Projektarbeit Musikprogrammierung
bei Prof. Dr. Denis Lorrain
IMWI
HfM Karlsruhe

von Tim Engel
Matrklnr.: 11698
SommerSemester 2014

1. Das Projekt
Ursprünglich hatte ich vor, eine lernende künstliche Intellegenz zu programmieren, die zum einen Proportionen in Harmoniefolgen und Notenintervallen der klassischen Musik – also z.B. (Sub-)Dominantbeziehungen, Medianten, Septimen, Terzen, usw. – lernt. Zum anderen sollte diese Intellegenz viele Portraitbilder von verschiedenen Gesichtern auf Proportionen der Merkmale im Gesicht – also Augen, Nase, Mund, Haaransatz, Ohren, Kinnform, Bartwuchs, Augenbrauen, usw. – analysieren und aus diesen Werten lernen. Hierbei wären Proportionen von Größe, relative Position/Abstand zueinander, Farbe u. Ä. die Daten, die gelernt würden.
Anschließend sollen die gelernten Proportionen miteinander verglichen werden und zugeordnet werden, so dass z.B. ein Augenabstand von 400 Pixeln dem Intervall einer Quarte oder blaue Augen einem Durakkord der Tonika zugeordnet wird.

Sobald eine Basis an gelernten Daten beider Seiten besteht und Verknüpfungen geschaffen sind, wäre das Ziel, dass man nun ein (neues) Gesicht als Bild-Datei einlesen kann, welches auf die oben genannten Merkmale des Gesichtes analysiert wird und aufgrund der schon zuvor gelernten Daten in eine Harmoniefolge/Melodie umgewandelt wird.

Als optionales weiteres Ziel hatte ich noch vor, das Ganze auch umzudrehen. Also daß man eine Melodiefolge einliest, die dann analysiert wird und auf Basis der schon gelernten Merkmale ein neues Gesicht zeichnet, das einer Umwandlung der Musik in ein Gesicht entspricht (nach den gelernten Verknüpfungen der jeweiligen Proportionen.)

Da dies allerdings einer künstlichen Intellegenz bedarf, derer ich mit meinen derzeitigen Programmierfähigkeiten nicht gerecht werden kann, habe ich diese Idee vorerst verworfen und habe versucht, das ganze herunterzubrechen auf folgende Idee:
Ich wollte mit Hilfe von Standbildern (z.B. JPG-Dateien) einen gegebenen Klang so verändern, dass das jeweilige Bild Pixel für Pixel auf RGB-Daten ausgelesen wird, die wiederum die Werte unterschiedlicher Parameter des Ausgangsklanges bestimmen. Konkret sähe das z.B. so aus, dass anfangs ein einfacher Sinuston erklingt, der sich aber im Laufe der Zeit um bestimmte Obertöne erweitert, die sich, je nach Werten der RGB-Daten, in der Frequenz und im Schwingungsverhalten verändern – mal in harmonischen Proportionen zum Grundton, mal in disharmonischen Proportionen. Dadurch bliebe der Ausgangston auf einer

Frequenz, aber die Klangfarbe würde stets variieren.

2. Bisherige Umsetzung

Dies habe ich (leicht verändert) so realisiert, dass ich mit Hilfe der Cinder-Library für C++ eine Bilddatei auf ihre RGB-Daten ausgelesen habe, diese dann in einer Textdatei gespeichert habe. Die Textdatei wird anschließend mit SuperCollider als Liste eingelesen. Daraufhin wird über diese Liste iteriert, so dass drei Oszillatoren gleichzeitig erklingen und miteinander moduliert werden. Zwei Sägezahn-Oszillatoren und ein Sinus-Oszillator erklingen somit in einer additiven Synthese gleichzeitig. Hierbei erhalten die Sägezahn-Oszillatoren die Rot- und Grün-Werte und der Sinus-Oszillator die Blau-Werte wobei die Sägezahn-Oszillatoren skaliert sind auf Frequenzen zwischen 60 Hz und 1000 Hz, der Sinus-Oszillator dagegen zwischen 30 Hz und 1570 Hz.

Wenn die Frequenzen wechseln, wird durch ein Lag-UGen interpoliert, um einen gleitenden Frequenzwechsel zu erhalten. Jede Frequenz klingt zwischen 0.2 und 1.0 Sekunden lang, was bis jetzt über einen Zufälligkeitsgenerator gesteuert wird.

Um eine übersichtliche Anzahl von RGB-Daten zu haben, habe ich die Möglichkeit einer Bereichsauswahl eingebaut, mit der man zuerst das gewünschte Bild anzeigen lassen kann, um dann mit der Maus ein Kästchen zu ziehen, das den Bereich markiert, dessen RGB-Daten in die Textdatei geschrieben werden. Denn hätte man ein Bild von nur 600x800 Pixeln, wären das schon 480 000 Pixel zu je 3 Werten, also 1 440 000 Werte, die es zu interpretieren gilt. Das wiederum hieße, bei einer durchschnittlichen Dauer von 0.6 Sekunden pro Pixel 288 000 Sekunden, also 80 Stunden.

Derzeit ist es so, dass das Erstellen der Liste mit den Daten allein in C++ (mit Cinder) geschieht, und die Klangerzeugung allein in SuperCollider. Beide müssen getrennt voneinander manuell ausgeführt werden. Denn es gibt zwar eine Library, mit der man in C++ (mit Cinder) SuperCollider ansteuern kann, die aber (bei mir) nicht richtig funktioniert. Ich kann leider keine Verbindung von C++ und SuperCollider herstellen. Weder diverse Diskussionen in Online-Foren zu diesem Problem (das anscheinend bei mehreren Personen besteht), noch Herr Romero (Dozent für SuperCollider) konnten mir dabei weiterhelfen.

3. Weitere Entwicklung

Die weitere Entwicklung habe ich vorerst eingestellt, da ich wegen noch zu unausgereiften Fähigkeiten im Programmieren die ursprüngliche Idee so stark abwandeln musste, dass ich mittlerweile nicht mehr wirklich hinter dem Ergebnis des Projektes stehe. Aber durch die intensive Beschäftigung mit dem Handling von Bildern in C++ konnte ich sehr viel dazulernen. Und durch die Beschäftigung mit einem solchen Projekt konnte ich Erfahrungen machen, wie es ist an einem solchen Projekt zu arbeiten, mit welchen Herausforderungen man konfrontiert wird und wie man damit umgeht.