

Labor Media Computing

Audio Visualizer



Studenten: Manuel Wehrli, Nicola Keller, Josiane Manera

Betreuung: Fredrik Gundelsweiler

Datum: 17. Januar 2015

# Inhaltsverzeichnis

[Inhaltsverzeichnis 1](#_Toc409448830)

[Einleitung 2](#_Toc409448831)

[Ausgangslage 2](#_Toc409448832)

[Aufgabenstellung 2](#_Toc409448833)

[Anforderungen 2](#_Toc409448834)

[Vorarbeit 3](#_Toc409448835)

[Recherche Technologien 3](#_Toc409448836)

[Programmablauf 4](#_Toc409448837)

[UI Prototyp 5](#_Toc409448838)

[Projektaufbau 6](#_Toc409448839)

[Technologien 6](#_Toc409448840)

[Maven 6](#_Toc409448841)

[Umsetzung 7](#_Toc409448842)

[Userinterface 8](#_Toc409448843)

[Video De- / Encoder 10](#_Toc409448844)

[Bildbearbeitung 11](#_Toc409448845)

[Audio Analyse 11](#_Toc409448846)

[Projekt Installation 12](#_Toc409448847)

[Voreinstellungen 12](#_Toc409448848)

[JDK Version 12](#_Toc409448849)

[JDK Version 12](#_Toc409448850)

[Maven 12](#_Toc409448851)

[Bedienung 13](#_Toc409448852)

[Fazit 14](#_Toc409448853)

[Quellen 15](#_Toc409448854)

# Einleitung

## Ausgangslage

Im Rahmen des Modules Media Computing soll in einer abschliessenden Arbeit das Gelernte in einer praktischen Arbeit umgesetzt werden. Das Projektteam hat sich dafür entschieden als Projekt einen Audio Visualizer zu implementieren.

## Aufgabenstellung

Die Aufgabenstellung des Dozenten sieht eine Implementierung eines Audio Visualizers vor, der auf den Input der Kamera oder eines Videomateriales reagiert, sowie eine Visualisierung des analysierten Bildes und der Musik auf Basis der Frequenzanalyse ausgibt.

## Anforderungen

Gemäss Aufgabenstellung wurden folgende Anforderungen für die Laborarbeit erhoben:

* Entwicklung eines Audio Visualizers, der auf Basis des Audiomateriales, das Bildmaterial des Videos anpasst und ein neues Video erzeugt.
* Erstellung eines Userinterfaces auf dem das Video abgespielt und konvertiert werden kann.
* Analyse des Audiomateriales
* Erstellen eines Mappings wie sich das Audio auf die Bildbearbeitung auswirken soll.
* Aufsetzen einer Entwicklungsumgebung unter Versionskontrolle.
* Recherchearbeit um passende Technologien zu finden.

# Vorarbeit

## Recherche Technologien

Das Team entschied sich anfänglich dafür mit den Technologien Java (als Basis und für das Userinterface), OpenCV (für die Bildbearbeitung) und Minim (für die Audioanalyse) zu arbeiten.

Im Verlauf des Projektes stellte sich heraus, dass OpenCV zu schwerfällig für die Anwendung ist. Die Bearbeitung der Bilder mit OpenCV erwies sich, im Verhältnis zu dem was gemacht werden sollte, als sehr aufwändig. Daraufhin wurde eine simplere Technik verwendet um die Bildsequenzen zu manipulieren.

Relativ bald nach den Recherchen der Technologien wurde klar, dass keine der ausgewählten Technologien, Videodateien decoden und wieder encoden kann, so dass es uns möglich gewesen wäre das Audiomaterial mit Minim zu untersuchen oder an die Frames des Videos zu gelangen. Nach weiteren Recherchen wurde dann das *Xuggler* Framework eingesetzt, mit dem sowohl Audio wie auch Bilder extrahiert werden konnten.

## Programmablauf

Der Programmablauf wurde zu Beginn des Projektes grob festgelegt. Darin ist vorgesehen, dass der Benutzer über die Oberfläche seine Videodatei laden kann und die Musik wird daraufhin bereits analysiert. Auch bei benutzerdefinierten Einstellungen, wie der Film umgewandelt werden soll, findet diese Analyse statt. Weiter kann das Userinterface den Film abspielen.

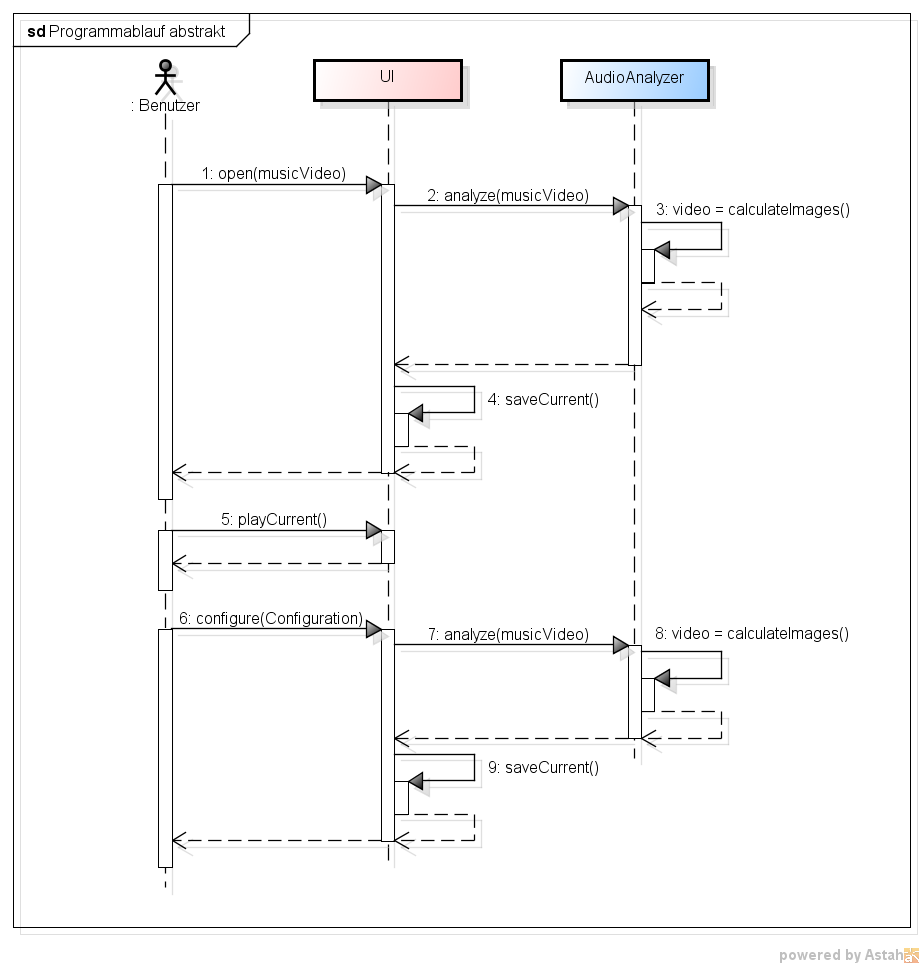


Abbildung 1: Erster grober Entwurf des Programmablaufes. Der AudioAnalyzer ist dabei ein Block aus verschiedenen Klassen die sich sowohl um die Audio Analyse wie auch das zusammenstellen des Filmes beinhaltet.

## UI Prototyp

Auf dem Userinterface ist die originale und die manipulierte Version des Videos zu sehen. Für das Abspielen oder anhalten beider Videos steht eine Toolbar zur Verfügung welche sich auf beide Videos auswirkt.



Abbildung 2: Papierprototyp des Userinterfaces.

# Projektaufbau

## Technologien

Das Projekt basiert auf folgenden Technologien:

* JDK 1.8
* Minim 2.2.0
* Xuggler 5.4

Als Versionskontrolle wird Github verwendet. Als präferierte Entwicklungsumgebung wird IntelliJ IDEA verwendet.

## Maven

Das Projekt ist als Mavenprojekt aufgesetzt. Leider war es bis zum Zeitpunkt der Abgabe nicht möglich auch das Framework *Minim* über Maven zu integrieren (vgl. Moten, 2015).

# Umsetzung

## Userinterface

Für das Userinterface wurde Java FX eingesetzt, das standardmässig in der JDK Version 1.8 integriert ist. JavaFX bot dem Team die Möglichkeit einfach Multimediaformate in das Userinterface einzubinden und abzuspielen. Unterstütze Formate sind dabei:

**Audio**

* MP3
* AIFF mit unkomprimierter PCM
* WAV mit unkomprimierter PCM
* MPEG-4 Multimedia-Container-Format mit AAC (Advanced Audio Coding) Audio

**Video**

* FLV mit VP6 Video und MP3 Audio
* MPEG-4 Multimedia-Container-Format mit H.264/AVC (Advanced Video Coding) Videoverarbeitung.



Abbildung 3: Fertiges Userinterface mit Play-, Pause- und Stop-Button.

## Video De- / Encoder

Beim Decodieren und Encodieren von Videodateien musste das Team als erstes nochmals eine Recherche betreiben um ein fähiges Framework zu finden, das unseren Anforderungen entspricht.

Als erstes Framework für das Decoden von Videos wurde *Humble Video* verwendet. Obwohl es momentan weiterentwickelt wird, steckte es noch in den Kinderschuhen. Es war sehr aufwändig Video und Audio zu extrahieren ohne gleich einiges über Implementierung dahinter wissen zu müssen. Einiges besser war API vom *Xuggler* Framework. Es gab weiter auch gute Dokumentationen, die beschrieben, wie man Bilder aus einem Videostream ziehen kann und ein Video aus Bilder erstellen kann (vgl. Moten, 2015). Mit der Schnittstelle MediaListener kann man sich, während des Lesevorganges des Videos, informieren lassen welches Bild und welcher Audiostream gerade gelesen wurden (siehe Listing 1). Anhand dieser Informationen kann die Audioanalyse und die Bildbearbeitung über die Schnittstelle IVideoProcessor angesprochen werden.

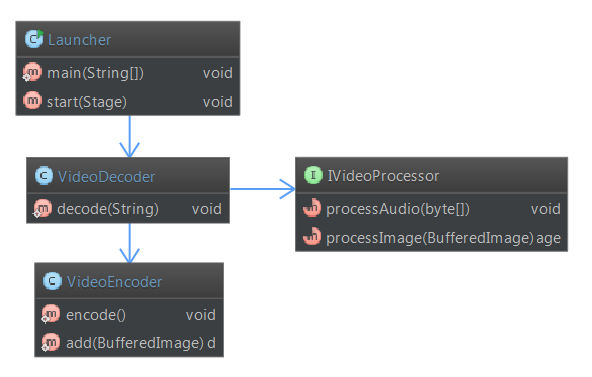


Abbildung 4: Die Applikation startet mit der Benutzeroberfläche. Wenn die Videodatei verarbeitet wird, kommt der VideoDecoder zum Zug und speichert Audio in ein Byte Array und die Frames in einzelne BufferedImages ab. Die bearbeiteten Bilder werden an den VideoEncoder übergeben der eine MP4 Datei mit Videocodec H.264 erstellt.

@Override  
public void onAudioSamples(IAudioSamplesEvent event) {  
 final IAudioSamples audioSamples = event.getAudioSamples();  
  
 final byte[] byteBuffer = audioSamples.getData().getByteArray(0, audioSamples.getSize());  
  
 processor.processAudio(byteBuffer);  
}

Listing 1: Eine der MediaListener Methoden die während des Lesevorganges von Videodateien von Xuggler aufgerufen wird. In diesem Fall kann das erhaltene Audiosample direkt als Byte Array einer Instanz von IVideoProcessor übergeben werden.

Xuggler stellt beim Schreiben einer Videodatei verschiedene Videokompressionen zur Verfügung. Für die Anwendung wurde die Videokompression H.264 verwendet, da JavaFX nur dieses Format abspielen.

## Bildbearbeitung

## Audio Analyse

# Projekt Installation

Das Projekt kann unter der <https://github.com/Violella/AudioVisualizer> bezogen werden.

## Voreinstellungen

### JDK Version

Da Xuggler kann in dem Umfang in dem es im Projekt verwendet wird nur mit einem 32bit JDK laufen. Auf 64bit creasht die VM da es Probleme mit dem darunterliegenden Framework FFmpeg gibt, das auf native DLLs zugreift. Als JDK wird die 32bit Version 1.8.0\_25 empfohlen.

### JDK Version

Durch die Verwenden eines 32bit JDKs, steht weniger Memory zur Verfügung. Dieses Problem kann gelöst werden mit den folgenden VM-Optionen:

-Xmx1g -XX:MaxHeapSize=1g

### Maven

Über Maven warden die für das Programm notwendigen Xuggler Libraries integriert. Vor dem Start des Projektes müssen diese über Maven bezogen werden und verlinkt werden.

# Bedienung

# Fazit

# Quellen

* **Moten David (2015)** *Maven or Gradle for build?* Auf GitHub am 11. Januar 2015. <https://github.com/ddf/Minim/issues/31>, abgerufen am 17. Januar 2015
* **Tsagklis Ilias (2011)** *Xuggler Tutorial: Frames Capture and Video Creation*. Auf Java Code Geeks am 23. Februar 2011. <http://www.javacodegeeks.com/2011/02/xuggler-tutorial-frames-capture-video.html>, abgerufen am 17. Januar 2015
* **Castillo Cindy (2013)** *Incorporating Media Assets Into JavaFX Applications*. Auf docs.oracle.com am April 2013. <http://docs.oracle.com/javafx/2/media/overview.htm>, abgerufen am 19. Januar 2015