Klang der Steine – Dokumentation

Amira Brandt (2357873), Henrike Bohnet (2269780)

GitHub-Ordner

https://github.com/Brandsatz/Audio-Video-Programmierung

Technische Umsetzung

Wir haben für die Umsetzung unseres Projektes zwei verschiedene Programmiersprachen verwendet. Zum einen Python, wo das Bild und Farbwerte analysiert und die ausgewerteten Daten per Midi-Message an Java-Script geschickt werden. Dies ist der andere Teil, wo die Soundausgabe stattfindet.

Zum Python-Teil:

Die Funktionen zur Analyse werden erst aufgerufen, nachdem ein Midi-Signal empfangen wurde, also nachdem der Benutzer auf den Button der Analyse gedrückt hat. Dann wird ein Screenshot erstellt. Dieses wird nun analysiert. Zuerst wird mithilfe der „rehcteck()“-Funktion die schwarzen Flächen rausgefiltert. Dabei wird die größte schwarze Fläche ermittelt. Anschließend wird nach vergleichbar großen Flächen geschaut. Da die Belichtung auf dem Foto teilweise sehr unterschiedlich ist, kann es sein, dass die eingestellte Maske mit den BGR-Werten den zweiten schwarzen stein nicht gut sichtbar werde lässt, da dieser heller belichtet ist. Darum wird, wenn keine weitere vergleichbare schwarze flache gefunden wird, die zweitgrößte Fläche ermittelt. Stimmt deren y-Koordinate in etwa überein mit der, der ersten schwarzen Fläche, so erkennt das Programm diese beiden schwarzen Fläche als die schwarzen Legosteine. Sollte es nicht so sein, wird ein Midi-Signal an JS geschickt, worauf ein Pop-Up auf der Webseite erscheint, das darauf hinweist, dass mein Fehler aufgetreten ist und die Analyse neu gestartet werden muss.

Gehen wir davon aus, dass die Analyse geklappt hat. Dann wird anhand der jeweiligen x- und y-Koordinaten der schwarzen Steine ein Zuschnitt des Bildes erstellt, um nur die Fläche zwischen den Steinen zu analysieren und mögliche Störfaktoren zu minimieren. Auch wird anhand der y-Koordinaten die Positionen berechnet der farbigen Steine. Wir haben den festen Wert von 8 farbigen Steinen zwischen den Schwarzen verwendet. Man könnte das ganze hier noch anpassen. Entweder, dass der Benutze vorher entscheidet, an wie vielen möglichen Positionen er die Steine platzieren will, oder dass das Programm selber erkennt wie viele mögliche Steine zwischen den Schwarzen Steinen positioniert werden können mithilfe der Länge der schwarzen Felder, die die gleiche Länge haben, wie die farbigen Steine.

Anschließend werden die einzelnen Farben nacheinander analysiert. Dazu haben wir das Bild in den HSV-Farbraum konvertiert, um die Fragen besser auszuarbeiten. Anschließend haben wir nach Flächen geschaut, die eine vergleichbare Größe haben, wie die schwarzen Steine. Wurden welche gefunden, wurde die Position des Mittelpunktes der Fläche bestimmt und errechnet an welcher Position der jeweilige Stein steht.

Diese Informationen (Farbe und Position) werden per Midi-Message JS geschickt.

Im Moment haben wir beim Aufbau Lücken zwischen den Steinen gelassen, um eine räumliche Trennung zu schaffen, damit gleichfarbige Steine, die nebeneinander liegen, einzeln zu erkennen. Auch hier könnte man im nächsten Schritt versuchen zu erkennen, wann gleichfarbige Steine nebeneinander liegen und diese getrennt behandeln.

Zum JavaScript-Teil:

Das JavaScript Programm ist in fünf Teilskripte unterteilt. Der eine ist zur Kamerasteuerung gedacht. Hier ist wichtig die Webcam beim Senden eines Midi-Signals von der HTML Seite zu entkoppeln, damit Python auf diese zugreifen kann, und um sie wieder aufzugreifen sobald ein neuer Drum-Loop eingerichtet werden soll.

Das Skript Slider greift auf die Slider auf der Webpage zu und analysiert einen Statuswechsel. In Convolver wird der Hall erzeugt. Und das Midi Skript hört auf eingehende Midi-Signale und versendet selbst das Start-Signal an Python.

Der wohl komplexeste Teil findet sich im Main Skript. Hier werden die einzelnen Sound Notes gestartet und der Loop abgespielt. Dieser Loop war wohl der komplizierteste Teil im Main Skript. Man braucht hierfür eine Funktion, die sich selbst wieder startet, die aber manuell abgebrochen werden kann, um den Loop auch wieder zu pausieren. Eine While-Schleife kommt hier schwer in Frage, da es schwer ist einen manuellen Abbruch und vor allem eine zeitliche Verzögerung für aufeinanderfolgende Töne zu implementieren.
Und eine reine rekursive Funktion würde sich auch direkt wieder aufrufen was zu einer zu hohen Anzahl an Audio Outputs zur Folge hätte.
Hier kommt die Funktion setTimeout(function(),intervall) ins Spiel. Sie startet eine Funktion nach einer festgelegten Zeit. Wenn man hier also die gleiche Funktion loop() nochmal aufruft und zwar nach der berechneten Zeit von acht Takten so wird nahtlos an einen Taktdurchlauf ein neuer gestartet. Bevor der Taktdurchlauf aber überhaupt berechnet und gestartet wird, wird erst einmal überprüft ob der Loop noch spielen soll, oder schon gestoppt wurde. Je nachdem landet man in der if() Bedingung, die den Loop startet oder einfach abbricht.

Um einen visuellen Taktdurchläufer anzutreiben muss zu Beginn des Loops die Länge eines Vierteltaktes aus der angegebenen Geschwindigkeit berechnet werden. Für jeden einzelnen Takt muss nun eine einzelne setTimeout() Funktion gestartet werden, welche die Funktion zur visuellen Veränderung des Taktbereiches aufruft. Und dies für jeden Takt hintereinander.

Auswertung des Zeitplans

Zu der Zeitplanung können wir sagen, dass wir trotz des Verschiebens der Analyse der Momentaufnahme hin zu Python, eine recht genaue Einschätzung des zeitlichen Aufwandes geliefert haben.
Das Interface in HTML zu erstellen und designen hat in etwa einen Arbeitstag in Anspruch genommen. Ganz zum Schluss, nach der Präsentation des Prototypen wurde es im Design noch etwas angepasst, aber nur damit es auch auf verschiedengroßen Monitoren gleich aussieht.
Die Erstellung der Videoanalyse in Python war dagegen ein wenig Zeitaufwendiger als geplant. Dies lag vor allem an der oben genannten Verlegung der Analyse hin zu Python und einigen Problemen, ausgelöst durch die Farberkennung und der Ausschnitt des Bildes mit Hilfe der schwarzen Steine. Dies hat die Bearbeitungszeit des Python Codes verlängert.

Außerdem hat die Implementierung einer Midi Kommunikation von JavaScript hin zu Python, also ein Midi-listener auf Seiten Pythons und die Implementierung der Midi Outputs auf Seiten JavaScripts, ein wenig Recherchezeit benötigt.

Somit sind wir für den Pythonteil mit Erstellung, Recherche und Problembehebung etwa auf eine Bearbeitungszeit von 35 Personenstunden gekommen.

Für den JavaScript-Teil kamen dann noch mal in etwa 25 Personenstunden hinzu. Hier hat der Soundloop mit der durchlaufenden Grafik den längsten Teil in Anspruch genommen. Mit Tests und der Vorbereitung zur Präsentation sind wir in etwa auf jeweils 40 Personenstunden gekommen und lagen damit sehr gut in unserem geschätzten Arbeitsaufwand.

Was wir für die Zukunft auf jeden Fall mitnehmen ist sich ein besseres Zeitmanagement zu überlegen. Wir hätten vor allem mehr Pause über die Weihnachtstage einplanen sollen, da hier für uns beide viel Zeit mit der Familie anstand.
Der Vorteil war jedoch, dass wir in einem Zweierteam waren und somit flexibler mit unseren Zeiten arbeiten konnten.

Fazit

Im Großen und Ganzen ist uns das Projekt gut gelungen und wir hatten Spaß daran.
Die Arbeit zusammen hat auch sehr gut geklappt, was nicht unbedingt selbstverständlich war, da wir uns vorher nicht kannten. Durch die derzeitige Situation konnten wir uns auch immer nur online treffen. Trotzdem haben wir schnell unseren Rhythmus gefunden und auch die Aufgabenteilung war relativ schnell klar.

Das Arbeiten am Projekt ist am Anfang gut in Gang gekommen, doch zwischenzeitlich hatten wir beide ein kleines Tief, aber danach konnten wir uns beide wieder gut motivieren. Und nach einer effektiven zweiten Hälfte ist am Ende ein vorzeigbares Produkt entstanden.

Zu diesem ist zu sagen, dass es unter gleichbleibenden (Licht-) Bedingungen auch stabil läuft. Der Knackpunkt liegt vor allem in der Erkennung der schwarzen Steine, da hierdrauf das gesamte weitere Programm aufgebaut ist. Wir haben uns für schwarze Steine entschieden, da wir dachten, dass diese sich gut von den anderen Steinen abheben. Allerdings war es doch schwerer als gedacht, da das Bild gut ausgeleuchtet sein muss, damit keine Schatten entstehen, die ebenfalls schwarz sind. Aber es sollte nicht so belichtet werden, sodass die schwarzen Steine reflektieren und somit nicht schwarz erscheinen. Im Nachhinein wäre es vielleicht besser gewesen eine andere Farbe sich auszusuchen, wie beispielsweise grün, da diese besser rauszufiltern ist.

Zudem sind einem im Nachhinein natürlich noch Ideen gekommen was man noch hätte ergänzen können.
Besonders die abschließende Präsentation der Projekte lieferte hierfür noch viel Inspiration. So sind Dinge, die wir noch weiterführend als sinnvolle Ergänzungen sehen, eine Farb-Kalibrierung, sodass man nicht auf eine konstante Beleuchtung angewiesen ist, sowie ein direkterer Eingriff in die Einstellungen im Taktdurchlauf. So dass die Geschwindigkeit oder der Stop Befehl nicht erst aktualisiert werden, nachdem der Takt einmal durchgelaufen ist.

Über unser Endergebnis sind wir aber, besonders auf Grunde der Bedingungen, die dieses Semester mit sich brachte, dennoch sehr stolz.