Der letzte Schrei

Emma Richter 2319547
Jette Heinsohn 2320987

AV-Programmierung WS19/20

Ziel

Das generelle Ziel des Projekts ist es die Steuerung von Sounds mit Hilfe von Bewegungen möglich zu machen. Dabei war die Idee mit Bewegungen auf der vertikalen Achse die Lautstärke zu regeln und auf der horizontalen Achse die Tonhöhe. Um die Steuerung für den Anwender einfacher zu machen, sollen die Änderungen abhängig von der Position des Kopfes beeinflusst werden, da dieser im Vergleich zum Körpermittelpunkt relativ einfach kontrolliert bewegt werden kann. Auch war es uns sehr wichtig, die Latenzen so gering wie möglich zu halten, um das Gefühl direkter Änderungen in den Geräuschen bei Bewegungsänderungen zu erzeugen.

Bedienkonzept

Für den Nutzer gibt es verschiedene Möglichkeiten zur Einstellungen. Zu allererst kann durch die Bewegung des Kopfes vor der Kamera die Tonhöhe und Lautstärke gesteuert werden. Des Weiteren können in der grafischen Benutzeroberfläche (GUI) die verschiedenen Modi, welche unterschiedliche Soundsamples beinhalten, ausgewählt werden. In der aktuellen Version gibt es eine einfache Sinusschwingung, einen DIY-Sound, in welchem man mit Hilfe der Schieberegler die Gewichtung zwischen Sinus-, Dreieck-, Rechteck- und Säbelzahnschwingung bestimmen kann, den sogenannten “Depeche Mode”-Sound, welcher ein Musiksample beinhaltet, welches bei den Bewegungen gepitcht wird, und den “Schrei”, welcher ein Schreisample enthält. Allerdings kann in diesem Modus nur die Lautstärke kontrolliert werden. Des Weiteren kann in der GUI das Mastervolume festgelegt werden und der Frequenzbereich zwischen dem linken und rechten Rand des Kameraausschnitts festgelegt werden. Auch wird die Bewegung des Kopfes, dem Nutzer in der GUI angezeigt.

Hardware

An Hardware wird zur Ausführung des Programmes nur ein Computer und mit diesem verbunden eine Webcam und Lautsprecher benötigt.

Software

Für die Umsetzung unseres Projektes benutzen wir Python 3.7 und OpenCV sowie PyQt 5 und Pure Data Extended.

OpenCV ist eine Pythonbibliothek, die zur Bildverarbeitung dient. Wir benutzen diese Bibliothek zusammen mit einem Haar-Casscade-Classifier, welcher auf die Erkennung von Gesichts trainiert wurde, um die Bewegungen des Kopfes zu tracken. Dabei verrechnen wir die entstehenden Werte mit Vorwerten, um die Bewegung flüssiger zu machen und große Sprünge, wie zum Beispiel durch ein falsch erkanntes Gesicht zu vermeiden.

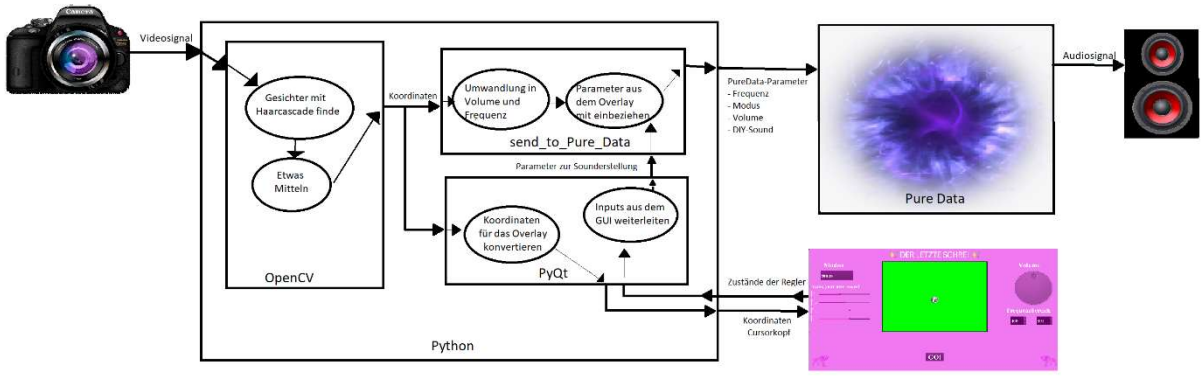
PyQt ist eine Bibliothek, mit welcher man grafische Oberflächen in Python erstellen kann. Wir haben mit dieser zunächst einmal ein Fenster erzeugt und dann in diesem die weiteren Elemente, sogenannte Widgets, platziert. In unserem Fall Combobox, Dial, Spinbox, Button, Slider, Painter und Labels, welche sowohl für Texte als auch für Bilder genutzt werden können. Auch bietet PyQt die Möglichkeit die Werte, die in den Widgets eingestellt werden, auszulesen und im weiteren Code zu verwenden, dieses passiert auch nahezu latenzfrei.

Zudem benutzen wir PyQt für Threads. Es werden Threads benötigt, damit Python die Oberfläche erstellen und laufend updaten kann und parallel dazu weiter das Gesicht tracken und die Soundparameter updaten kann. Der GUI-Abschnitt unseres Codes ist im Hauptabschnitt und das Tracking und senden an Pure Data zur Erstellung des Sounds findet im Thread statt. Der Thread wird gestartet mit dem Aufrufen einer Funktion, die an einen Knopf gebunden ist. Innerhalb des Threads können Signale gesendet werden, die in derselben Funktion, die den Thread gestartet hat, ausgelesen werden. Gesendete Signale führen dazu, dass alle Funktionen die mithilfe von Signalen aufgerufen werden, geupdatet werden. Dies gilt für die Koordinaten, die zu dem GUI gesendet werden, um die visuelle Darstellung der Bewegung zu updaten, als auch für die Schalter und Regler, die innerhalb des GUI ein Signal aussenden, sobald mit ihnen interagiert wird. Die Werte, die von den Knöpfen geändert werden, sind alle globale Variablen, damit sie auch von den Funktionen innerhalb des Threads genutzt werden können.

Pure Data ist eine visuelle Programmiersprache für die Realisierung von Sounds. Dabei gibt es Blöcke, die man erstellt und miteinander verbindet. Zu diesen Blöcken gehören zum Beispiel Oszillatoren, um Schwingungen zu erzeugen und mathematische Operatoren um diese zu formen. Der Sound, welcher von Pure Data ausgegeben wird, updatet sich automatisch, sobald sich einer der Eingabewerte ändert.
In Pure Data haben wir mehrere Codeblöcke, die verschiedene Sounds erstellen, am einfachsten die Sinusschwingung mit einem Oszillator und ein steuerbarem Multiplizier-Block, um die Amplitude der Schwingung und somit die Lautstärke zu kontrollieren. Diese Blöcke-Ansammlungen werden mithilfe von sel-Blöcken auf null gestellt. Diese senden ein Impuls an ein Null-Feld, wenn der Input 1 ist. Vor dem Input befindet sich ein ungleich-“jeweiliger Modus”-Block, der 1 ausgibt, wenn der Modus nicht ausgewählt ist. Ähnlich wird mithilfe der Spigots der übertragene Lautstärkewert nur umgesetzt, wenn der Input 1 ist. So wird der Lautstärkewert nur durchgelassen, wenn der entsprechende Modus ausgewählt ist.
Die Samples werden mit Hilfe von tabwrite in ein Array geschrieben, welches dann ausgelesen wird. Bei dem Musik Sample wird mit Hilfe von einer Granularsynthese und einer Hanning-Fensterung die Tonhöhe verändert, ohne dabei die Geschwindigkeit zu beeinflussen.
Für den DIY-Sound werden die einzelnen Schwingungen erstellt und dann ihre Lautstärke entsprechend verändert, um ihr Verhältnis den Schieber Stellungen in der GUI anzupassen. Das Volume ergibt dabei zusammen immer 1 durch eine Berechnung in unserem Python Code, welche die Gewichtungen der verschiedenen Schwingungen in ein Verhältnis setzt, ohne die Lautstärke zu beeinflussen.

Signalfluss

In der folgenden Grafik wird der Signalfluss zwischen der Hardware, Python und Pure Data dargestellt. Die Datei ist in einer besseren Auflösung noch einmal im Git zu finden.



Reflexion

Insgesamt wurde das generelle Ziel erreicht. Das Programm kann einen Ton ausgeben, dessen Eigenschaften mit der Bewegung des Kopfes gesteuert werden können. Dabei ist die Latenz gering genug, sodass ein Gefühl von sofortigen Reaktion auf die Bewegung vorhanden ist.

Während der Entwicklung des Programms, hatten wir mehrere Probleme, die uns teilweise sehr viel Zeit geraubt haben. Ursprünglich wollten wir unser Programm in JavaScript und python umsetzen. Allerdings haben wir die Verbindung zwischen den verschiedenen Frameworks nicht aufgebaut bekommen. Dazu kam noch, dass wir vorher noch nie mit Java bzw. JavaScript gearbeitet hatten, sodass dort unsere Kenntnisse begrenzt waren. Auf der Suche nach einem neuen GUI wollten wir zuerst Kivy benutzen. Allerdings hat diese auf Laptop zunächst nicht funktioniert. Als wir daraufhin versucht haben einige Elemente neu zu instalieren, hat Python auf dem Laptop erst nach einer Woche und nach doppelten neuinstallieren wieder funktioniert. Ein weiteres Problem, welches wir mit dem Bau unserer GUI hatten, ist das wir ursprünglich geplant hatten das Video in unser GUI einzubauen, wo sich nun das grüne Rechteck befindet. Dies könnten wir nicht umsetzen, da wir nicht einmal einen Weg gefunden haben das Video gleichzeitig mit dem GUI in einem separaten Fenster anzeigen zu lassen.Um dem Nutzer aber immer noch die Möglichkeit zu geben die eigenen Bewegungen zu verfolgen, haben wir deshalb nur die Koordinaten übermittelt und diese grafisch in einer dem Bildausschnitt entsprechendem Fenster anzeigen lassen. Ein weiteres Problem war, dass wir auch bei dem Schreisample die Tonhöhe variabel machen wollten. Allerdings hat dies selbst mit Hanning-Fensterung und einer Verkleinerung der Stufen in der Granularsynthese schlechter geklappt als mit anderen Samplen. Tendenziell haben wir bemerkt dass es mit Samplen besser geklappt hat, deren Frequenzspektrum weniger gleichmäßig verteilt ist. Dies ist bei Stimmsamples leider nicht der Fall, weswegen diese schwer zu Pitchen sind, ohne die zeitlichen Parameter zu verändern. Ein weiteres Problem bestand darin, dass manchmal Gesichter erkannt werden, wo keine sind, und dadurch Pure Data zum Absturz gebracht wurde. Durch Einbeziehung vorheriger Koordinaten in die

Aktuellen, wurde das Problem minimiert, sodass die Positionsdaten nicht mehr ruckartig springen, kleinere Ausschläge gibt es trotzdem noch, welche zu einem leichten ruckeln führen können.

Zudem hätten wir mit mehr Zeit gerne noch weitere Features eingebaut. Zum Beispiel wäre das parallele Tracking von mehreren Gesichtern für die Erzeugung von mehreren voneinander unabhängigen Tönen eine schöne Erweiterung gewesen. Außerdem hätten wir gerne die Funktion unterstützt, dass man das Video zusammen mit dem Ton aufnehmen kann. Allerdings waren wir uns auch nicht sicher inwiefern das mit unseren Fähigkeiten umsetzbar gewesen wäre, da dafür vermutlich Codecs notwendig gewesen wären um das Video mit dem Ton zu verknüpfen. Auch ein Spielelement, indem man mit bestimmte Kopfbewegungen Melodien nachbauen muss, angelehnt an GuitarHero, hätten wir spannend gefunden.