AVPRG: Gesichtsmusikant

Alina Böttcher, Marcel Plaga, Marvin Steinmetz, Linh Do

1 Einleitung

Im Rahmen der Lehrveranstaltung Audio-Video-Programmierung im 5./6. Semester, welche sich im Allgemeinen mit der Programmierung von Echtzeit-Effekten im Audio- und Videobereich auseinandersetzt und sich als Wahlpflichtmodul an Media-Systems- wie auch Medientechnik-Studierende gleichermaßen richtet, sollen im Wintersemester 2016/17 in Gruppen von jeweils 4 Studierenden verschiedene Projekte unter dem Themenschwerpunkt „Facial Expression Audio Synthesizer“ umgesetzt werden. Diese sollen im Januar auf einer Ausstellung (zusammen mit der Lehrveranstaltung IT-Systeme) einem Publikum bestehend aus Studierenden, Hochschulangehörigen sowie auch Firmen vorgeführt werden.

Als Grundlage hierfür ist anhand von Beispielen aus den Bereichen der Audiosignalerzeugung und MIDI-Steuerung sowie der Bildverarbeitung mit OpenCV eine Einführung in die Programmiersprache C++ und die Entwicklungsumgebung QT-Creator gegeben worden. Als Studienleistung zählt neben dem Projekt sowie dessen Vorführung selbst auch die Erstellung eines Theoretischen Papers, welches sich gezielt mit einer bestimmten Thematik im Bezug zur Umsetzung des Projektes auseinandersetzen soll. Ebenso ist von den Gruppen ein technisches Konzept der geplanten Projekte anzufertigen. Weiterhin soll die Vorstellung eines Prototypens eine frühzeitige Evaluierung der Umsetzung des Projektes ermöglichen.

2 Projektziel

Mit der Software „Gesichtsmusikant“ erhält der Anwender die Möglichkeit mit Hilfe seines Gesichtes Musiksamples abzuspielen oder Töne zu generieren. Dazu positioniert sich der Anwender zunächst vor einer Kamera, über welche sein Gesicht erkannt wird. Auf einem Bildschirm steht eine Benutzeroberfläche zur Verfügung.

Hierrüber wird bei einem Anwenderwechsel zunächst ein Kalibrierungsprozess aufgerufen, über welchen die Wertebereiche der Steuerparameter optimal an das Gesicht des neuen Nutzers angepasst werden. Dieser Vorgang beinhaltet z.B. dass der Mund so weit wie möglich geöffnet werden muss oder auch die Augenbrauen maximal hoch und runtergezogen werden sollen. Prinzipiell kann ein neuer Nutzer die Anwendung auch ohne Kalibrierungsprozess nutzen, dieser muss dann jedoch mit einer ggf. eingeschränkten und fehlerhaften Steuerung rechnen.

Da sich die Zielgruppe, welche sich nach den Besuchern der Ausstellung richtet, aus einem breiten Spektrum von musikalisch unerfahrenen bis hin zu musikalisch sehr affinen Anwendern zusammensetzt, besteht für den Anwender die Möglichkeit über die Benutzeroberfläche zwischen zwei Modi mit unterschiedlichen Funktionen und Benutzeroberflächen umzuschalten.

An die musikalisch eher unerfahrenen Nutzer richtet sich dabei der Sample-Modus. Hier kann der Anwender auf der Benutzeroberfläche zwischen verschieden Sample-Sets z.B. Drum-Set, Cartoonset

oder auch gemischte Sets wählen. Dadurch werden dann den verschiedenen Mimiken des Anwenders einzelne Samples fest zugeordnet. So wird beispielsweise wenn das Drum-Set ausgewählt ist beim Öffnen des Mundes eine Kick-Drum abgespielt oder beim Hochziehen der Augenbrauen eine Hi Hat ausgelöst. Weitere Möglichkeiten hat der Anwender hier nicht, hier liegt der Fokus vor allem auf dem Rumspielen mit verschiedenen lustigen Samplesets und dem Herausfinden der Zuordnung.

Musikalisch bewandertere Nutzer bekommen mit dem Umschalten in den Synthesizer-Modus schon interessantere Möglichkeiten zur Klangerzeugung. Hier steuern die einzelnen Elemente eines Gesichtes grundlegende Funktionen eines Audio-Synthesizers wie Oszillator-, Modulations-, Filter-, Hüllkurven- und Effektparameter. Auf der Benutzeroberfläche kann der Anwender zunächst zwischen verschiedenen Belegungen umschalten und bekommt hier auch die detaillierte Zuordnung der Gesichtselemente zu den Steuerfunktionen angezeigt. In einem zusätzlichen Fenster auf dem Bildschirm wird zudem der Synthesizer direkt eingeblendet. Hier kann der Anwender direkt auch visuell nachverfolgen wie sich die Parameter ändern. Zudem kann hier der besonders interessierte Anwender auch noch einzelne Funktionen anpassen und umschalten, wie z.B. die Signalformen, Filtertypen oder auch die Oktavlage wählen.

Natürlich dürfen und sollen auch beide Zielgruppen beide Modi ausprobieren, wobei in dem Synthesizer-Modus das Ergebnis auch stark vom Nutzer abhängen kann.

3 Anforderungsanalyse

Das Erkennungsprogramm soll dem Benutzer die Möglichkeit geben sich zu entscheiden, ob er lieber Laute und Geräusche abspielen oder mit Tönen spielen möchte. Das soll die Möglichkeit eröffnen das nicht nur Benutzer, die sich mit Musik auskennen, die Software nutzen können, sondern auch Laien, die keine Ahnung von Synthesizern haben.

Der Musikkenner kennt sich mit Synthesizern aus und soll mit der Software die Möglichkeit einer alternativen Bedienung eines kleinen Synthesizers bekommen. Für diesen Benutzer ist die Variante mit den einzelnen Tönen und den Synthesizer typischen Manipulierungsmöglichkeiten. Je nach Gesichtsausdruck bzw. Position des Kopfes werden die einzelnen Regler des Synthesizers manipuliert.

Die zweite Art von Benutzer kennt sich nicht gut bis gar nicht mit Synthesizern aus. Für diese Benutzer ist der zweite Modus der Software gedacht. Hier ist kein Synthesizer zu finden. Es werden je nach Gesichtsausdruck bzw. Position des Kopfes einzelne kurze mp3 Sounds abgespielt.

Da die Bewegungen im Gesicht bzw. mit dem Kopf schnell wechseln können, muss die Reaktion des Systems möglichst schnell erfolgen. Dadurch soll verhindert werden, das ein Benutzer schon drei Bewegungen weiter ist aber das Programm z.B. den Regler im Synthesizer noch nicht bewegt hat.

4 Technische Rahmenbedingungen

Für unser Projekt benutzen wir übliche Windows Desktop-Computer sowie Windows Laptops. Es ist dringend erforderlich, dass es eine Kamera/Webcam zur Aufnahme des Gesichts gibt und dass der PC Töne ausgeben kann. Für die Entwicklung wird für das Projekt die Programmiersprache C++ und dazu die IDE QT Creator verwendet, da wir uns bereits in den Vorlesungen bei beidem hereingearbeitet und beschäftigt haben. Eine Limitierung des QT Creators gegenüber anderen IDEs wie zum Beispiel Microsoft Visual Studio ist vor allem, dass der Debugger weniger gut funktioniert und wenig bei der Entwicklung des Projekts hilft. Im Moment sehen wir noch nicht so ein großes Problem darin. Aber im Laufe der Zeit wird sich zeigen, ob das für das Projekt ein größeres Hindernis darstellt und man deswegen zusätzlich eine andere IDE zum Debuggen verwenden müsste.

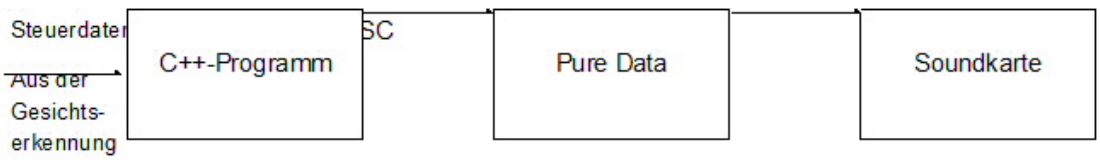
Die Audiotechnische Umsetzung des Synthesizers soll in Pure Data erfolgen. Zum einen weil hier bereits schon Erfahrungen innerhalb der Gruppe vorhanden sind und es sich um eine Open-Source-Lösung handelt und zum anderen weil es sehr offen in den Möglichkeiten ist und vor allem auch direkten Echtzeiteingriff und Visualisierung der Parameter ermöglicht, was gerade in den Testphasen einen großen Vorteil darstellt um die Parameter und Wertebereiche einstellen zu können und ähnliches.

Für die Übertragung der Audiosteuerparameter soll hier Open Sound Control (OSC) verwendet werden, da es zum einen MIDI immer weiter ablöst und an vielen Stellen bereits Standard bei der Übertragung von Audiosteuerungsdaten ist und zum anderen speziell für die Echtzeitverarbeitung entwickelt wurde. Im Gegensatz zu MIDI kann OSC auch über das Netzwerk übertragen werden, was ggf. ein Vorteil sein kann wenn ein PC für die Gesichtserkennung und einen weiterer für die Audioausgabe erforderlich würde. Auch kann die Unterstützung von Gleitkommazahlen und die Möglichkeit der größeren Datentypen einen weiteren Vorteil darstellen um gewisse Parameter besonders feinschrittig weitergeben zu können.

5 Technisches Konzept

Sound

Über ein C++-Programm sollen die extrahierten Steuerparameter von der Gesichtserkennung in OSC-Nachrichten verpackt werden und an einen PureData-Patch mit dem Synthesizer weitergeleitet werden. Dafür wird in C++ die Oscpack-Bibliothek genutzt, welche alle wichtigen Funktionen für das Erstellen von OSC-Nachrichten enthält.



Pure Data soll dann die Nachrichten entgegennehmen und an die jeweiligen Funktionen weiterleiten. Der Synthesizer soll dabei in etwa aus den folgenden Funktionen bestehen:

Oszillatoren:

Oszillator 1	Signalform (Sinus, Rechteck, Dreieck, Sägezahn) Oktave (32‘, 16‘, 8‘, 4‘, 2‘)
Oszillator 2	Waveform (Sinus, Rechteck, Dreieck, Sägezahn) Oktave (32‘, 16‘, 8‘, 4‘, 2‘) Pitch
Suboszillator	1 oder 2 Oktaven unter Oszillator 1

Filter:

Typ Auswählbar (HP, TP, BP)
Flankensteilheit wählbar (12, 24, 36db / Oktave)
Grenzfrequenz einstellbar (20 Hz …. 20 kHz) Resonanz einstellbar

Modulation:

Low Frequency Oscillator	LFO Rate (0.1 … 100) Signalform
Oszillator 1 und 2	Tonhöhe und Amplitude Auswählbar und stärke einstellbar
Filter	Grenzfrequenz und Resonanzfrequenz Auswählbar sowie stärke einstellbar

Hüllkurven:

3 ADSR Hüllkurven einstellbar
Je für Tonhöhe, Lautstärke, Grenz- und Resonanzfrequenz zuordbar

Mischer:

Oszillator 1
Oszillator 2
Suboszillator
Rauschen (Wählbar zwischen Pinken und Weißem Rauschen)
Reverb
Master

Gesichtserkennung

Damit die Frames von der Webcam erfasst, wiedergegeben und bearbeitet werden können, wird die freie Programmbibliothek OpenCV verwendet. Für die Gesichtserkennung (Face Detection) sowie die Punkteerkennung auf dem Gesicht (Face Point Localization) standen mehrere Bibliotheken/Frameworks zur Auswahl, wie das Flandmark Point Detector, das CLM-Framework oder ebenfalls OpenCV. Letztendlich fiel die Wahl auf Dlib, die zwar weniger bietet als andere

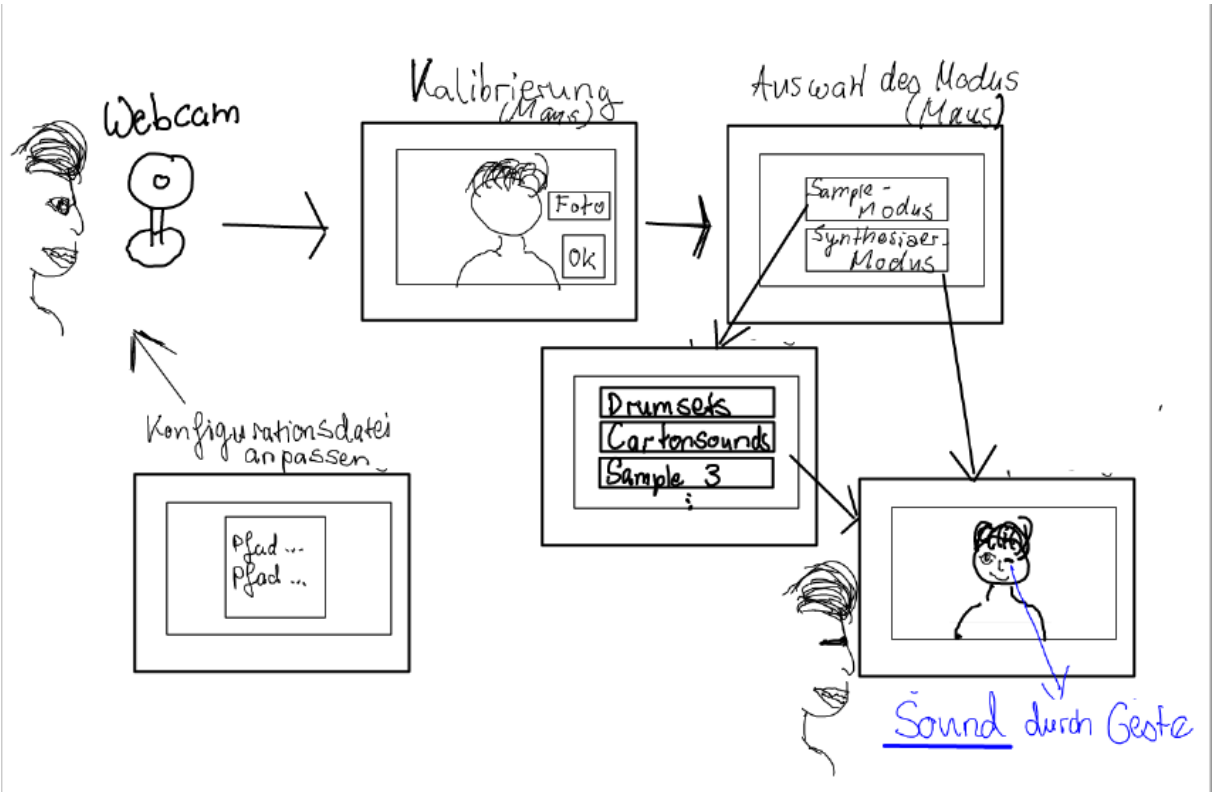
Frameworks, jedoch für die Gesichts- und Punkteerkennung völlig ausreicht und auch beim Praxistest mit der Geschwindigkeit punkten konnte.

Interpretation der Landmarks

Nach der Erkennung der Landmarks können wir über die shape()-Funktion die verschiedenen Koordinaten unserer Schlüsselpositionen abfragen und aus ihnen unsere Parameter berechnen. So setzt sich in unserem Modell der Augenbereich aus jeweils 6 Landmarks zusammen. Aus der Entfernung der Punkte werden wir so den Öffnungsgrad der Gesichtspartie ermitteln. Diese Methode lässt sich auf die Mund- und Augenpartien anwenden. Für die Augenbrauen müssen wir dann die Nasenrücken als festen Ankerpunkt für die Positionsermittlung verwenden. Um Fehler bei stark variierenden Gesichtern zu vermeiden ist eine kurze Kalibrierung vor der Verwendung unserer Anwendung geplant in der unsere Parameter als Vergleichswerte für den neutralen Gesichtsausdruck des Nutzers berechnet und gespeichert werden.

6 Bedienkonzept

Bevor man das Programm startet, muss der Benutzer in einer Konfigurationsdatei bestimmte Sachen anpassen. Bisher ist es nur vorgesehen, dass der Pfad der Trainingsdateien sowie der Pfad der Audiodateien angepasst werden müssen. Möglicherweise stellt sich später heraus, dass noch andere Sachen in der Konfigurationsdatei angepasst werden müssen.
Beim Start des Programms gelangt man zunächst zur Kalibrierung. Da jedes Gesicht verschiedene Gegebenheiten hat, muss jedes einzeln für sich kalibriert werden, sodass die Wertebereiche der Steuerparameter angepasst werden. (Nachtrag: 29.01.2017 wurde aus Zeitmangel weggelassen).
Ist dies getan, gelangt man in ein Menü, in dem man zwischen zwei verschiedene Modi auswählen kann. Zum einen dem Sample-Modus, in denen der Benutzer mit fertigen Sample Sets herumspielen kann und zum anderen dem Synthesizer-Modus, in denen verschiedene Synthesizerfunktionen abgespielt werden können. Die Erzeugung der Klänge soll dabei ausschließlich mit dem Gesicht funktionieren.



7 Zeitplan

Zeitraum	Teilnehmer	Aufgabe/Milestone	Geplante Zeit in h
01. -06.11	LD	Recherche	10
	AB		10
	MS		10
	MP		10
07. -13.11	LD	Ausprobieren und festlegen, welche Frameworks/ Bibliotheken verwendet werden sollen; Schreiben des Papers	10
	AB	Einarbeitung Musikabspielen in C++	15
14. -21.11	LD	**Meilenstein: Prototyp Gesichts- und Punkteerkennung**	20
	AB	Schreiben der MyMediaPlayer-Klasse	20
	MP	PureDataPatch einrichten	10
22. -28.11	LD	Konfigurationsdatei schreiben	10
	AB	SoundModus-Klasse schreiben	4
	MP	PureDataPatch einrichten	15
	MS	Kalibrierungssequenz programmieren	8
29.11 -5.12	LD,MS	Menü/ Bedienung programmieren, Konzept der Punkteverarbeitung (z.B. wann ist der Mund geschlossen, Gesichtsposition), Berechnungen dazu	20
	AB	Verknüpfen der Soundmodi mit Gesichtserkennung	5
		Definieren der abzuspielenden Sounds	6
	MP	PureDataPatch + OCS	20
06. -12.12	LD,MS	Programmierung Punkteverarbeitung	15
	AB	QWidget für Moduswahl	4
13. -19.12	LD,MS	**Meilenstein: Punkteverarbeitung fertig programmieren**	15
	MP	**Meilenstein: Grober PureDataPatch per OSC steuerbar**	20
	AB	Verknüpfen der SoundModus Klasse mit dem Gesichtserkennungsalgorithmus	5
20. -27.12	LD,AB,MS	Zusammenfügen der Programmteile	20
	MP	Zuordnung der Soundsteuerung+Testläufe	20
28.12 -02.01	LD,AB,MP,MS	**Meilenstein: Betaversion des Projektes fertigstellen**	50
03. -06.01	LD,AB,MP,MS	Testing, Fehler/Bugs beheben,Feinabstimmungen	40
17. -23.01	LD,AB,MP,MS	Ausstellung vorbereiten	30

Linh Do(LD), Marcel Plaga(MP), Alina Böttcher(AB), Marvin Steinmetz(MS)

8 Teamplanung

An dem Projekt „Gesichtsmusikant“ arbeiten im Rahmen des Kurses Audio-Video-Programmierung Alina Böttcher, Linh Do, Marcel Plaga und Marvin Steinmetz. Die für unser Projekt erforderlichen Teilbereiche konnten in 4 Aufgabengebiete aufgeteilt werden und werden von unseren Projektteilnehmern einzeln bearbeitet und im Anschluss zusammengetragen.

Audioeffekte und Soundsteuerung:

Marcel Plaga übernimmt im Rahmen des Projektes die Umsetzung unseres Synthesizers über PureData. Als unser Audio Fachmann realisiert er die benötigten Audioeffekte und die Übertragung der Steuerparameter. Im weiteren Verlauf des Projektes wird sich Marcel um die Optimierung der Audiosteuerung und die Zuordnung der Programmparameter zu Audioeffekten kümmern

Gesichtserkennung:

Linh Do übernimmt den Teilbereich der Gesichtserkennung. In diesem Bereich ist das Auswerten des Videostreams, das Erkennen von Gesichtern und das Erkennen wichtiger Gesichtsmerkmale zu realisieren. Als weiterführende Aufgaben wird das Zusammenfügen aller Bereiche als ein Projekt, Optimieren und die Fehlerbehebung in der Landmarkgenerierung anfallen.

Gesichtsinterpretation:

Marvin Steinmetz ist für die Interpretation der Landmarks zuständig und wird anhand der gelieferten Landmarks die Parameter für die Audioeffekte und Samples generieren. Die benötigte Kalibrierungssequenz für wechselnde Nutzer gehört ebenfalls zu diesem Aufgabengebiet.

Sounds :

Alina Böttcher übernimmt die Auswahl unserer Soundsamples sowie die benötigten Klassen für das Abspielen von Sounddateien in unserem Projekt. Zusätzlich realisiert sie das Benutzerinterface und die Wechselfunktion zwischen unseren Anwendungsmodi.