *renderFrame()* Funktion ausgeführt, die zum einen die Animation mit dem *requestAnimationFrame()* ausführt und zum anderen über eine *AnalyserNode* die Werte der Frequenz der Sounddatei ausliest und in einem Array speichert und ausgelesen. Die ausgewerteten Daten, werden dann unter Berücksichtigung der Canvas-Größe bearbeitet und als Bars in den Canvas gezeichnet, die die jeweilige Höhe oder Tiefe der Frequenz visualisieren