Medientechnik
Sommersemester 2016

Übung 05 (JavaFX Video)

Terminübersicht

Nr	Zeitraum	Thema
0	18.04. - 21.04.	Organisatorisches, Bildbearbeitung
1	09.05. - 12.05.	JavaFX Einführung (GUIs, Szenengraph)
2	17.05. - 19.05.	Design Patterns: MVC, Observer
3	23.05. - 25.05.	Bildfilter programmieren
4	30.05. - 02.06.	Videobearbeitung
5	**06.06. - 09.06.**	**Video Steuerung + Effekte mit JavaFX**
6	20.06. - 23.06.	Audiobearbeitung
7	27.06. - 30.06.	Audio mit JavaFX

Agenda

- JavaFX Effects & FXML Wiederholung
  - Filter auf Bild anwenden (Snippet)
  - FXML Instanziierung
- Media Package
  - Hierarchie / Theorie
  - MediaView
  - MediaPlayer
- Hands-On
  - Video Player GUI
    - FXML Layout
    - VideoController Klasse
    - Effekte.

Wie•der•ho•lung

RECAP

JavaFX: Effekte auf Bilder anwenden

```
public class EffectExample extends Application {
    @Override
    public void start(Stage primaryStage) throws Exception {
        FlowPane root = new FlowPane();

        ImageView imageView = new ImageView("beispielBild.png");
        GaussianBlur blur = new GaussianBlur();
        imageView.setEffect(blur);

        root.getChildren().add(imageView);
        Scene scene = new Scene(root);
        primaryStage.setScene(scene);
        primaryStage.show();
    }
}
```

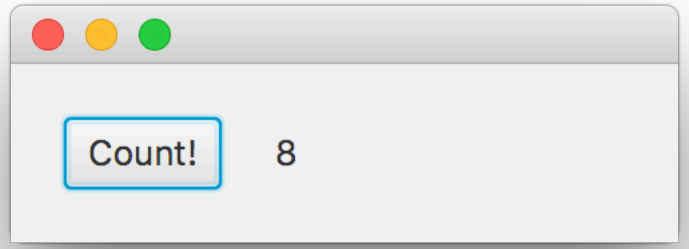
examples.effects.EffectExample.java

FXML Warm-Up Quiz

1. Was ist ein „Namespace“ bei XML?
2. Welchen Teil der MVC Architektur realisiert eine .fxml Datei?
3. Wofür wird `fx:id` benötigt?
4. Was macht die @FXML Annotation?
5. Wie legt man fest, welche Methode eines Programms sich um Actions kümmern soll (innerhalb von `onAction`)

FXML Beispiel: Layout

```
<FlowPane hgap="20.0"
          vgap="20.0"
          prefHeight="50.0"
          prefWidth="250.0"
          xmlns="http://javafx.com/javafx/8.0.65"
          xmlns:fx="http://javafx.com/fxml/1"
          fx:controller="examples.fxml.Controller">
    <children>
        <Button mnemonicParsing="false"
                onAction="#handleButton"
                text="Count!"/>
        <Label fx:id="output"/>
    </children>
    <padding>
        <Insets bottom="20.0"
                left="20.0"
                right="20.0"
                top="20.0"/>
    </padding>
</FlowPane>
```



examples.fxml.layout.fxml

FXML Controller

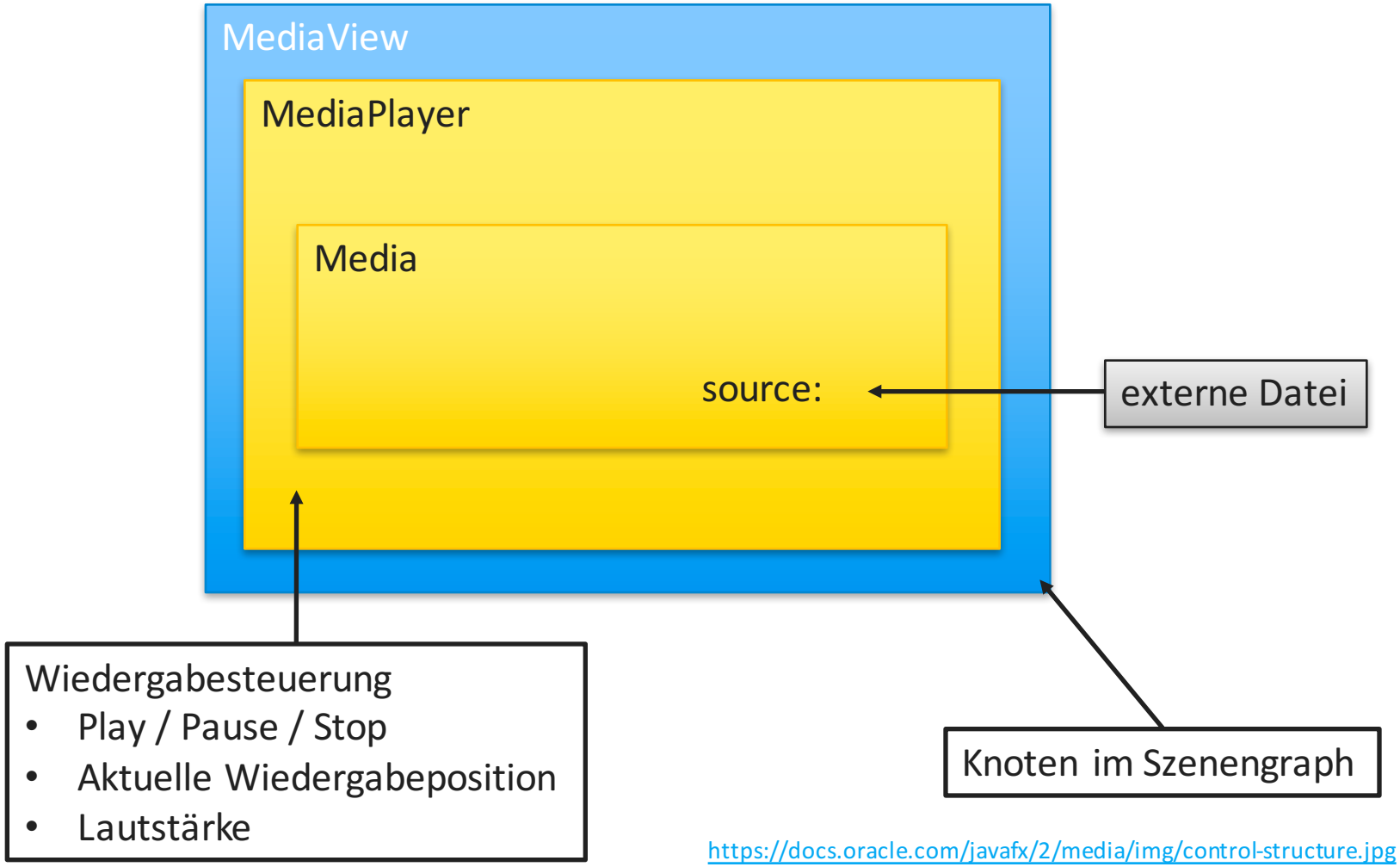
```
public class Controller {

    private int count = 0;

    @FXML private Label output;

    @FXML private void handleButton() {
        this.output.setText(String.valueOf(++count));
    }
}
```

examples.fxml.layout.fxml

Standbilder + Zeitachse + Ton.

BEWEGTBILD MIT JAVAFX

Einbettung von Videos in JavaFX Programme



https://docs.oracle.com/javafx/2/media/img/control-structure.jpg

MediaView Klasse

- Ähnlich zu ImageView
- Erbt von Node
  javafx.scene.Node → javafx.scene.media.MediaView
  Kann also zu Container Elementen hinzugefügt werden
- mediaPlayer Property
  - MediaPlayer Instanz
  - übernimmt die Steuerung des Mediums
  - MediaView bietet nur einen Platz zur Anzeige in der GUI

https://docs.oracle.com/javafx/2/api/javafx/scene/media/MediaView.html

MediaPlayer Klasse

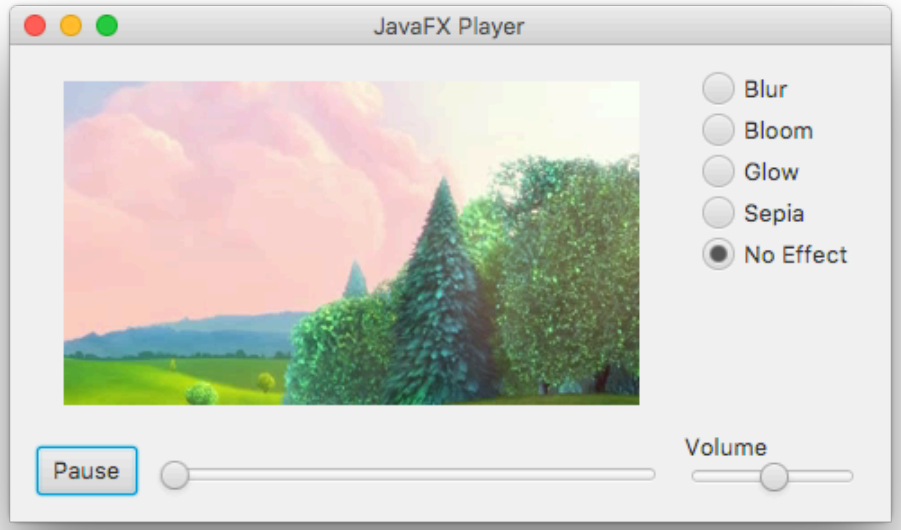
- Übernimmt die **Steuerung** des Mediums
- Properties (Auszug):
  - autoPlay (Boolean): gibt an, ob automatisch begonnen werden soll
  - totalDuration (Duration): Zeitinformationen zum Film
  - currentTime (Duration): aktuelle Wiedergabeposition
  - volume (Double): Lautstärke im Intervall [0;1]
- Wichtige Methoden:
  - play() / pause() / stop()
  - seek(Duration time)

https://docs.oracle.com/javafx/2/api/javafx/scene/media/MediaPlayer.html

Ziel der heutigen Übung

Video-Abspielsoftware mit Effekten

- Play Button
- Slider für Zeit und Laustärke
- Radio Buttons für Effektauswahl



Code Gerüst:
www.medien.ifi.lmu.de/lehre/ss16/mt/uebung/ressourcen/mt_material05.zip

Videoplayer Layout (layout.fxml)

```
<BorderPane xmlns="http://javafx.com/javafx/8.0.65"
            xmlns:fx="http://javafx.com/fxml/1"
            fx:controller="videoplayer.VideoController">
  <padding>
    <Insets bottom="10" left="10" right="10" top="10"/>
  </padding>
  <center>
    <MediaView fx:id="mainMediaView" fitHeight="180" fitWidth="320">
      <mediaPlayer>
        <MediaPlayer fx:id="mainMediaPlayer" autoPlay="true">

          <media>
            <!-- TODO: use a local file -->
              <Media
source="http://download.blender.org/peach/bigbuckbunny_movies/BigBuckB
unny_320x180.mp4"/>
          </media>

        </MediaPlayer>
      </mediaPlayer>
    </MediaView>
  </center>
</BorderPane>
```

Lokale Datei verwenden

- Hinweis:
  - Es ist sinnvoll, die Datei herunterzuladen
  - Weniger Abspielverzögerung und Traffic
  - Bessere Steuerungsmöglichkeiten

- `source` Attribut der letzten Folien ersetzen durch lokale URL: file:///pfad/zur/video/datei.mp4

Videoplayer (Main.java)

```
public class Main extends Application {

  public static void main(String[] args) {
    launch(args);
  }

  @Override
  public void start(Stage primaryStage) throws IOException {
    Parent root = FXMLLoader.load(
                              .getClass()
                              .getResource("layout.fxml"));
    Scene scene = new Scene(root);
    primaryStage.setScene(scene);
    primaryStage.setTitle("JavaFX Player");
    primaryStage.show();
  }
}
```

Breakout: Play Button

- Momentan wird das Video automatisch abgespielt.
- Eure Aufgabe(n):
  - das autoPlay Attribut aus der MediaView entfernen
  - Play Button zum Layout hinzufügen
  - Im Controller eine Methode erstellen, die aufgerufen wird, wenn der Button geklickt wird
  - In dieser Methode das Video starten.
  - Sobald der Button geklickt wurde, wird “Pause” statt “Play” angezeigt
  - (falls noch Zeit ist: die Möglichkeit zum Pausieren des Videos einfügen)
- Zeit: 15 Minuten.
- Code Gerüst: www.medien.ifi.lmu.de/lehre/ss16/mt/uebung/ressourcen/mt_material05.zip

Endlich Special Effects.

VIDEO EFFEKTE

ImageView vs MediaView

- Beide Views:
  Effektverhalten inklusive Methoden ist **identisch**!
- imageView.setEffect(Effect effect)

http://i2.kym-cdn.com/entries/icons/original/000/012/982/post-19715-Brent-Rambo-gif-thumbs-up-imgu-L3yP.gif

Effektoptionen (Wiederholung)

Sepia

Bloom

Glow

Blur

RadioButtons im Layout

```
<right>
  <VBox prefHeight="200.0" prefWidth="100.0" spacing="5.0”
        BorderPane.alignment="CENTER">
    <children>
      <RadioButton fx:id="blurRadio"  onAction="#handleRadioButton" text="Blur">
        <toggleGroup>
          <ToggleGroup fx:id="effectToggles" />
        </toggleGroup>
      </RadioButton>
      <RadioButton fx:id="bloomRadio"  onAction="#handleRadioButton"
                   text="Bloom" toggleGroup="$effectToggles" />
      <RadioButton fx:id="glowRadio"  onAction="#handleRadioButton"
                   text="Glow" toggleGroup="$effectToggles" />
      <RadioButton fx:id="sepiaRadio" onAction="#handleRadioButton"
                   text="Sepia" toggleGroup="$effectToggles" />
      <RadioButton fx:id="noEffectRadio" onAction="#handleRadioButton"
                   selected="true" text="No Effect"
                   toggleGroup="$effectToggles" />
    </children>
    [... weitere Layout Properties (z.B. Padding) ...]
  </VBox>
</right>
```

videoplayer.layout.fxml

Neue Instanzvariablen

```
@FXML     private RadioButton sepiaRadio;
@FXML     private RadioButton blurRadio;
@FXML     private RadioButton bloomRadio;
@FXML     private RadioButton glowRadio;
@FXML     private RadioButton noEffectRadio;


private SepiaTone sepiaTone = new SepiaTone();
private GaussianBlur gaussianBlur =
                              new GaussianBlur();
private Glow glowEffect = new Glow();
private Bloom bloomEffect = new Bloom();
private Effect currentEffect;
```

videoplayer.Controller.java

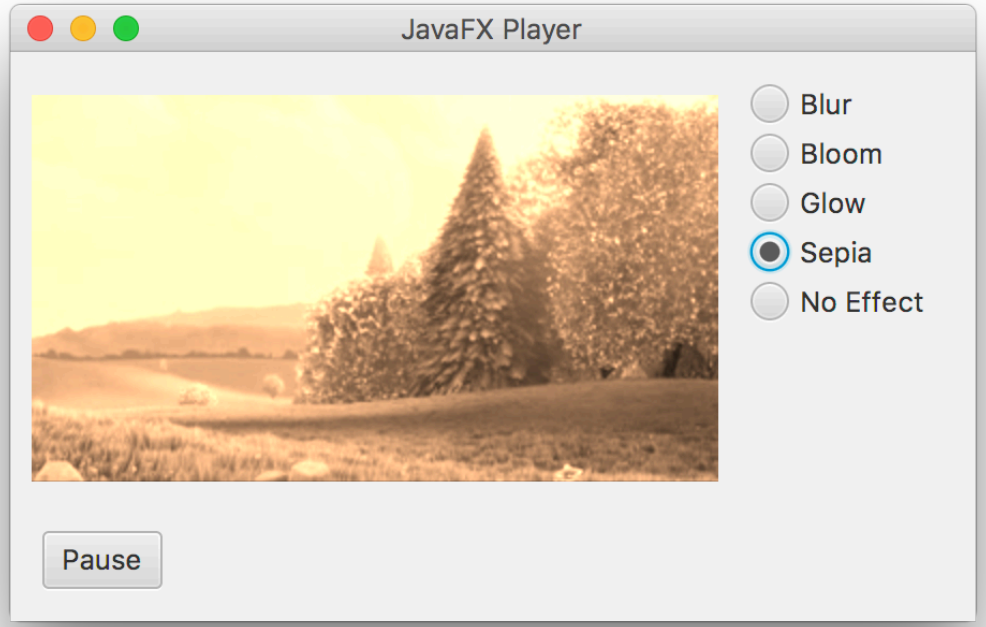
Radio Button Action Handler

```
@FXML
protected void handleRadioButton() {

  if (sepiaRadio.isSelected()) {
    currentEffect = sepiaTone;

  } else if (blurRadio.isSelected()) {
    currentEffect = gaussianBlur;

  } else if (bloomRadio.isSelected()) {
    currentEffect = bloomEffect;

  } else if (glowRadio.isSelected()) {
    currentEffect = glowEffect;

  } else if (noEffectRadio.isSelected()) {
    currentEffect = null;
  }
  // actually apply the effect.
  mainMediaView.setEffect(currentEffect);
}
```

videoplayer.Controller.java

Zwischenergebnis

Bei kleinen Videos tritt keine Verzögerung bei Anwendung der Effekte auf. Für Mutige: Full HD Video einbinden...



Noch mehr

STEUERUNGSELEMENTE

Layout Modifikationen in <bottom>

```
<bottom>
  <HBox prefHeight="20.0" prefWidth="200.0" spacing="10.0"
        BorderPane.alignment="CENTER">
    <children>
      <Button fx:id="playButton"
              onAction="#handlePlayButton" text="Play">
        <HBox.margin><Insets top="8.0"/></HBox.margin>
      </Button>
      <Slider fx:id="playTimeSlider" HBox.hgrow="ALWAYS">
        <HBox.margin><Insets top="16.0"/></HBox.margin>
      </Slider>
      <VBox prefHeight="200.0" prefWidth="100.0">
        <children>
          <Label text="Volume"/>
          <Slider fx:id="volumeSlider"
                  max="1.0" min="0"
                  prefHeight="16.0" prefWidth="60.0"
                  value="0.5"/>
        </children>
      </VBox>
    </children>
  </HBox>
</bottom>
```

videoplayer.layout.fxml

Breakout: Lautstärke Slider

- Der Lautstärke Slider soll auf Nutzereingaben reagieren.
- Vorschlag: Data binding!
- Schritte:
  - Instanzvariable ”volumeSlider” im Controller
  - in der initialize Methode das Databinding an die volume Property des mediaPlayer Objekt einrichten.
  - Lautstärke standardmäßig auf 0.5 setzen.
- Zeitrahmen: 10 Minuten.

Zeit-Slider: Vorkehrungen

- Neue Instanzvariablen:

```
@FXML     private Slider playTimeSlider;
private Duration duration;
private final int timestep = 100;
```

- Duration: Informationen über Länge
- timestep:
  - Hintergrund: Wir wollen die Position des Zeit-Slider nicht jede Millisekunde aktualiseren (Overhead)
  - Lösung: wir aktualisieren den Slider nur alle 100 Millisekunden (= 1/10 Sekunde)

Zeit-Slider: Maximum setzen

- Damit der Slider korrekt funktioniert, müssen wir seinen maximalen Wert festlegen
- Bevor wir mit dem Zeitslider interagieren können, müssen wir warten bis der MediaPlayer den “ready” Status erreicht hat.
  ➔ Wann steht das Video bereit / Wann ist es gepuffert?
- Code

```
this.mainMediaPlayer.setOnReady(() -> {
  this.duration = this.mainMediaPlayer.getTotalDuration();
  this.playTimeSlider.setMax(
                this.duration.toMillis() / this.timestep);
});
```

Zeit-Slider: Wert aktualisieren

- **Ziel**: Immer wenn sich die currentTime Property des MediaPlayer Objekts ändert, soll der Zeit-Slider anpassen

- **Problem**: Databinding funktioniert nicht “out-of-the-box” mit currentTime ☹
  (ObjectProperty kann nicht direkt an DoubleProperty gebunden werden)

- **Lösung**: .currentTimeProperty().addListener(...)

Zeit-Slider: Event Listener

```
this.mainMediaPlayer.currentTimeProperty().addListener((ov) -> {

  Duration currentTime = mainMediaPlayer.getCurrentTime();
  // don't do anything if there is already a change going on.
  if (!playTimeSlider.isValueChanging()) {
    // timestep defines when to move the slider forward.
    // --> every 100 milliseconds (1/10th of a second).
    playTimeSlider.setValue(
        currentTime.toMillis() / timestep);
  }
});
```

Seek / Scrub

- **Ziel**: Wenn man den Slider zieht, soll die Wiedergabeposition automatisch angepasst werden.
- **Wichtige Property**: valueChanging (Boolean)
  zeigt an, ob gerade am Wert etwas geändert wird = wenn Nutzer interagiert

```
this.playTimeSlider.valueProperty()
    .addListener((ov) -> {
      if (this.playTimeSlider.isValueChanging()) {
        this.mainMediaPlayer.seek(
            new Duration(
                this.playTimeSlider.getValue() *
                this.timestep));
      }
    });
```

Wrap-Up Quiz

1. Welches der 3 wird als Knoten in die GUI gehängt:
   `MediaView` `MediaPlayer` `Media`

2. Wie wird die Wiedergabe eines geladenen Videos gestartet?

3. Wie fügt man einen Weichzeichnereffekt dem Video hinzu?

4. Warum haben wir bei der Berechnung der Position des Zeit-Sliders die “timestep” Variable verwendet?

5. Warum haben wir beim “Scrubbing” die valueChanging Property geprüft?

Ankündigung

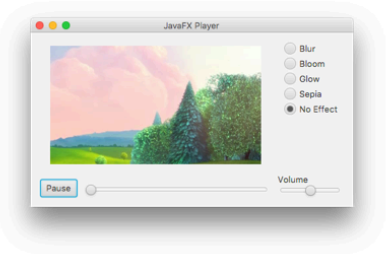
- Gastvortrag von Volker Gabriel in der Medientechnik Vorlesung
- **Wann?** 10.06. 2016 10:00
- **Wo?** Schellingstr. 3, S006
- **Was?** Einsichten in die Arbeit eines Kameramanns + Hands-On Workshop

Übungsblatt 5

Übungsblatt 5 – JavaFX Video

Aufgabe 1: Media Player aus der Übung

In Übung 5 wurde vor Ort die Basisfunktionalität eines JavaFX Programms implementiert, mit der sich ein Video grundlegend abspielen und steuern lässt. Außerdem werden auf dem Video Effekte angewandt. Sollten Sie diese Übung verpasst haben, ist es für die folgenden Aufgabenteile notwendig, die Basisfunktionalität des JavaFX noch zu implementieren. Das Ausgangsmaterial mit einer funktionierenden GUI finden Sie bei den Ressourcen auf der Webseite. Für die weitere Implementierung nehmen Sie bitte die Code-Auszüge auf den Folien zu Hilfe.



Für diese Aufgabe ist keine separate Abgabe nötig, weil sie bereits Teil der Übung war.

Aufgabe 2: Erweiterung des MediaPlayers

Der MediaPlayer aus der Übung (bzw. Aufgabe 1) soll nun erweitert werden. Fügen Sie diese Funktionalitäten hinzu:

A) **Mehrere Effekte gleichzeitig aktivierbar.**
Hierfür sollten die RadioButtons durch Checkboxen ersetzt werden. Die Programmlogik lässt sich in wenigen Zeilen ergänzen (siehe Folie 22 von Übung 5).
Tipps: Die Klasse Effect besitzt die Methode b.setInput(Effect a), mit der sich ein neuer Effekt b auf einen bereits aktivierten Effekt a, anwenden lässt.

LMU München – Übungen zur Vorlesung Medientechnik – Sommersemester 2016

B) **Dateiauswahl.**
Erlauben Sie den Nutzern die Auswahl einer eigenen Videodatei. Hierbei ist darauf zu achten, dass nicht alle Videotypen unterstützt werden. Die Auswahl der Datei soll über einen FileChooser Dialog (vgl. Übung 03) erstellt werden.
   a. Erstellen Sie eine sinnvolle Möglichkeit für die Nutzer des Programms, den Dateiöffnungsdialog zu starten. Ein Menü würde sich z.B. anbieten.
   b. Bringen Sie die unterstützten Formate in Erfahrung
   c. Achten Sie auf eine geeignete Ausnahmenbehandlung.

C) **Video-URLs.**
Die Nutzer des Programms sollten das abzuspielende Video nicht nur über die Dateiauswahl ändern können, sondern auch durch Eingabe einer URL zu einem Video, das daraufhin geladen wird.

Hinweis: Es ist nicht gedacht, z.B. YouTube URLs anzugeben, sondern Pfade zu Videodateien auf Servern, z.B.
http://download.blender.org/peach/bigbuckbunny_movies/BigBuckBunny_320x180.mp4

D) **Dauer des Videos als Label.**
Die GUI soll an geeigneter Stelle (z.B. rechts neben dem Timeslider) die gesamte Dauer des Videos in der Form Minuten:Sekunden (MM:SS) anzeigen.

E) **Aktuelle Abspielposition als Label.**
Derzeit ist die Abspielposition nur am aktuellen Wert des „playTimeSlider" zu erkennen. Für den Nutzer wäre es jedoch hilfreich, die aktuelle Position in der Form Minuten:Sekunden (MM:SS) ablesen zu können. Diese Anzeige kann links neben der Dauer stehen, z.B. so:

01:30 / 02:50

F) **(Optional) Fügen Sie weitere Effektoptionen hinzu.**
G) **(Optional) Fügen Sie die Möglichkeit hinzu, Screenshots des Videos zu erstellen.**
H) **(Optional) Fügen Sie weitere Kontrollmöglichkeiten hinzu.**
Zum Beispiel Buttons mit denen im Video 15 Sekunden vor oder zurück gesprungen werden kann.
I) **(Optional) Fügen Sie "Keybindings" hinzu.**
Lassen Sie die Nutzer mit der „J" Taste 15 Sekunden zurück springen.
„Leertaste" oder „K" starten oder pausieren das Video. Mit der „L" Taste springt man 15 Sekunden vorwärts.
Demo: Drücken Sie die J / K / L Taste während Sie im Browser ein YouTube Video ansehen.

Speichern Sie Ihre Antwort in der Datei „aufgabe2.pdf" und fügen Sie sie Ihrer Abgabe hinzu.

Vielen Dank!

WELCHE FRAGEN HABT IHR?

Links

- https://docs.oracle.com/javafx/2/api/javafx/scene/media/package-frame.html