# Projektkonzept: DIY-3D-Soundkulisse

Einleitung

DIY-3D-Soundkulisse ist ein Projekt zum Thema „tangible sound control“ und beschäftigt sich mit der Frage, wie man einem Benutzer die Möglichkeit geben kann, die Soundkulisse eines gegebenen Raumes mittels greifbarer Kontrollelemente nach seinen Wünschen anpassen zu können.

Entstanden ist die Idee bei der Vorführung der 3D-Soundanlage des Tonlabors der HAW. Eine der ersten Sachen die uns in den Sinn kamen, war dass es damit möglich sein müsste, einem User einen Raum soundtechnisch so einzurichten, wie er es will. Seien es harmonische Naturklänge, vertraute Geräusche von zuhause oder andere noch so verrückte Dinge – man schiebt sich einen Klang dorthin wo man ihn haben will, richtet sich den Raum ein, quasi wie ein Zimmer, und schließt dann die Augen um sich der Soundkulisse hinzugeben.

Das Projekt wird als Prüfungsleistung im Rahmen des Kurses „Audio-Video-Programmierung“, gelehrt von Herrn Plaß und Herrn Sudau, umgesetzt. Zur Videoprogrammierung wird mit openCV, einer C++-Bibliothek, gearbeitet. Der Audio-Teil wird mittels der WebAudio-API, einer JavaScript-Bibliothek, umgesetzt.  
Dabei sind die Ziele des Kurses das Erlernen der Grundlagen von Audio- und Video-Programmierung, sowie die Vertiefung dieser Kenntnisse, im Rahmen eines Projektes zum oben genannten Thema.

Zum Abschluss des Semesters wird DIY-3D-Soundkulisse in einer Live-Demo präsentiert und für Benutzer im wahrsten Sinne des Wortes greifbar gemacht.

# Projektziel

Das Ziel des Projektes „DIY-3D-Soundkulisse“ ist es, Benutzern zu ermöglichen, einem Raum eine Soundkulisse ganz nach Ihren Belieben zu verpassen.

Ausgangslage ist eine schlichte Platte, die den aktuellen Raum abstrakt visuell repräsentieren soll – jegliche Gegenstände die zur Einrichtung gehören, werden gestrichen, es geht nur darum mit ihrer Hilfe ein Gespür dafür zu bekommen, wo sich etwas im Raum befindet.   
Auf dieser Platte können dann Objekte platziert werden. Jedes einzelne Objekt symbolisiert einen bestimmten Sound, eine Note oder eine Melodie. Die Objekte unterscheiden sich in Farbe, Form und Größe, wobei diese Unterschiede die auditive Repräsentation des Objektes weiter parametrisieren – so werden die Parameter Farbe, Form und Größe übersetzt in Parameter aus der Soundtechnik wie Geschwindigkeit, Lautstärke und weitere.

Beispiele für solche Objekte sind Quader, Zylinder, Pyramiden und weitere geometrische Figuren.

Die Position der Objekte auf der räumlichen Repräsentation, spiegelt die Position des dreidimensionalen Klangs im Raum wieder. So kann man sich einfach eine Soundquelle an die Stelle schieben, an der man sie gerne hätte.

Neben der Platte gibt es noch ein Web-Interface. Dieses wird auf einem Laptop oder Tablett bereitgestellt. Es bietet die Möglichkeit, den verschiedenen Formen unterschiedliche Sounds zuzuweisen – hier programmiert sich der User quasi seine Hardware, die er dann physikalisch positionieren kann. Weiterhin hat er die Möglichkeit, zwischen verschiedenen Klangkulissen (Natur, Stadt, Haus, …) zu wählen.

Das Projekt wird prototypenhaft gehalten und soll als Beweis der technischen Möglichkeit dienen. Dafür grenzen wir uns mit der Zielsetzung auf wenige bestimmte Figuren und Sounds ein um den Rahmen des Projektes nicht zu sprengen. Auch der Raum wird, auf das am Campus der HAW verfügbare, Tonstudio beschränkt. Grund dafür ist, dass wir dort eine 3D Soundanlage haben, die wir bespielen können.

Für die Präsentation wird ein Stuhl in der Mitte des Tonstudios gestellt. Hierhin kann sich der User setzen. Vor ihm steht ein kleiner Tisch, mit der Platte, die den Raum repräsentiert und einem Laptop/Tablett. Ein Kreuz markiert die Mitte des Raumes. Neben ihm befindet sich eine Kiste, mit allen verfügbaren Figuren, sowie eine kleine Anleitung und Projektbeschreibung. Über und neben der Platte sind zwei Kameras installiert, welche die Szene auf der Platte filmen, um daraus die Soundkulisse zu erzeugen. Der Nutzer kann dann auf der Platte seine eigene Soundkulisse bauen, erzeugen und dynamisch verändern. Ihm steht nun offen, den Sound im Raum zu erkunden, oder sich einfach entspannt zurückzulehnen.

# Anforderungsanalyse

Unsere Anwendung soll möglichst intuitiv bedienbar sein und den Nutzer zugleich animieren experimentell neue Sounds und Klangkompositionen zu erstellen.

Der Nutzer kann zu Beginn eine Soundumgebung auswählen. Ihm werden unterschiedliche Kulissen zur Auswahl gestellt, beispielsweise Stadtgeräusche, Waldgeräusche, Meeresgeräusche oder Computerspielsounds. Wurde die Kulisse über ein Interface ausgewählt, darf der User am Tisch platznehmen. Die Tischfläche ist ca. 50x50 cm groß und zunächst leer. Der Spieler findet neben dem Tisch einige Formen, wie Würfel, Bälle und Pyramiden in verschiedenen Größen und Farben vor. Auch Knete in verschiedenen Farben stehen dem Spieler zur Verfügung. Ihm steht hierbei völlig offen, wie er die Objekte auf der Tischfläche platziert.

Er kann dabei die Formen stapeln, nachträglich verschieben, aus der Knete abstrakte Formen kneten und Kneten verschiedener Farben vermischen.

Der Nutzer kann nachträglich den Sound der jeweiligen Objekte nach seinem Belieben anpassen.

Mit einer kleinen Verzögerung sollen die Objekte auf dem Tisch wiederkehrende Sounds, passend zur anfangs gewählten Soundkulisse, erzeugen. Dabei ist die Schallquelle im Raum abhängig von der Position des Objekts auf dem Tisch. Die Größe, die Farbe und die Form des Objekts bestimmen die Lautstärke, die Abspielgeschwindigkeit und den Sound.

Im Folgenden stellen User Stories die Anforderungen an unsere Anwendung aus Sicht des Nutzers dar.

* Als Nutzer möchte ich meine Soundkulisse zunächst selbst bestimmen können.
* Als Nutzer möchte ich jedes Objekt separat platzieren, verwenden und bewegen können.
* Jedes Objekt soll gemäß seiner Farbe die Abspielgeschwindigkeit darstellen.
* Jedes Objekt soll gemäß seiner Größe die Lautstärke des Klanges darstellen.
* Jedes Objekt soll gemäß seiner Form den Sound defaultmäßig bestimmen.
* Als Nutzer möchte ich den Sound zu meinen Objekten ändern können.
* Die visuelle Position eines Objektes soll in eine auditive Soundquelle im Raumes übersetzt werden.
* Die Fläche, die mir als Nutzer zur Verfügung steht um die Objekte anzuordnen, soll im Verhältnis 1:10 zum auditiven Raum stehen.
* Eine Veränderung der Szene soll direkt in einer Veränderung des Sounds resultieren.
* Die Latenzzeit zwischen einer Veränderung der Objekte und der Veränderung des Sounds soll unter 5 Sekunden liegen.
* Als Nutzer möchte ich mich frei im auditiven Raum bewegen können.
* Als Nutzer möchte ich meinen erschaffenen Sound aufnehmen und ihn später abspielen können

# Technische Umsetzung und Rahmenbedinungen

Mit den Anforderungen, die der Kurs mit sich bringt, kommen einige technische Limitationen. Aufgabe ist es, mit der C++-Bibliothek openCV und mit der JavaScript basierten WebAudio-API zu arbeiten.

Unsere Software besteht also aus einem Frontend aus JavaScript (speziell WebAudio-API), HTML5 und CSS. Hierüber soll der User Sounds auf Objekte mappen und die Art der Klangkulisse konfigurieren können.

Im „Backend“ arbeiten wir mit C++ und openCV um mittels Bildverarbeitungsalgorithmen die aktuelle Szene (Anordnung der Objekte auf der Platte) zu analysieren und an das Frontend weiterzugeben – dort werden dann die gewählten Sounds erstellt und abgespielt.

Da die Installation von C++ und openCV, wie wir sie im Unterricht kennengelernt haben, bestimmte, sensible Ausgangsbedingungen (Windows PC, MS Visual Studio 2015 Express, …) erfordert, haben wir uns dafür entschieden, die Software für Windows-Systeme, egal ob Laptop oder Desktop-PC zu entwickeln. Theoretisch könnte dabei aber aufgrund der heutzutage fast überall verfügbaren Technologien, zumindest das Frontend auf fast jeder Plattform laufen.

Speziell für die Umsetzung unseres Konzeptes, ist die Abhängigkeit von der 3D-Soundanlage im Tonstudio der HAW. Das limitiert uns sowohl auf Seiten der Hardware als auch auf Seiten der Umsetzung und Durchführung des Projektes – trotz Simulation der Umgebung auf den heimischen Rechnern/Laptops, müssen wir immer wieder zum Arbeiten, Entwickeln und vor allem zum Testen des Projektes in die Universität und dort im Tonstudio die Software testen und weiterentwickeln. Trotz dieser Einschränkung haben wir uns dafür entschieden, da wir die Möglichkeiten die uns die Anlage bietet sehr spannend finden.

# Technisches Konzept

Wir haben mehrere Herausforderungen zu bewältigen.

Eine der Aufgaben wird die Programmierung eines UIs zur Übersicht, sowie zur Bearbeitung der Szenerie. Dieses wollen wir, wie bereits erwähnt, webbasiert umsetzen. Als Lösung dafür greifen wir auf HTML5 und CSS zu und bedienen uns der WebAudio-API, mit der es recht simpel ist, Sounds im Browser zu erzeugen. Der Grund dafür ist, dass der Kurs diese Technologien gelehrt hat und sie für das Projekt voraussetzt.

Weiterhin sollen hier die Daten aus der Video-Analyse hingeschickt werden und über MIDI-Daten ausgelesen werden. Daraus werden dann die 3D-Sounds erzeugt.

Diese Daten wiederum sollten im Webinterface angezeigt werden (Block\_A bekommt ein Info-Fenster mit Parametern wie Position, Lautstärke, …) und gegebenenfalls über UI-Elemente konfigurierbar sein.

Der größere und herausforderndere Teil wird die Video-Analyse der aktuellen Szenerie. Wir müssen mittels Kamera in Echtzeit Daten an die Web-Schnittstelle senden.

Dabei ist die erste Schwierigkeit, räumliche Tiefe und Positionen zu erkennen. Wir haben bisher in openCV nur Bilddateien analysiert und dort beispielsweise mit Farbinformationen gearbeitet. Unsere bisherigen Recherchen haben ergeben, dass man schon mit zwei Kameras Ansätze finden kann, um Positionen im Raum zu bestimmen. Ein Ansatz wäre dann, das gleiche Bild aus zwei Winkeln aufzunehmen, und die entstehenden Bilder abzugleichen. Darüber könnte dann eine Kalibrierung durchgeführt werden, mit Hilfe derer man dann Punkte im Raum zur Orientierung festlegt.

Sobald ein Raum erkannt werden kann, müssen wir uns darum kümmern, Objekte in diesem zu erkennen. Um dieses vorerst möglichst simpel zu halten, beschränken wir uns auf Objekte mit einer Form und mehreren verschiedenen Farben – dort können wir mit dem Wissen aus der Vorlesung aufbauen.

Ist auch das erledigt, kümmern wir uns um die Implementation verschiedener Formen. Hierbei müssen wir beachten, dass ein Computer natürlich nicht ad-hoc irgendwelche Formen erkennt – wir müssen uns Algorithmen überlegen, damit er das tut. Problematisch könnten dabei auch Rotationen und Skalierungen werden. Wir bräuchten hierfür einen Ansatz, der eine gegebene Figur normalisiert und dann mittels eines Algorithmus auf z.B. Skalierungen überprüft. Eine Idee hierfür wäre jeder Form einen markanten Aufkleber zu verpassen, der über Farberkennung identifiziert wird und dann die Informationen beherbergt, um welche Form es sich handelt.

Zu guter Letzt müssen wir uns überlegen, inwiefern wir die Klötze die Sounds parametrisieren lassen – also ob die Größe die Lautstärke beeinflusst und ähnliches. Das ist eine Designentscheidung, mit der wir im Laufe des Projektes experimentieren werden, um eine möglichst intuitive Bedienung zu ermöglichen.

Die letzte Herausforderung ist die Kommunikation zwischen C++/openCV und der WebAudio-API, doch

auch dafür haben wir bereits im Unterricht erste Schritte getan. Der Ansatz über MIDI-Daten wird sehr wahrscheinlich auch für unser Projekt ausreichen.

# Bedienkonzept

Zum Beginn der Nutzung von „DIY-3D-Soundkulisse“ wird der Nutzer dazu aufgefordert den Raum (auf der Platte) zu kalibrieren. Dafür läuft im Webinterface ein Video mit angemessenen Instruktionen, die zeigen wie der Kalibrierungsprozess abläuft.

Dieser Schritt wird vor jeder ersten Benutzung nötig sein – wechselt der User zwischendrin, muss der Raum nicht neu kalibriert werden.

Nach der Kalibrierung findet sich der User vor einem Webinterface wieder, welches möglichst intuitiv sein soll. Das heißt im konkreten Fall, dass keine grafischen Spielereien oder Effekte ablenken von dem Inhalt der Seite. Das oberste Menü ist ein Kontextmenü, über welches Sounds auf bestimmte Objekte gemappt werden können und die Art der Soundkulisse angepasst wird. Es ist also dafür da, allgemeine Einstellungen zu der Szene zu treffen.

Darunter befindet sich ein etwas größeres Panel, welches mehrere Informationen bereitstellt:

Eine Hierarchie der aktuellen Szene (Welche Objekte befinden sich in der Szene?), als Liste. Daneben befindet sich ein scrollbares Fenster, in welchem jedes Objekt aus der Szene ein Objektpanel bekommt. Dieses bietet detailliertere Informationen über das Objekt: Sound(datei), Lautstärke, Geschwindigkeit und weitere Parameter. Eventuell könnte man hierüber eine Alternative Bedienbarkeit dieser Parameter ermöglichen. Dafür bräuchte es dann entsprechende Slider etc.

Der Großteil der Bedienung wird allerdings losgelöst von programmierbaren Interfaces stattfinden. Als Nutzer sitzt man vor einer Platte, die den Raum darstellt. Dieser wird auf der (falls nötig) noch auf der Platte gekennzeichnet. Innerhalb des Raumes auf der Platte können Bauklötze (jeder mit einer geometrischen Form) platziert werden. An der Stelle im echten Raum taucht dann ein Sound auf. So kann man sich sehr intuitiv die Soundkulisse so basteln, wie man es gerade im Kopf hat und erhält direkt auditives Feedback.

# Zeitplan

Meilensteine:

|  |  |  |
| --- | --- | --- |
| **Datum** | **Meilenstein:** | **Zeitaufwand [PH]** |
| 10.12.17 | openCV mit 2 Kameras parallel betreiben | 80 |
| 14.12.17 | Algorithmus zur Erkennung des Raumes implementieren | 80 |
| 15.12.17 | Erkennung verschiedener Objektformen in openCV | 80 |
| 14.12.17 | Erstes Frontend-Mockup | 12 |
| 17.12.17 | Objekte im dreidimensionalen Raum erkennen (mit openCV) | 25 |
| 20.12.17 | Kalibrierungsprozess zusammen mit Frontend | 30 |
| 28.12.17 | MIDI-Daten an WebAudio-Frontend übertragen | 15 |
| 30.12.17 | Frontend finalisieren | 27 |
| 04.01.18 | Sounds auf Basis der übermittelten MIDI-Daten erzeugen | 12 |
| 04.01.18 | Sounds auf Basis der MIDI-Daten im Raum positionieren | 18 |
| 06.01.18 | Mapping von Audiosnippets auf die Objekte | 10 |
| 08.01.18 | Erstellen von Soundkulissen | 8 |
| 12.01.18 | Test | 40 |
| 15.01.18 | Troubleshooting | 30 |
| 16.01.18 | Präsentation | 4 |
|  | **Gesamt** | **471** |

# Teamplanung

|  |  |  |
| --- | --- | --- |
| **Name** | **Matrikelnr.** | **Aufgabe** |
| **Kevin Hagen** | 2270985 | Videoprogrammierung – Erkennung von Objekten und Formen |
| **Jordanis Lazaridis** | 2142772 | Audioprogrammierung – Frontendentwicklung |
| **Carina Krafft** | 2269579 | Videoprogrammierung – Entwicklung von Algorithmen zur Raumerkennung und Positionsbestimmung |
| **Irena Becker** | 2238833 | Schnittstellenprogrammierung |