Projektdokumentation

Der Gesichtsmusikant

von:

Linh Do

Marcel Plaga

Marvin Steinmetz

Alina Böttcher

1. Aufgabe des Projektes

Mit der Software „Gesichtsmusikant“ wird über Gesichtstracking die Bewegung des Gesichtes eingefangen und mit Geräuschen wiedergegeben.

Dabei kann zwischen zwei Modi entschieden werden. Zum einen ist da der Sample Modus. Darin werden abgeschlossene Bewegungen gedeutet und dann ein kurzer Sound abgespielt. So ruft z.B. das Öffnen des Mundes das Trompeten eines Elefanten hervor.

Zum anderen ist da der Synthesizer Modus. Dieser ermöglicht es mithilfe des Gesichtes mit einem Ton zu spielen. Dazu wird auf dem Bildschirm ein Synthesizer eingeblendet, dessen Hebel sich mit der Gesichtsbewegung verändern.

1. Bedienungsanleitung

Für den Synthesizer Modus muss zusätzlich noch Pure Data gestartet werden. Siehe Installationsanleitung.

|  |  |  |
| --- | --- | --- |
| Gesichtsbewegung | Sample Modus | Synthesizer Modus |
|  |  |  |
| Augenbrauen hoch | Hupe |  |
| Mund auf | Adler |  |
| Kopf links/rechts rotieren | Mücke/Klirren |  |
| Kopf vorne/hinten bewegen | Elefant/Ente |  |
| Kopf links/rechts bewegen | Ozean/Blubbern |  |

1. Installationsanleitung

**Pure Data:**

Es wird die Pure Data Extended Portable benötigt. Diese kann unter folgendem Link heruntergeladen werden.

<http://sourceforge.net/projects/pure-data/files/pd-extended/0.43.4/Pd-0.43.4-extended-windowsxp-i386.zip/download>

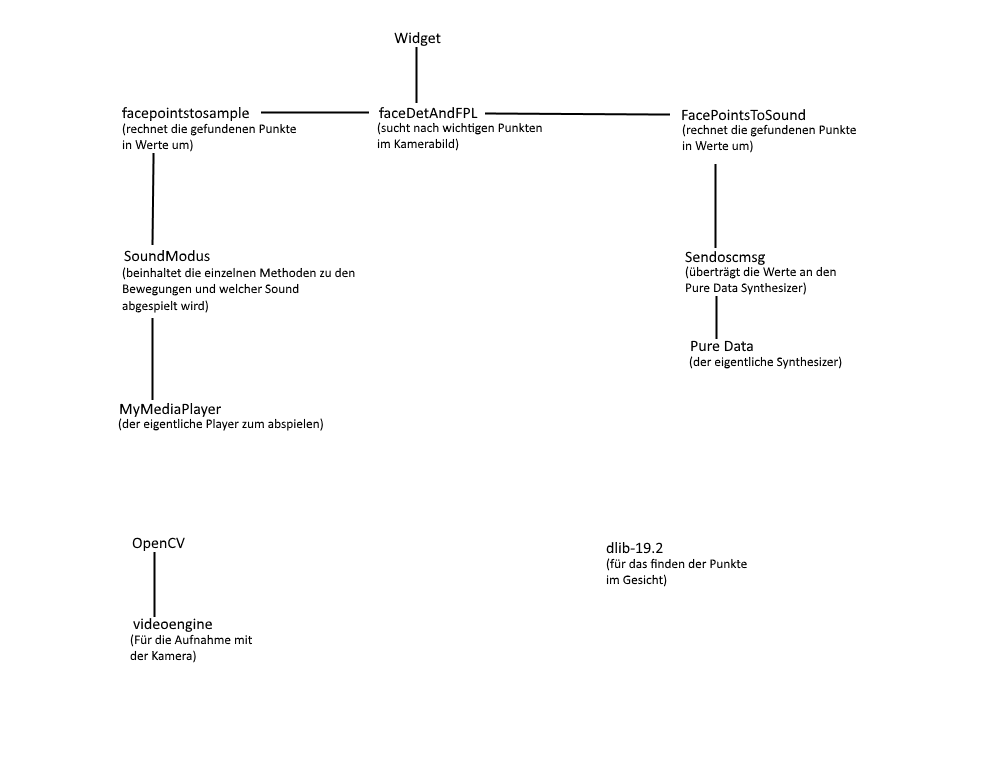
Die heruntergeladene Datei muss entpackt werden. Im Entpackten Ordner befindet sich eine Datei mit dem Namen: pd-extended.bat

Diese muss durch die gleichnamige Datei aus dem Programmordner ersetzt werden.

Mit einem Doppelklick auf die ersetzte Datei wird Pure Data gestartet.

Jetzt muss mit dem sich öffnenden Programm die ebenfalls im Programmordner befindliche Datei: Gesichtsmusikant.pd geöffnet werden.

1. Systemarchitektur



1. Beschreibung eines technischen Teilaspektes

* man könnte die Parameterberechnung erklären??

mouthOpen = (double)

((shapes[0]. part(62)- shapes[0]. part(66)). length())

\* 100 /

((shapes[0]. part(0) - shapes[0]. part(16)). length()\* ratioMouth);

1. Auswertung:
   1. Was hat funktioniert

Ziemlich zuverlässig hat dlib gearbeitet. Nur bei sehr schlechten Lichtverhältnissen und bei zu üppigen Bärten hat es Erkennungsprobleme gegeben.

Ebenfalls gut funktioniert hat die Zusammenarbeit der Computer. Wir hatten keine Versionsprobleme oder irgendwelche anderen Probleme die bei der Zusammenarbeit mit mehreren Computern und mehreren Programmen entstehen können.

* 1. wo waren die Grenzen

Eine große Hürde waren die fehlenden c++ Kenntnisse.

So ist z.B. häufig ist der Fehler aufgetaucht, dass das komplette Programm abgestürzt ist, wenn man keine Pointer verwendet hat.

Ein anderes Beispiel wäre, dass die Überlegung da war, das die Überwachung der Punkte in einem Thread läuft und das umrechnen und ausgeben in einem anderen. Das beides sozusagen unabhängig voneinander läuft. Leider ist es nicht gelungen das zu implementieren.

Es ist immer der Fehler aufgetaucht, dass er ein & erwartet obwohl eines vorhanden war.

Dadurch, dass wir eine Aufgabe hatten, die für uns alle absolutes Neuland war, war vorher schlecht abzuschätzen, wie die genaue Klassenstruktur aussehen wird. Das hat dazu geführt, dass wir zwischenzeitlich etwas Chaos hatten, welche Klasse für was zuständig ist.

* 1. was würden wir beim nächsten Mal anders machen

Durch andere Projekte ist das Projekt zum Anfang hin etwas vernachlässigt worden. Eine bessere Zeitplanung wäre für das nächste Mal ein gutes Ziel.