Technische Dokumentation: Formenklang

für den Kurs Audio-Video-Programmierung bei Prof. Dr. Andreas Plaß & Jakob Sudau
von Nadine Krietenbrink (2322554) und Caroline Wolf (2320493)

Kurzbeschreibung

Formenklang ist ein interaktives Audio- und Videoprogramm zur Erstellung von Audio-Sequenzen. Mit Hilfe einer Kamera werden Formen mit der jeweiligen Farbe erkannt, weitergeleitet und im Webbrowser entsprechend der Zeitachse abgespielt.

Technisches Setup

Um das Projekt nutzen zu können, benötigt man ein bestimmtes Setup. Für die Videoaufnahme ist eine Webcam erforderlich, die mit dem Rechner verbunden ist. Auf dem Rechner müssen QT und ein Quelltext-Editor installiert sein. Der Quelltext-Editor sollte die Möglichkeit haben einen Live Server zu starten. Zur Übertragung der MIDI-Daten wird zusätzlich das Programm LoopBe benötigt. Die Wiedergabe auf dem Rechner erfolgt im Web-Browser Chrome.

Technische Umsetzung

Im folgenden Klassendiagramm sind die wichtigsten Variablen und Methoden der einzelnen Klassen definiert. Zur Erhaltung der Übersichtlichkeit sind unwichtige Variablen nicht einbegriffen.

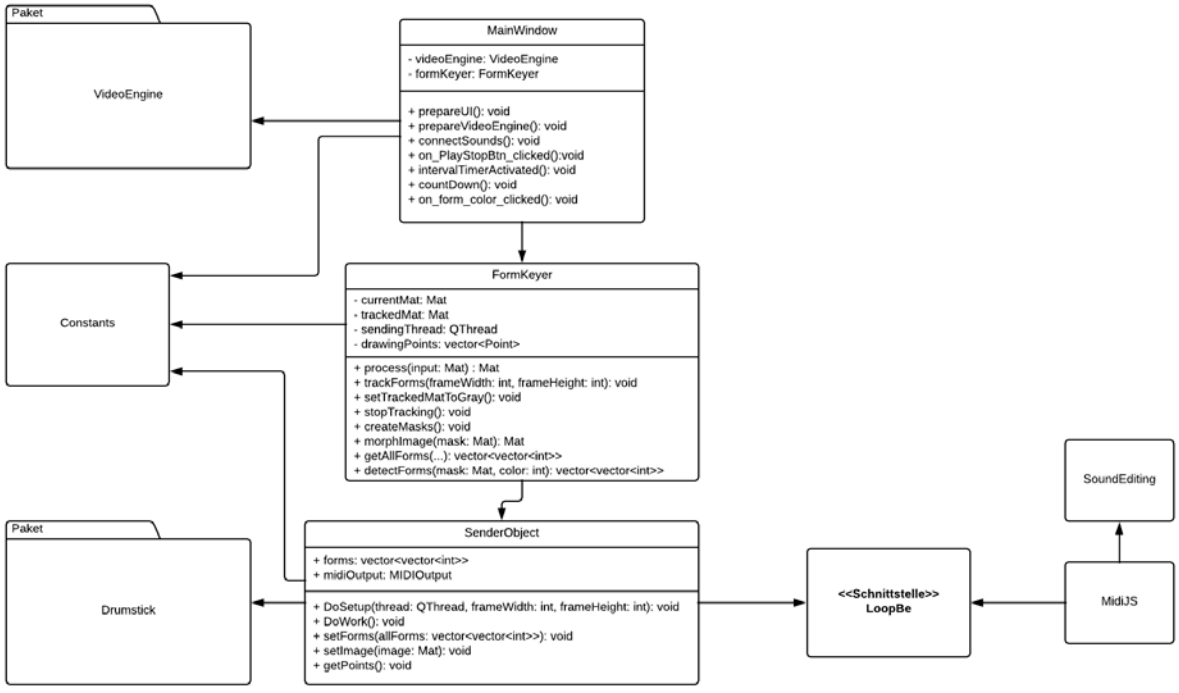

Abbildung 1Klassendiagramm Formenklang

VideoEngine:

Das Paket VideoEngine dient zur Verarbeitung und Darstellung des Kamerabildes.

Constants

Diese Klasse beinhaltet alle Konstanten, welche in dem Projekt benötigt werden. Dazu gehören die HSV-Werte der einzelnen Farben, die Farb- und Formdefinition und die Intervallzeiten für die Timer.

Drumstick

Dieses Paket wird genutzt, um die MIDI-Daten zu verschicken.

MainWindow:

Die MainWindow-Klasse kümmert sich um das UI, die Interaktionen, die Verknüpfung der Sound-Dateien, das Starten der VideoEngine und das Timing des Programmes. Sie wird beim Starten des Programms ausgeführt.

- **prepareUI() :** Ausblenden des Timers in der Oberfläche
- **prepareVideoEngine():** Einstellungen zur VideoEngine und starten
- **connectSounds():** Verknüpft die Sounds, welche beim Klicken auf eine Form abgespielt werden
- **on_playStopBtn_clicked():** Funktion, die ausgeführt wird, wenn der „Tracking starten/stoppen"-Button geklickt wird. Sie unterscheidet, ob das Tracking bereits aktiviert ist. Je nachdem wird der Timer „intervalTimer" gestartet oder gestoppt. Dieser Timer führt alle 5 Sekunden die Funktion „intervalTimerActivated" aus.
- **intervalTimerActivated():** Diese Funktion aktiviert die Funktion trackForms des Objekts formKeyer. Dadurch wird das Tracking alle 5 Sekunden wiederholt. Zusätzlich startet die Funktion einen weiteren Timer, der jede Sekunde abgefeuert wird. Dieser dient zur Darstellung des Countdowns auf der Oberfläche.
- **countDown():** Countdown-Funktion zur Darstellung des Timers auf der Oberfläche.
- **on_<form>_<color>_clicked():** <form> und <color> sind mit dem jeweiligen Element auszutauschen. Diese Funktionen dienen zum Abspielen des Sounds der jeweiligen Form bei Mausklick.

FormKeyer:

Die FormKeyer-Klasse dient der Formenerkennung. Sie wird aus der MainWindow-Klasse heraus aktiviert. Die Frames zur Verarbeitung werden durch die VideoEngine in der Funktion process geliefert.

- **process():** Bereitet die Ausgabe für das getrackte Bild vor. Wenn das Tracking noch nicht gestartet ist, liefert es ein graues Bild. Wenn es gestartet ist, liefert es das getrackte Bild. Zusätzlich zeichnet die Funktion die Punkte der bereits abgespielten Formen an der entsprechenden Position. Die Daten dazu bekommt sie durch eine getter-Methode von SenderObject.
- **trackForms():** Diese Funktion wird durch den intervalTimer aus der MainWindow-Funktion alle 5 Sekunden ausgeführt, wenn das Tracking aktiv ist. Sie bereitet das senderObject vor und verknüpft es mit dem Thread, welcher für das Timing und Senden der MIDI-Daten zuständig ist. Sie nimmt sich das aktuelle Frame, welches in der process-Funktion aktualisiert wird, und führt alle notwendigen Funktionen durch, um Form und Farbe zu erkennen. Diese Informationen übergibt sie dann dem senderObject und startet den Thread.
- **setTrackedMatToGray():**
  Wenn das Tracking gestoppt wird, wird mit Hilfe dieser Funktion das getrackte Bild wieder auf ein graues Bild gesetzt.
- **stopTracking():**
  Beendet den Thread, wenn das Tracking gestoppt wird.
- **createMasks():**
  Erstellt für die Farben Rot, Grün, Blau und Gelb jeweils ein 1-Bit Bild im Rahmen der angegebenen HSV-Werte. Diese Mat-Objekte übergibt die Funktion anschließend der morphImage-Funktion.

- **morphImage():**
  Wendet die Funktion morphologyEx mit dem Wert MORPH_OPEN auf das Mat-Objekt an. Dadurch werden kleine Störungen aus dem Bild entfernt. Das structuringElement bestimmt dafür das Element, mit dem über das Bild gegangen wird.
- **getAllForms():**
  Führt für jede Farb-Maske die Funktion detectForms durch und sammelt anschließend alle erfassten Formen in einem Array.
- **detectForms():**
  Dient zur Erkennung der Formen. Sie bekommt ein 1-Bit Bild übergeben und die zugehörige Farbe. Auf das Bild wird die Methode findContours angewandt und die Punkt-Sammlungen der einzelnen Konturen in das Array contours gespeichert. Anschließend werden in einer for-Schleife alle Konturen analysiert. Die boundingRect-Funktion legt ein Rechteck um die Kontur, wodurch der Mittelpunkt der Kontur berechnet werden kann. Die approxPolyDP-Funktion approximiert ein Polygon aus den Punkten der Kontur. Dieses Polygon besteht aus wenigen Vertices, die die Form beschreiben. Anhand der Anzahl der Vertices kann die Form festgelegt werden. Bei 3 Vertices ist es ein Dreieck, bei 4 ein Rechteck und bei 5 ein Pentagon. Der Form-Wert und die Position werden in dem forms-Array gespeichert.

SenderObject

Diese Klasse wird durch den Thread ausgeführt. Sie ist für das Abschicken der MIDI-Daten passend zur Position der Form auf der Zeitachse zuständig.

- **DoSetup():**
  In dieser Funktion werden alle Vorbereitungen zum Versenden der MIDI-Daten getroffen. Sie baut eine Verbindung zum MIDI-Output auf und wählt LoopBe zur Übertragung aus. Des Weiteren wird die Verbindung mit dem Thread aufgebaut.
- **DoWork():**
  Diese Funktion wird durch die trackForms-Funktion aus der FormKeyer-Klasse abgerufen. Sie sortiert das übergebene forms-Array nach dem x-Wert. In der for-Schleife wird dann jede einzelne Form durchgegangen. Die Wartezeit wird berechnet aus bereits gewarteter Zeit und der insgesamt zu wartenden Zeit. Dieser Wert wird dann mittels der msleep-Funktion des Threads gewartet. Erst dann wird der MIDI-Datensatz losgeschickt. Zusätzlich wird noch der Mittelpunkt der Form als Punkt abgespeichert, damit die process-Funktion aus der FormKeyer-Klasse diesen Punkt abrufen und zeichnen kann.

Midi.js

In dieser Datei wird Midi eingerichtet, die Sounds werden geladen und die Visualisierung der Sounds im Browser bestimmt.

- **initialize():**
  Beim ersten Aufruf der index.html in Chrome wird diese Funktion aufgerufen.
  Per for-Schleife werden die Midi-Werte der Sounds an die Funktion loadSounds() weitergegeben. Midi wird eingerichtet und mögliche Fehler abgefangen.
  Die Funktion MIDIMessageEventHandler(event) wird aufgerufen. Sie bekommt ihr Event von Drumstick aus QT übergeben. Die hier relevante Funktion aus Drumstick lautet: „void MIDIOutput::***sendNoteOn***(int chan, int note, int vel)“. Int chan, int note und int vel sind gleichzusetzen mit noteNumber, x- und y-Wert in midi.js.
  Die Funktion noteOn() bekommt diese Werte übergeben und ruft dann die Funktion playSound() auf. Anhand der noteNumber, also dem Midi-Wert für die Form und Farbe wird ein Bild in das

img-Feld geladen und eine Beschreibung gesetzt. Anhand der x- und y-Werte wird dann die Animation des Bildes berechnet und im Browser angezeigt.

- **loadSounds():** Die Sounds werden passend zu ihrem Midi-Wert mit loadSounds(midiWert) per GET-Request in einen Buffer geladen.
- **playSound():** Die Funktion wird in der noteOn-Funktion in initialize() aufgerufen und bekommt dort den Midi-Wert und den y-Positionswert der Form übergeben. Die Funktion setupSound() aus soundEditing.js wird aufgerufen und danach das Abspielen des Sounds gestartet.

SoundEditing.js

Diese Datei enthält die Funktionen für die Filter und Regler, mit denen die Sounds bearbeitet werden können.

- **setupSound():** Die Funktion wird in midi.js von playSound() aufgerufen. Die Audio-Nodes für individuelle Lautstärke, Gesamtlautstärke, Frequenz, Verzerrung und Filter werden erst miteinander und zuletzt mit dem AudioContext verknüpft.
- **Abschnitt Frequency, Volume(Gain), Distortion:** Anhand der ID des Reglers wird festgestellt, welcher Regler verändert wurde. Dessen aktueller Wert wird dann geändert und auf den Sound übertragen. Die Funktion makeDistortionCurve() berechnet die Verzerrungskurve des Sounds.
- **Abschnitt Filter:** Bestimmte Frequenzen des Sounds werden anhand des jeweiligen Filters herausgefiltert.
- **Abschnitt Reverb:** Es wird geprüft, ob „kein Nachhall“ ausgewählt ist. Falls dies der Fall ist, wird die aktuelle ConvolverNode getrennt. Wenn ein Nachhall gewünscht ist, wird dessen Sound per GET-Request geladen und danach eine ConvolverNode erstellt und mit den anderen AudioNodes verknüpft. Wenn bereits eine ConvolverNode verknüpft ist, muss diese erst getrennt werden, da diese nur einmal mit einem Sound verwendet werden kann.