

Labor Media Computing

Audio Visualizer



Studenten: Manuel Wehrli, Nicola Keller, Josiane Manera

Betreuung: Fredrik Gundelsweiler

Datum: 17. Januar 2015

# Inhaltsverzeichnis

[Inhaltsverzeichnis 1](#_Toc409281031)

[Einleitung 2](#_Toc409281032)

[Ausgangslage 2](#_Toc409281033)

[Aufgabenstellung 2](#_Toc409281034)

[Anforderungen 2](#_Toc409281035)

[Vorarbeit 3](#_Toc409281036)

[Recherche Technologien 3](#_Toc409281037)

[Programmablauf 4](#_Toc409281038)

[UI Prototyp 5](#_Toc409281039)

[Projektaufbau 6](#_Toc409281040)

[Technologien 6](#_Toc409281041)

[Maven 6](#_Toc409281042)

[Umsetzung 7](#_Toc409281043)

[Userinterface 7](#_Toc409281044)

[Video De- / Encoder 7](#_Toc409281045)

[Bildbearbeitung 7](#_Toc409281046)

[Audio Analyse 7](#_Toc409281047)

[Projekt Installation 8](#_Toc409281048)

[Voreinstellungen 8](#_Toc409281049)

[JDK Version 8](#_Toc409281050)

[JDK Version 8](#_Toc409281051)

[Maven 8](#_Toc409281052)

[Bedienung 9](#_Toc409281053)

[Quellen 10](#_Toc409281054)

# 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

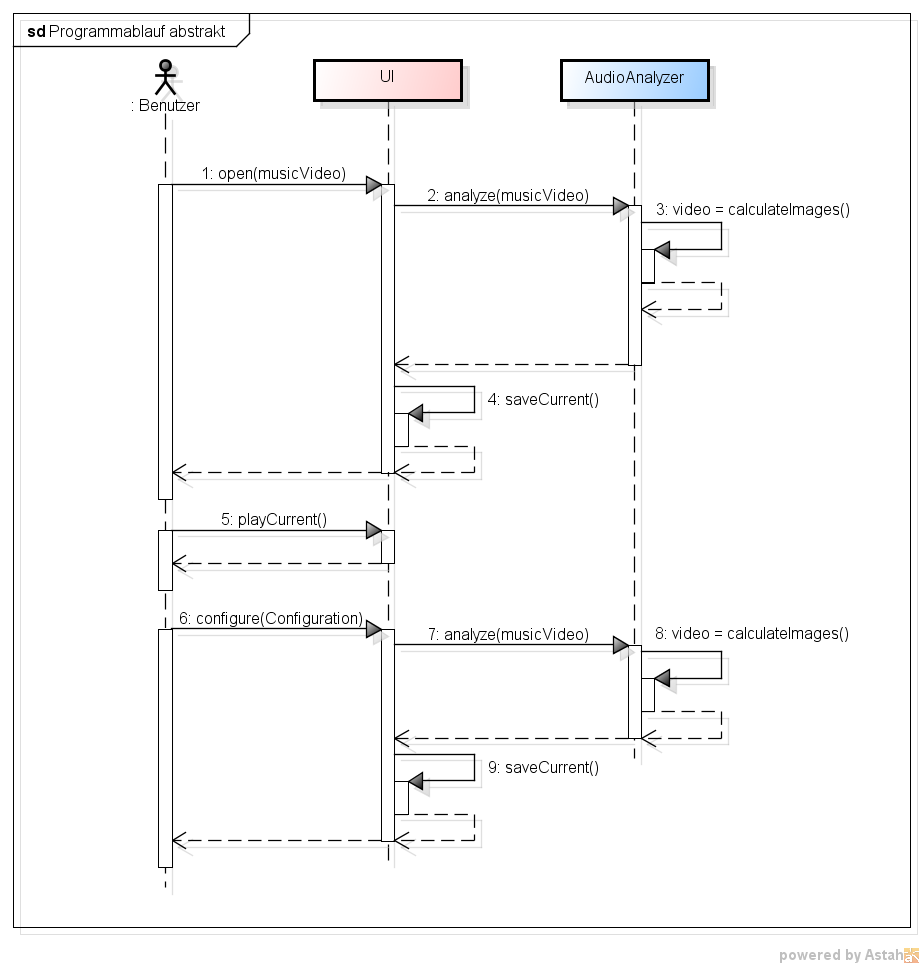
## Recherche Technologien

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



## UI Prototyp



# Projektaufbau

## Technologien

Das Projekt basiert auf folgenden Technologien:

* JDK 1.8
* Minim 2.2.0
* Xuggler 5.4

Als Versionskontrolle wurde Github verwendet. Als präferierte Entwicklungsumgebung wurde IntelliJ 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. Das Team entschied sich dafür, da es mit Java FX sehr einfach möglich ist Medienformat einzubinden.

## Video De- / Encoder

Beim Decodieren und Encodieren von Videodateien mussten wir als Erstes noch 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 z.B Bilder aus einem Videostream ziehen kann und ein Video aus Bilder erstellen kann (vgl. Moten, 2015). Das

## 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

# 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