MIDI MIMIK

Gesicht erkannt

wählt Instrument aus

Musik startet

Zwinkern links

Zwinkern rechts

wendet Effekt an

FX

Nicolas Graf, Dennis Kruschel, Tobias Lamping, Nikolas Baumgartner, Antonia Schwab

Nicolas Graf (Mat-Nr. 2094723) , Dennis Kruschel (Mat-Nr. 2148336) , Tobias Lamping (Mat-Nr. 2217288),
Nikolas Baumgartner (Mat-Nr. 2175418), Antonia Schwab (Mat-Nr. 2202193)

Inhaltsverzeichnis

Projektziel

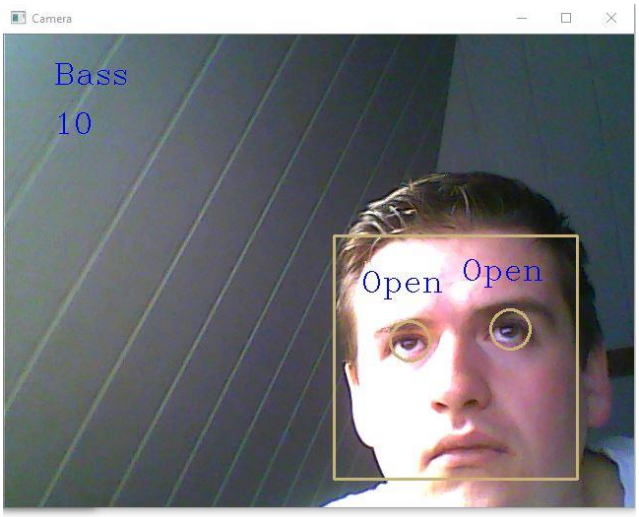
Bei dem Projekt handelt es sich um einen Synthesizer, welcher mit dem Gesicht gesteuert werden kann. Mithilfe von bestimmten Gesichtsausdrücken können so beispielweise Instrumente hinzugeschaltet oder auch lauter bzw. leiser gestellt werden. Das Ziel des Projekts besteht hauptsächlich daraus, herauszufinden, was man für so eine Funktion benötigt und wie man sie realisiert, um eine Basis für ein Programm zu entwickeln.

Bedienungsanleitung

Für das Programm werden folgende Geräte benötigt:

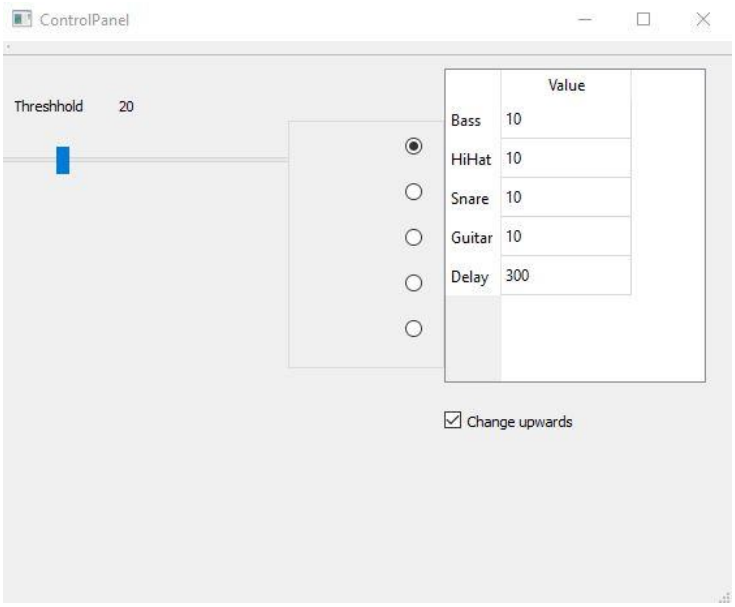
- Desktop-PC/Laptop
- Kamera
- Lautsprecher

Beim Start des Programms erscheinen folgende Fenster:



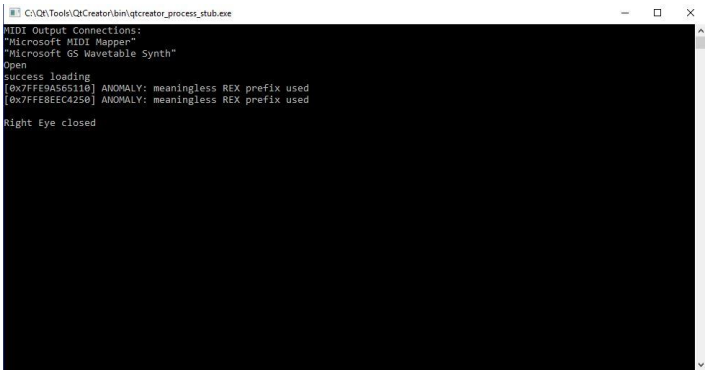
Anzeige der Face-Cam

Das Rechteck symbolisiert , ob das Programm das Gesicht erkennt. Anhand der Kreise, sieht man, wo das Programm die Augen findet. Über jedem Auge befindet sich außerdem das Wort „open“ oder „closed“, je nachdem ob das Auge offen oder geschlossen ist. In der Ecke oben links befindet sich der Name des aktuell ausgewählten Instruments und dessen Lautstärkewert.



Steuerpult

Das Steuerpult (ControlPanel) zeigt alle Einstellungsmöglichkeiten.



Konsole mit Informationen

Hier werden Informationen auf der Konsole ausgegeben. Zum Beispiel, welches Auge geschlossen wurde usw. Diese Konsole dient eher zu Testzwecken und ist nicht für den Endbenutzer vorgesehen.

Ablauf und Bedienung des Programms:

Sobald ein Gesicht erkannt wird, startet ein leiser Musiktrack. Das ControlPanel auf der linken Seite zeigt verschiedene Instrumente, die man auswählen kann. Schließt man das rechte Auge, wechselt man zwischen den Instrumenten. Um die Lautstärke, des aktuell ausgewählten Instruments zu ändern, schließt man das linke Auge.

Mit der Checkbox „Change Upwards“, die man nur per Mausklick bedienen kann, wird bestimmt, ob die Lautstärke für ein bestimmtes Instrument hoch oder

runtergedreht wird. Bei aktivierter Checkbox wird der Ton lauter. Ist die Checkbox deaktiviert, wird der Ton leiser.

Mit dem Delay bestimmt man die Geschwindigkeit. Der Wert gibt dabei die Pausen zwischen den einzelnen Noten an. Je höher der Wert ist, desto langsamer wird der Musiktrack abgespielt.

Die Wertigkeit der Lautstärke wird in 10ner Schritten, die des Delays in 25er Schritten erhöht. Wird der maximale Wert (bei Lautstärke 120/beim Delay 400) erreicht, so wird dieser beim höher drehen auf den minimalen Wert gesetzt (bei der Lautstärke auf 10/beim Delay auf 200).

Der Threshold Wert, der nur per Mausklick bestimmt werden kann, ermöglicht es die Feineinstellung für die Augenerkennung zu justieren. Je nach Lichtverhältnissen kann hier der Schwellenwert zwischen Schwarz und Weiß eingestellt werden, mit dem das Programm erkennt ob ein Auge offen oder geschlossen ist.

Installationsanleitung

Um das Programm zu entwickeln braucht man folgende Schritte:

1. QT Creator herunterladen und installieren
2. OpenCV herunterladen und installieren
3. Pfade zu OpenCV Bibliotheken in Projektdatei anpassen

4. Pfade der Haar Kaskaden in OpenCV

5. Programm kann nun angepasst und gestartet werden

Beschreibung eines technischen Teilaspektes

Der folgende Quelltext beschreibt einen Teilaspekt, um ein geschlossenes Auge zu erkennen:

```
double getEyeQuotient(Mat equalizedInput){
    //Rechenwerte anlegen
    int numBlackRows = 0;
    int numBlackRowsMax  = 0;
    int numBlackCols  = 0;

    //Für alle Spalten
    for(int x = 0; x < equalizedInput.cols; x++){
        if (numBlackRows >= 1){
            numBlackCols++;
        }
        numBlackRows = 0;
        //Für alle Reihen
        for(int y = 0; y < equalizedInput.rows; y++){
            //Unser Pixel an der Stelle
            Vec3b inputPixel = equalizedInput.at<Vec3b>(y, x);
            //inputPixel beinhaltet Werte für alle 3 Farben, wir brauchen aber nur eine da wir ein Binärbild kriegen
            int blue = inputPixel[0];

            //Falls der Pixel schwarz ist
            if(blue == 0){
                numBlackRows++;
            }
            if (numBlackRows > numBlackRowsMax){
                numBlackRowsMax = numBlackRows;
            }
        }
    }

    //Quotienten anlegen und größeren Wert durch den kleineren teilen
    double eyeQuotient;

    if(numBlackRowsMax >= numBlackCols){
        eyeQuotient = (double)numBlackCols/numBlackRowsMax;
    } else{
        eyeQuotient = (double)numBlackRowsMax/numBlackCols;
    }
    return eyeQuotient;
}
```

Das Prinzip dahinter ist ganz einfach. Es wird im Hintergrund ein Binärbild von schwarzen Pixeln erstellt. Anhand dieser Werte wird ein Quotient ausgerechnet und zurückgegeben. Folgende Bilder zeigen ein Beispiel:

Auge ist geschlossen

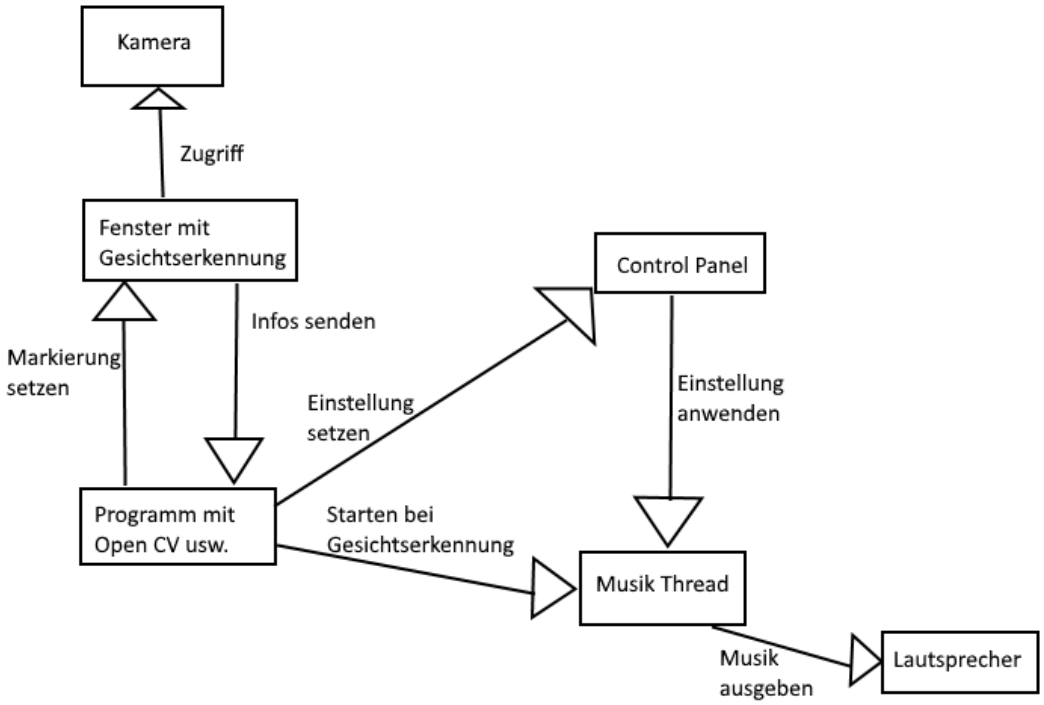
Auge ist geöffnet

Je mehr schwarze Pixel zusammen sind, desto höher ist der Quotient und das Auge ist wahrscheinlich geöffnet (linkes Bild). Verteilen sich die schwarzen Pixel zu einer schmaleren Linie, wird der Quotient kleiner und das Auge ist wahrscheinlich geschlossen.

Die Berechnung des Quotienten wird für jedes Auge einzeln ausgeführt. Gibt die Berechnung einen Quotienten von 0,3 oder kleiner zurück, so zählt das jeweilige Auge als geschlossen. Andernfalls ist es geöffnet.

Systemarchitektur

Folgendes Diagramm zeigt die grobe Systemarchitektur von MIDI MIMIK.



Grobe Systemarchitektur

Folgende Frameworks/Komponenten werden verwendet:

- OpenCV und Cascades (Bildbearbeitung bzw. Gesichtserkennung).
- Midi (Für die Wiedergabe des Tons usw.)
- Kamera (Damit Gesicht überhaupt erkannt werden kann)
- Lautsprecher (Ausgabe des Midi-Tracks)

Wird das Programm gestartet, öffnen sich neben der Konsole zwei Fenster. Ein Fenster greift auf die Kamera zu und erkennt ein Gesicht. Dem Programm wird dann mitgeteilt, wo sich ein Gesicht befindet und ob Augen erkannt werden. Anschließend setzt das Programm die jeweiligen Markierungen mithilfe von Open CV. Außerdem gibt es aus, ob ein Auge offen oder geschlossen ist.

Wird ein Gesicht erkannt, so startet das Programm den Musik Thread, welcher eine MIDI-Melodie abspielt und über die Lautsprecher ausgibt. Vom ControlPanel werden Einstellungen wie Lautstärke einzelner Instrumente oder die Geschwindigkeit des gesamten Soundtracks übergeben.

Das ControlPanel wiederum wird vom Programm mithilfe der Gesichtserkennung gesteuert. Schließt der Benutzer zum Beispiel das linke Auge, bekommt das Programm diese Information und verändert damit die Einstellung im ControlPanel. In diesem Fall wird die Lautstärke des aktuell ausgewählten Instruments erhöht.

Auswertung

Was hat (gut) funktioniert:

Das Abspielen der verschiedenen Töne mit MIDI war leicht in dem Projekt einzubinden und hat somit auch die wenigsten Probleme bereitet.

Was hat nicht so gut funktioniert/Wo liegen die Grenzen:

Die größten Schwierigkeiten in dem Projekt lagen darin, die Gesichtserkennung einzubinden. Open CV alleine hat dafür nicht gereicht. Es mussten zusätzlich noch Kaskaden für das Gesicht und für die Augen eingebunden werden. Erst dann konnte das Programm Gesicht und Augen erkennen. Um zu erkennen, ob ein Auge geschlossen oder geöffnet ist, musste allerdings auch noch ein Binärbild im Hintergrund erstellt und ausgelesen werden. Dann sollte das Programm theoretisch erkennen, ob ein Auge geschlossen oder geöffnet ist.
Beim Testen stellte sich aber heraus, dass dies nicht bei jedem Gruppenmitglied einwandfrei funktionierte. Sobald ein Auge geschlossen war, wurden entweder keine Augen mehr oder das geschlossene Auge noch als geöffnet erkannt.
An der genauen Erkennung liegen momentan also noch die Grenzen.

Was könnte man beim nächsten Mal anders machen:

Bei einem zukünftigen Projekt sollten die einzelnen Klassen geplant werden, bevor man mit dem Schreiben des Codes beginnt. So hat man einen sauberen Quellcode und die Übersichtlichkeit wird enorm gesteigert. Außerdem kann man das Programm dann noch viel einfacher erweitern und Fehler schneller finden.