HAW Hamburg
Studiengang Medientechnik
Wahlpflichtmodul Technik
Audio- und Videoprogrammierung
Wintersemester20/21

Projektdokumentation
„Paperbeat“

Tim Hoffmann, 2324704
Paulo Shklovsky, 2317471
Niklas Wehl, 2320262

Abgabe:
26. Januar 2021

git-Repositorie:
https://github.com/TimHoffmann1998/Beat-Tafel

Ziel der Veranstaltung Audio- und Videoprogrammierung war es, eine Software zu entwickeln, welche durch einen Live-Videofeed Einfluss auf ein Audioereignis nimmt. Unser Projekt „Paperbeat“ ist das Ergebnis dieser Zielsetzung.

Kernstück des Projektes ist eine Tafel mit einem Raster. Der Benutzer kann mit dieser Tafel in musikalisch aktiv werden und durch ein einfaches System sowohl einfache als auch komplexere Rhythmen legen.

Damit der gewünschte Beat wiedergeben wird, müssen auf dem Rasterfeld Farbkarten in die gewünschte Reihenfolge gelegt werden. Wie bereits im Konzept dieses Projekts beschrieben, handelt es sich um ein 4x4-Felder große Tafel. Dabei stellen die Zeilen einen spezifischen Teil eines Instrumentes dar und die Spalten die Taktabschnitte. Zugrunde liegt ein 4/4 Takt, welcher durch geschicktes Legen in verschiedene Taktstrukturen umgewandelt werden kann. Insgesamt gibt es fünf verschiedene Farbkarten, jede Farbe repräsentiert eine andere Taktung des gewünschten Audios. Damit lassen sich eine Vielzahl an Beats erstellen. Maximal sind 16 Anschläge möglich.

Taktcode	Dezimalwert	Farbton (HSV)
1 1 1 1	15	0
1 0 1 0	10	190
1 0 0 0	8	340
0 1 0 0	4	40
0 0 0 1	1	123

Abbildung 1: Zuordnung der Farben zu den Taktcodes

Der User kann durch einen Start/Stopp-Button, als auch durch Schieberegler für Gain und bpm (Beats per minute) zusätzlich Einfluss auf den Beat nehmen. Der gelegte Rhythmus wird einmal pro Sekunde ausgewertet und mittels MIDI an die Webseite übertragen.

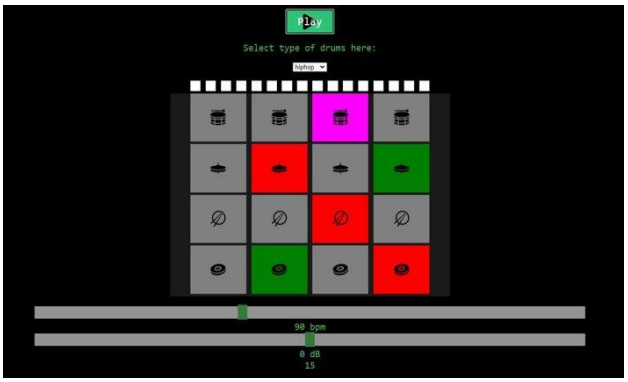

Abbildung 2: Benutzeroberfläche mit Steuerelementen

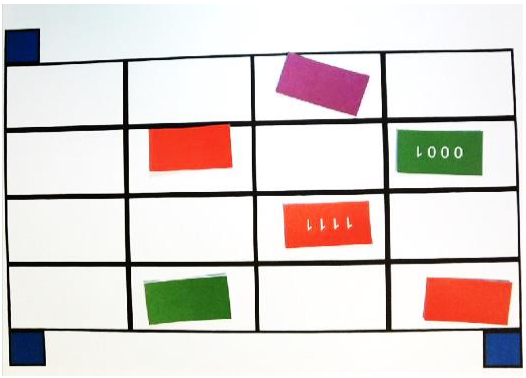

Abbildung 3: Tafel mit Farbfeldern

Zur Nutzung der Software sind lediglich eine Webcam und das gedruckte Feld mit Farbfeldern nötig. Bei der Verwendung der Tafel ist zu beachten, dass ein gleichmäßiges und farbechtes Licht auf das Papier fällt, damit das Auslesen fehlerfrei ablaufen kann. Zudem ist ein qualitativ guter Druck ebenso zu empfehlen, wie auch eine weiße Hintergrundfläche.

In den folgenden Punkten werden die einzelnen Komponenten der Software dargestellt und die markanten Punkte der Software genauer erläutert.

Videoverarbeitung in Python

Gestartet ist die Umsetzung des Projektes mit der Videoverarbeitung. Erstes Ziel war es, einen Videofeed einzulesen, diesen darzustellen und nach den Farben zu filtern. Als erstes Testmaterial diente ein Beispielvideo, welches auf Basis des Konzeptes angefertigt wurde und mit zufälligen Farbfeldern bestückt war. Zum Auswerten der Tafel wurde auf aus den Hausaufgaben bekannte Methoden und Algorithmen zurückgegriffen.

Für die Erkennung der Farben wird das Bild in Parameter des HSV-Farbraums unterteilt und nach im Anfang des Programms definierte Farbwerte ausgewertet. Die entstanden Masken müssen anschließend mit Hilfe des Medianfilter von unerwünschten Fragmenten bereinigt werden, um eindeutige Regionen zu erkennen. Diese Regionen sind die jeweiligen Papierkarten auf dem Rasterfeld mit jeweils identischen Farben. Damit sind die ersten wichtigen Schritte getan, nun müssen die einzelnen Felder noch eindeutig unterschieden und dessen Position berechnet werden.

Zum genauen Unterscheiden der Felder wird die Funktion „cv2.findContours“ verwenden. Diese bildet unsere gewünschten Regionen und gibt jeder einen Index. Die gewonnen Informationen werden in eine Funktion überführt, sodass für jedes Feld eine individuelle Farbnummer generiert wird. Diese Feldnummer setzt sich aus den dem Farbton (Hue) und den Y- und X-Koordinaten des Objektes zusammen. Der Farbton wurde bereits am Anfang des Skriptes definiert, also müssen nur noch die Position des Feldes ermittelt werden. Dazu werden bis auf eine die Masken der einzelnen Regionen geschwärzt und der Schwerpunkt des übergebliebenen Feldes ermittelt. Die X- und Y-

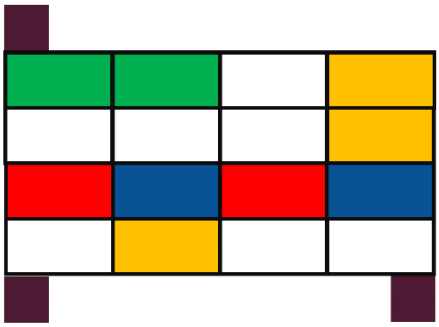

Abbildung 4: Bespiel-Tafel

Abbildung 5: Maske von roten Farbkarten

Werte des Schwerpunktes bilden nun die Position dieses Feldes. Zusammen mit der dem Farbton wird aus allen drei Werten die Feldnummer generiert. Dabei steht an erster stelle der Farbton, dann der Y-Wert und dann der X-Wert. Zur Trennung der Parameter wir jeweils ein Punkt zwischen diesen eingesetzt. Ergebnis ist ein String mit einer eindeutigen Definition des jeweiligen Feldes.

Beispiel für eine Feldnummer: *120.234.489* → *Farbton (Hue) . Y-Wert . X-Wert*

Mit Hilfe dieser Nummer können nun weitere Schritte erfolgen. Zum Einen erfolgt nun eine Einsortierung der Felder in das Raster und zum Anderen kann nun auch der passende Taktcode bestimmt werden. Die Taktcodes mit den zugehörigen Farben sind in der Abbildung 1 dargestellt. Die Codes sind Dezimalzahlen, welche in Binär umgewandelt die Taktstruktur der jeweiligen Zelle darstellen. Diese werden mittels bedingter Verzweigungen den Feldern zugewiesen. Daran anschließend müssen noch die Felder in die richtige Zelle einsortiert werden. Dazu müssend die X- und Y-Koordinaten ebenfalls eine Abfrage durchlaufen. In dieser Abfrage werden Bedingungen gestellt, welche durch die Kalibrierungspunkte dem Bild der Kamera entsprechen angepasst sind. Dies ermöglicht eine Verschiebung der Tafel im 2-Dimensionalen Raum.

Die Sortierung der Papierkarten im Skript erfolgt in einer Liste. Dabei beginnt die Liste mit dem Wert oben Links im Raster und folgen dann zeilenweise dem Raster nach unten rechts. Jene Elemente, die in der Liste nicht gefüllt werden, da die Zelle nicht belegt ist, werden mit einer Null komplettiert.

Diese Liste wird mit allen Elementen am Ende des Ausleseprozesses seriell über einen MIDI-Kanal verschickt, also alle 16 Informationen. Zu Beginn jeder Übertagen wird eine Startzahl übermittelt, die 127. Das Pythonskript übersendet unabhängig vom JavaScript. Als Übermittlungskanal wird ein MIDI Kontrollkanal verwendet. In der MIDI-Nachricht wird das 3. Byte mit unserer Information bestückt. Als Output wird LoopBe1 verwendet. Nun erfolgt die weitere Verarbeitung in JS.

Audioverarbeitung in Javascript

MIDI Daten

Für die Kommunikation zwischen Python und Javascript nutzen wir das MIDI Protokoll. Dafür werden die einzelnen Taktslots, als 4 Bit Information in einer MIDI-Nachricht verschickt. Die Bits stehen hier für die einzelnen Sechzehntelnoten und eine 1 bedeutet, dass das Instrument gespielt wird, eine 0 dass es aussetzt. Für einen Vollen Takt, müssen demnach für 16 Felder 16 Nachrichten verschickt werden, die alle jeweils 4 Bit Informationen als Dezimalzahl enthalten.
Um diese Informationen in ein Format zu bringen, das für eine einfache Weiterverarbeitung bzw. Ausspielung des Taktes geeignet ist, wird die empfangene Dezimalzahl zunächst in einen String umgewandelt. Dieser String entspricht dem 4 Bit Muster der Dezimalzahl. Danach werden die einzelnen Bits als Zahlen in einen Taktarray geschoben, der in x-Richtung die Zeit und in y-Richtung die Instrumente darstellt.
Zur Synchronisierung wird vor den 16 MIDI-Nachrichten eine Präfix Nachricht geschickt, die den Wert 127 enthält - dieser Wert kommt in den anderen Nachrichten nicht vor. Dann werden die 16 Nachrichten über einen Counter abgezählt, der Takt gespeichert und zurückgesetzt und der Counter wieder auf 0 gesetzt.

Soundscheduling und Timing

Ein weiterer wichtiger Teil des Programms ist das Timing der Drumsamples, sodass diese in einem regelmäßigen Takt gespielt werden. Das Programm sollte dabei aber nicht in einer Schleife gefangen sein, damit noch andere Aufgaben ausgeführt werden können.
Wenn der “Play” Knopf gedrückt wird, wird die Zeit, wann die nächste Note gespielt werden soll (nextNoteTime), auf die aktuelle Zeit des AudioContext gesetzt (context.currentTime) und die currentNote auf 0. Dann wird eine scheduler Funktion gestartet.
Die scheduler Funktion fragt nun ab ob context.currentTime größer ist als nextNoteTime, also ob genug Zeit vergangen ist, um die nächste Note zu spielen. Ist dies der Fall, werden alle Noten für diesen Taktschlag (currentNote) gespielt und die nextNoteTime festgelegt. Am Schluss wird festgelegt, dass die scheduler Funktion nach einer Wartezeit von einem 20tel Taktschlag erst wieder aufgerufen werden soll, damit das Programm nicht in einer Schleife gefangen bleibt.

Audiograph

Der Audiograph der WebAudioAPI besteht in unserem Fall aus drei Arten von Nodes. Die ersten sind die BufferSourceNodes in die die Drumsamples geladen werden. Dann gibt es noch eine GainNode für den Mastergain und eine Destination. Die Audio Buffer werden in einen Array geladen und geändert, wenn andere Drumsamples (hiphop, beatbox und natural) ausgewählt werden, geändert.
Dir Gainvalue kann über einen Slider angepasst werden.

HTML & CSS

Das HTML-Skript beinhaltet, neben des Play-/Stopp-Buttons und der Dropdown-Auswahl des Drum-Stils, vor allem eine Veranschaulichung des Taktes in 4/4eln, verteilt auf eine Reihe aus 16 Blöcken und einen Block aus 16 Blöcken im 4x4 Raster, welches das aktuell aktive, bzw. übertragene Raster darstellt.
Die Taktung läuft über eine Javascript Funktion, welche auf die Taktung des Schlagzeug-Elemente zurückgreift und im CSS Skript die Farbe des jeweils aktiven Taktviertels zu Orange ändert, während die restlichen als „Weiß“ definiert sind.
Beim Raster wird das über MIDI übertragene Array ausgelesen und die jeweiligen Dezimalzahlen geben an, welche Farbe die jeweilige Block-ID in CSS Skript annimmt.
Unterm Raster besteht noch die Möglichkeit die Geschwindigkeit in Beats pro Minute, sowie den Gain in dB zu bestimmen.
Im CSS Skript wurden vor allem Button, Slider und die Hintergrundfarben ein wenig angepasst, damit einerseits das Raster besser zur Geltung kommt, andererseits um die Web-Applikation ein wenig intuitiver zu gestalten.

Drum-Samples

Mit zur Aufgabe wurde das Erstellen der eigenen Drum-Samples, welche soundtechnisch allgemein bekannte Bereiche abdecken sollten. Diese umfassen ein möglichst natürliches Schlagzeug, Hip-Hop Drums, sowie mittelmäßig gelungene Beatboxaufnahmen. Diese wurde alle auf grob -3dBFS gemischt, um ein Übersteuern bei recht natürlichem Einsatz der Samples zu vermeiden. Die Samples wurden größtenteils mit leichtem Verzerrer, leichter Kompression, Equilizer, sowie kurzem Hall effektiert. Vor allem Snaredrums und Kickdrums bestehen aus mehreren Layern von Samples und sind im Pitch an die restlichen Drums angepasst.

Aufwandsschätzung

Der durchschnittliche Arbeitsaufwand pro Woche pro Person: ca. 5 Stunden

Organisation + Planung: ca. 12 Stunden

Python + Tests + Debugging: ca. 25 Stunden

JavaScript + Test + Debugging: ca. 20 Stunden

Mischung und Sounddesign (Drums): ca. 3 Stunden

HTML/CSS + Test + Debugging: ca. 10 Stunden