Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Alexander Eggers (2189213)
Lennart Przygode (1845165)
Nadine Winter (2191707)
Ninja Rüsch (2203425)

Projektdokumentation

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Alexander Eggers (2189213)
Lennart Przygode (1845165)
Nadine Winter (2191707)
Ninja Rüsch (2203425)

Inhaltsverzeichnis Seite

Hochschule für Angewandte
Wissenschaften Hamburg
Hamburg University of Applied Sciences

Alexander Eggers (2189213)
Lennart Przygode (1845165)
Nadine Winter (2191707)
Ninja Rüsch (2203425)

1. Aufgabe des Projektes

Das Ziel des Projektes war es ein Spiel umzusetzen, welches unterschiedliche Gesichtsgesten verwendet, womit die Spieler virtuell ein Glas mit Wasser füllen können. Um möglichst genaue Ergebnisse bei der Gesichtserkennung zu erhalten, wollten wir eine Dlib Implementierung verwenden. Innerhalb des Spiels sollten zwei Spieler (Teams) gegeneinander antreten können. Jeder Spieler sollte einen Sound zu hören bekommen, welches als Referenz dienen sollte. Wenn der jeweilige Spieler an der Reihe war, musste dieser diesen Sound versuchen anhand der Lautstärke nachzustellen. Gewonnen hatte der Spieler, welcher am dichtesten am original Sound war.

Die verfügbaren Gesichtsgesten sind:

- **Mund auf**: Die Lautstärke des eigenen Sound wird lauter bzw. Wasser wird in das Glas gelassen
- **Mund zu**: Lautstärke wird nicht weiter geändert bzw. es wird kein Wasser mehr in das Glas gelassen

Über die GUI sollte der Spieler durch das Spiel geführt werden sodass dieser eine kurze Einführung erhält und am Ende des Spiels das Ergebnis des Spiels.

2. Bedienungsanleitung

Wird das Projekt in QT ausgeführt öffnet sich ein Fenster. Das erste Fenster zeigt das Startbild mit dem Namen des Spiels. Durch den Button “Next” unterhalb des Bildes gelangt der Spieler zu dem nächsten Fenster.
Auf dieser ist eine Kurzanleitung, wie das Spiel funktioniert. Hat der Spieler die Anleitung gelesen bestätigt er dieses wieder durch das Klicken auf den “Next” Button.
Nun wird der erste Spieler gefragt, ob er bereit ist das Spiel zu starten. Dafür muss dieser Kopfhörer aufsetzen und anschließend durch die Betätigung des “Play Game Sound” Buttons das Musikstück abspielen. Zu hören ist das Geräusch, wenn Wasser in ein Glas gefüllt wird. Das Geräusch wird in einer bestimmten Lautstärke abgespielt, die sich der Spieler merken muss. Ist dieser der Meinung sich die Lautstärke eingeprägt zu haben, gelangt er über “Confirm” auf das nächste Fenster.

Alexander Eggers (2189213)
Lennart Przygode (1845165)
Nadine Winter (2191707)
Ninja Rüsch (2203425)

Durch das öffnen des “Start facial detection” Button startet die Webcam und das Gesicht des Spielers wird erkannt. Öffnet dieser seinen Mund wird das Wassergeräusch anfangs mit der niedrigsten Lautstärke abgespielt und erhöht sich je länger der Mund offen ist. Beim Schließen des Mundes stoppt die Wiedergabe des Geräusches und fängt wieder bei der letzten Lautstärke wieder an sobald der Mund wieder geöffnet wird. Meint der Spieler die zuvor gehörte Lautstärke erreicht zu haben, muss er das Webcam- Fenster schließen und mit “Confirm” bestätigen.
Nun ist der zweite Spieler an der Reihe. Dieser kann sich ebenfalls das Geräusch mit der zu erreichenden Lautstärke anhören und den versuch wagen diese durch die Gesichtserkennung zu erreichen.

Hat der zweite Spieler seinen Wert gespeichert erfolgt in dem nächsten Fenster die Auswertung. Die Werte werden mit dem vorgegebenen Wert der Lautstärke verglichen. Wessen Wert näher an diesem ist gewinnt. Auf dem Fenster steht welcher Spieler gewonnen hat. Haben die Spieler den gleichen Wert gespeichert wird ein Unentschieden angezeigt.

Anschließend kann das Spiel durch “Restart” von Neuem gestartet oder durch “Quit” beendet und gleichzeitig geschlossen werden.

Alexander Eggers (2189213)
Lennart Przygode (1845165)
Nadine Winter (2191707)
Ninja Rüsch (2203425)

Clickflow

3. Installationsanleitung

→Downloadlinks im Quellenverzeichnis

Für die Entwicklung des Projektes in C++ braucht man das Qt-Framework, Visual Studio 2015 (den Compiler vc14) und OpenCV zur Bild- und Videobearbeitung. Bei der Installation von Visual Studio 2015 muss man bei der benutzerdefinierten Installation darauf achten, dass man das Package für C++ inklusive den C++ Compiler mit ausgewählt hat. Als Kontrolle kann man die Größe der Installationsdatei von ca 9GB als Anhaltspunkt nehmen.

Alexander Eggers (2189213)
Lennart Przygode (1845165)
Nadine Winter (2191707)
Ninja Rüsch (2203425)

Hat man diese heruntergeladen und installiert bzw. entpackt muss man nun noch die Umgebungsvariable für OpenCV einstellen, indem man eine neue Variable (OPENCV_DIR) definiert und den Pfad zum Build-Verzeichnis hinzugefügt.
Für die Gesichtserkennung muss man nun noch eine OpenCV C++ Library installieren namens Dlib. Um diese zu installieren benötigt man Cmake (cross-platform make) auf seinem Rechner, falls noch nicht vorhanden. Nun folgen ein paar Eingaben in der Konsole in dem Dlib-Verzeichnis.

```
mkdir build
cd build
cmake .. -G"Visual Studio 14 2015 Win64" -DCMAKE_INSTALL_PREFIX=D:\dlib_build
cmake --build . --config Release --target install
```

Hat man diesen Build erfolgreich abgeschlossen muss man in der Projekt .pro Datei des in Qt erstellten Projektes noch den Pfad der Dlib hinzufügen.

```
QT += core
QT -= gui

CONFIG += c++11

TARGET = dlibWin2
CONFIG += console
CONFIG -= app_bundle

TEMPLATE = app

SOURCES += main.cpp

INCLUDEPATH += "D:\dlib_build\include"
LIBS += -L"D:\dlib_build\lib" -ldlib
```

Um nun ein Gesicht zu tracken und im Gesicht die einzelnen Partien Mund, Augen, Nase erkennen zu können, verwenden wir ein trainierbares Objekt “Shape Predictor”, welches bis zu 68 Messpunkte im Gesicht setzen kann. Diesen Predictor kann man mit Hilfe des Links im Quellenverzeichnis herunterladen, extrahieren und den Pfad in der jeweiligen .cpp Datei einfügen, in der man die Gesichtserkennung nutzen möchte.

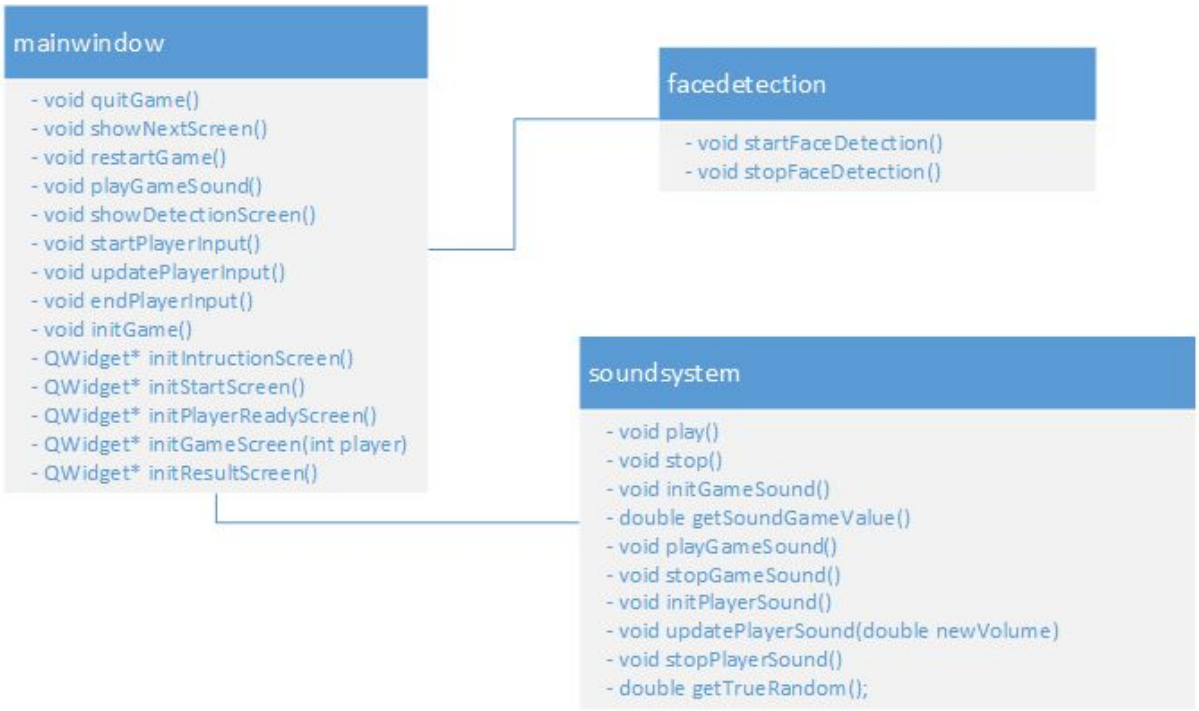
Alexander Eggers (2189213)
Lennart Przygode (1845165)
Nadine Winter (2191707)
Ninja Rüsch (2203425)

4. Systemarchitektur

Die Klasse mainwindow.c stellt in dem Projekt den Mittelpunkt für alle möglichen Operationen dar. Von dieser Klassen werden alle Referenzen zu den Anderen erstellt. Innerhalb der mainwindow.c wird ein QStackedWidget verwendet um die unterschiedlichen GUI Fenster umzusetzen. Vorteil dieser Implementierung ist, dass es nicht nötig ist Werte zwischen den GUI Fenstern zu verschieben. Jedes Fenster hat im mittleren Bereich ein QLabel, welches ein Bild hält. Unter dem Bild sind Buttons um die Navigation innerhalb des Spiels sicherzustellen. Die Buttons sind mit der QT connect-Funktion umgesetzt.

Um den Sound innerhalb des Spiels abspielen zu können, wurde im mainwindow.c eine Referenz zur soundsystem.c erstellt. Diese sollte unter anderem einen zufälligen double errechnen, welcher als Referenz Wert eingesetzt werden sollte. Dieser Referenzwert ist das Spielziel, welches Spieler versuchen sollen zu erreichen. Des Weiteren hat diese Klasse Funktionen um zwischen den Spieler Sound und den Referenz Sound zu differenzieren. Dies ist wichtig für das Abspielen des richtigen Sounds.

Die Gesichtserkennung ist über die facedetection.c über die mainwindow.c verbunden. Der Hauptteil dieser Klasse ist über die startFaceDetection() umgesetzt, welches den eigentlichen Prozess startet. Die stopFaceDetection() sollte diesen Prozess wieder anhalten.



Alexander Eggers (2189213)
Lennart Przygode (1845165)
Nadine Winter (2191707)
Ninja Rüsch (2203425)

5. Technischer Teilaspekt - Gesichtserkennung

Wie zuvor schon beschrieben, nutzen wir für die Gesichtserkennung die Dlib-Bibliothek. Dabei wird jeder Frame jedoch zunächst mit Hilfe von OpenCV aus dem VideoCapture (wie Webcam o. Ä.) als Mat-Objekt geliefert. Dieses wird daraufhin in ein cv_image Objekt der Dlib-Bibliothek umgewandelt, mit welchem erst die eigentliche Gesichtserkennung stattfinden kann.

Über einen frontal_face_detector (auch von Dlib geliefert) werden dann zunächst alle Gesichter im Frame erkannt und die Positionen in einem Array aus Rechteck-Koordinaten gespeichert. Über diesen Array kann nun mit dem zuvor schon beschriebenen Shape-Predictor (einem trainierten Modell, welches für die Erkennung von Gesichtsposen und der 68 Landmarks zuständig ist) iteriert werden. Der Predictor liefert dabei die Positionen der Landmarks in 68 Parts als sogenannte Shapes wieder, welche pro Part über X- und Y-Koordinaten verfügen. Auf jedes dieser Parts lässt sich direkt zugreifen, welches uns die Berechnung der Distanz verschiedener Landmarks ermöglicht. Für die Erkennung des offenen Mundes benötigen wir z. B. Part 60 und 64 (die Mundwinkel links und rechts) sowie die Parts 62 und 66 (die zentralen Landmarks des Mundes im oberen und unteren Bereich). Um das Öffnen und Schließen des Mundes auch unabhängig von der Auflösung der Kamera zu realisieren, bedienen wir uns der Berechnung eines Ratios des Verhältnisses von Breite und Höhe des Mundes. Hierbei wird die Höhe des Mundes durch die Breite geteilt und geht demnach gegen Null, wenn dieser geschlossen ist. Ab wann der Mund als offen gilt, haben wir durch Erfahrungswerte bestimmt (in der endgültigen Version lag dieser Wert bei 0.14). Diese relativ naive Variante funktioniert sehr performant.

Zusätzlich wurde ein Trigger implementiert, der erst zum Abspielen des Soundfiles bzw. Senden des entsprechenden Events führt, sobald der Mund einmalig mehrere Frames hintereinander als Offen erkannt wurde. Dies ist explizit nötig, um ein mehrfaches starten des gleichen Audiofiles zu unterbinden. Erst wenn der Mund einmal geschlossen wurde, ist das Neustarten des Audiofiles durch das Öffnen des Mundes wieder möglich. Um jedoch auch die Spiellogik zur korrekten Berechnung des Wasserstandes zu befähigen, wird ein zweiter Update-Event gefeuert, welcher nicht an den zuvor beschrieben Trigger gebunden ist.

Alexander Eggers (2189213)
Lennart Przygode (1845165)
Nadine Winter (2191707)
Ninja Rüsch (2203425)

6. Auswertung

Die Einbindung der Dlib-Bibliothek erwies sich schwerer als erwartet. Erst nach dem eigenen Kompilieren, konnte diese wie in der Dokumentation beschrieben genutzt werden. Dazu musste jedoch auch der richtige Compiler (Visual Studio 2015 64-Bit) installiert werden, was uns nicht von Anfang an klar war. Die ersten Versuche mit dem alternativen MinGW Compiler verliefen erfolglos und raubten uns einige Stunden. Auch wenn dieses Problem schlussendlich durch den richtigen Compiler gelöst werden konnte, mussten wir diese Lösung zusätzlich auf allen beteiligten Rechnern umsetzen, bevor wir tatsächlich als Gruppe weiter arbeiten konnten.

Da die Gesichtserkennung über Dlib den Main Thread blockiert, mussten wir auf das Umlegen der Audioausgabe in einem eigenen Thread ausweichen. Dieses Blockieren führt jedoch auch dazu, dass der Spielablauf nicht genau so ist, wie geplant. Man muss nun jedes mal das Webcam Fenster schließen, bevor man das Spiel fortsetzen kann.

Auch die Erzeugung und Manipulation des Wasser-Sounds, konnten wir nicht nach unseren Vorstellungen umsetzen. Das Abspielen des aufgenommenen Wassers Klangs war zunächst nur als ein Platzhalter implementiert, den wir jedoch bis zum Ende hin beibehalten haben. Genauso verhält es sich mit der simplen Manipulation durch die Erhöhung der Lautstärke.

Nichtsdestoweniger sind wir mit dem Ergebnis zufrieden und haben einiges über die Tücken der C++ Programmierung kennen gelernt. Insbesondere die Problematik mit den unterschiedlichen Compilern haben wir als Java-Programmierer so noch nicht erlebt.

Alexander Eggers (2189213)
Lennart Przygode (1845165)
Nadine Winter (2191707)
Ninja Rüsch (2203425)

7. Quellen, Links

Github Repository “Fill Your Glass”: https://github.com/Mordag/fill-your-glass

Installations Links:
1. Visual Studio 2015 Community: https://www.visualstudio.com/downloads/
2. Qt Framework: https://www.qt.io/download-open-source/
3. OpenCV: http://opencv.org/downloads.html
4. Dlib: http://dlib.net/ (blauer download-Button der zip Datei)
5. CMake: https://cmake.org/
6. Shape Predictor: http://dlib.net/files/shape_predictor_68_face_landmarks.dat.bz2

Literatur- und Bildnachweis:
1. Compile dlib on Windows, *[aufgerufen zuletzt am 24.01.2017],* https://forum.qt.io/topic/69007/compile-dlib-on-windows