Medientechnik
Sommersemester 2016

Übung 07 (JavaFX Audio)

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

- Wiederholung: Media Player
- Digitale Medien:
  Audio Frequenzbänder
- Code-Along:
  GUI zur Visualisierung des
  Frequenzspektrums einer Audiodatei

Wie•der•ho•lung

RECAP

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

Much analyze. So audio.

AUDIOVERARBEITUNG MIT JAVAFX

Ziel der heutigen Übung

https://youtu.be/qlCygEWCGfI

Frequenzspektrum

- In einem Signal enthaltene Frequenzen und deren Intensität
- Intensität ≈ Lautstärke
- Digitalisierung:
  - **Diskretisierung:** Frequenzen auflösen
    Nyquist Theorem: Abtastrate > 2 * Maximalfrequenz.
  - **Quantisierung**: Dynamikunterschiede auflösen

Frequenzbänder

- Auch “Frequenzgruppen”
- Psychoakustisches Modell:
  - **Maskierungseffekte**: Nicht alle Frequenzen werden gleich wahrgenommen. Grad der Maskierung abhängig von der Signalintensität im kritischen Band einer Frequenz
  - zwischen 24/27 kritische Bänder (je nach Auslegung des Modells)
  - kritische Bänder **unterschiedlich breit**
  - Stichwort: “Bark-Skala”

- In JavaFX: 128 gleich breite Bänder standardmäßig zur Verfügung.

https://de.wikipedia.org/wiki/Frequenzgruppe

Los geht's! Audio Spektrum GUI

- IDE starten
- Material inklusive Codegerüst und MP3-Datei: https://www.medien.ifi.lmu.de/lehre/ss16/mt/uebung/ressourcen/mt_material07.zip

- Dateien:
  - **Controller.java**
  - FXMLMain.java
  - layout.fxml

Überblick über die TODOs in Controller.java

1. Instanzvariablen: MediaPlayer Setup
2. Initialisierung des AudioSpectrumListener
   → `initialize(...)`
3. Zufällige Farbe generieren, wenn in die GUI geklickt wird
   → `changeColor()`
4. Wenn die Leertaste gedrückt wird, soll das Audio je nach aktuellem Status entweder abgespielt oder pausiert werden.
   → `togglePlaying()`
5. Dialog um neue Audio-Datei zu öffnen
   → `handleFileOpen()`

Schritt 1: Instanzvariablen

```
private static String mediaUrl =
    Controller.class.getResource("/mix.mp3").toString();

private static Media audioFile =
    new Media(mediaUrl);

private static MediaPlayer mediaPlayer =
    new MediaPlayer(audioFile);

private boolean isPlaying = false;

private static Paint color = Color.rgb(0,0,0);
```

Schritt 2: AudioSpectrumListener

- Schnittstelle mit genau einer Methode:

```
void spectrumDataUpdate(double timestamp,
                        double duration,
                        float[] magnitudes,
                        float[] phases);
```

- Für uns relevant: `float[] magnitudes`
  - Array mit negativen Dezibel-Werten einzelner Frequenzbänder
  - Standardwerte zwischen -60 dB und 0 dB
  - Standardmäßig 128 gleich große Bänder
- Dokumentation: https://docs.oracle.com/javase/8/javafx/api/javafx/scene/media/AudioSpectrumListener.html

Schritt 2: Lambda-Ausdruck

```
 mediaPlayer.setAudioSpectrumListener(
        (double timestamp,
         double duration,
         float[] magnitudes,
         float[] phases
        ) -> {
          // Sub-TODO:
          // a) Höhe/Breite des Pane's ausrechnen
          // b) Durch alle Bänder iterieren
          // c) Rechteck für jedes Band zeichnen

        });
}
```

Schritt 2: Höhe und Breite der Anzeige

```
double paneHeight = equalizerContainer.getHeight();
double paneWidth = equalizerContainer.getWidth();


int rectangleWidth =
        (int) (paneWidth / magnitudes.length);


// remove all rectangle before we add new ones.
equalizerContainer.getChildren().clear();
```

Breakout:

- Durchschnittlichen Pegel ausrechnen
- Ergebnis auf der Kommandozeile / ausgeben (per System.out...)

```
-60.0
-60.0
-60.0
-59.87892681360245
-58.0079225897789
-57.9500108063221
-58.075687140226364
-58.14387458562851
-57.99502211809158
-58.12375953793526
-58.0863258838653356
-58.39883941411972
-58.32557538151741
-59.27411088347435
-58.901950389146805
```

- Zeit: 5 Minuten.

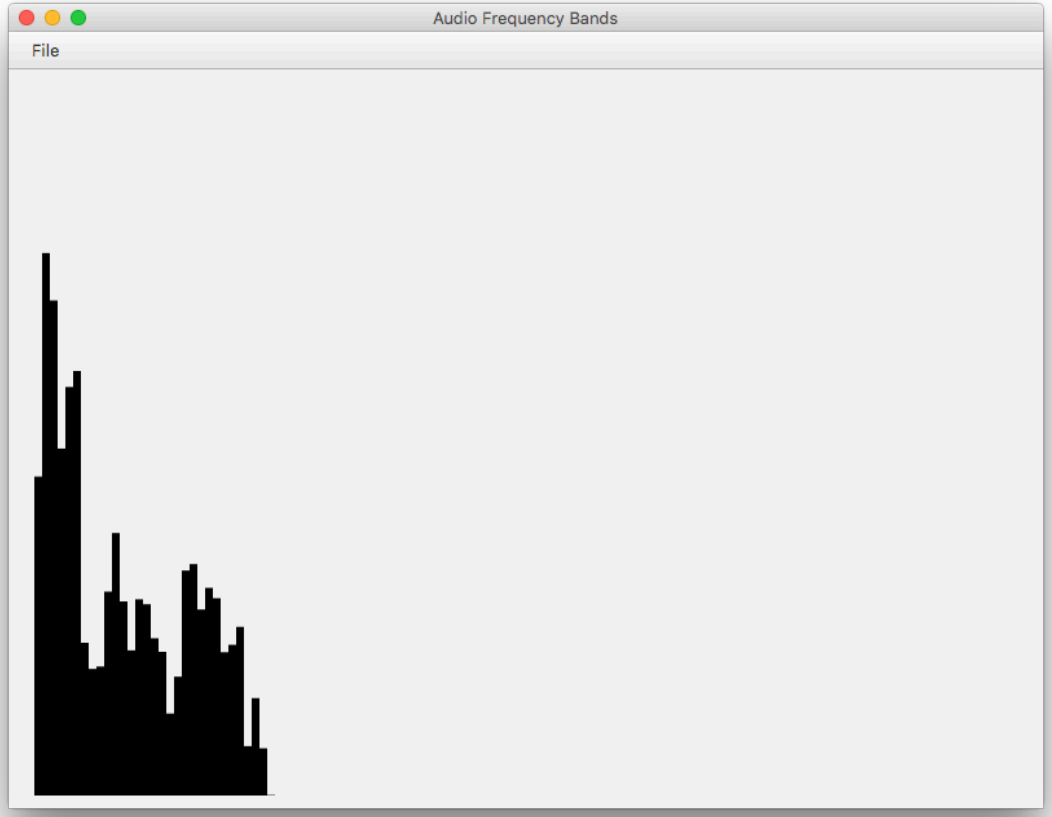
Schritt 2: Höhe des Rechtecks bestimmen.

```
// b) Durch alle Bänder iterieren
for (int i = 0; i < magnitudes.length; i++) {
  // what's the current magnitude?
  float bandLevel = magnitudes[i];
  // we don't start on the left, but with some margin
  int xOffset = 10;
  // by default, -60dB is the lowest value.
  // but we need rectangles with positive height.
  // so we use this number to adjust the size.
  int minMagnitude = 60;

  // let's make the bands a little bigger by default:
  int gain = 2;

  // what's the rectangle's original width?
  double height = (bandLevel + minMagnitude) * gain;
  // probably that's too small (not more than 5-6 px).
  // so we calculate the percentage regarding the peak
  // and relate this percentage to the window height.
  // normalize the height:
  double heightRatio = height / minMagnitude;
  // actually adjust the height:
  height = paneHeight * heightRatio;

  // --> nächste Folie
}
```

Schritt 2: Rechtecke zeichnen

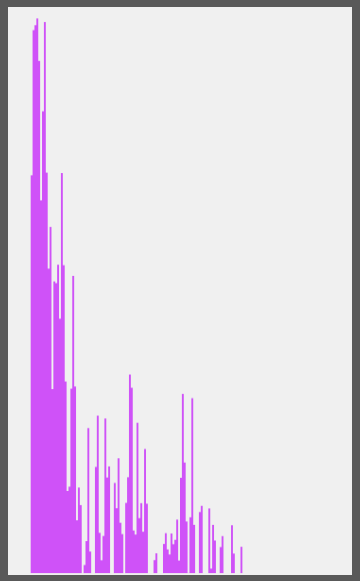
```
//
// c) Rechteck für jedes Band zeichnen
//
Rectangle rectangle = new Rectangle();

rectangle.setX(xOffset + (i * rectangleWidth));
// move it up to make it visible.
rectangle.setY(paneHeight - height);

rectangle.setWidth(rectangleWidth);
rectangle.setHeight(height);

rectangle.setFill(color);
equalizerContainer.getChildren().add(rectangle);
```

Zwischenergebnis



Breakout: Anzahl der Frequenzbänder verändern

- Tipp:
  setAudioSpectrumNumBands(int num)



- Was fällt auf, wenn man die Anzahl erhöht? Was kann man beobachten wenn man die Anzahl verringert?

Überblick über die TODOs in Controller.java

1. Instanzvariablen: MediaPlayer Setup
2. Initialisierung des AudioSpectrumListener
   → `initialize(...)`
3. Zufällige Farbe generieren, wenn in die GUI geklickt wird
   → `changeColor()`
4. Wenn die Leertaste gedrückt wird, soll das Audio je nach aktuellem Status entweder abgespielt oder pausiert werden.
   → `togglePlaying()`
5. Dialog um neue Audio-Datei zu öffnen
   → `handleFileOpen()`

Schritt 3: Zufällige Farbe erzeugen

```
@FXML
protected void changeColor() {
  // STEP 3
  Random random = new Random(System.currentTimeMillis());
  int red = random.nextInt(255);
  int green = random.nextInt(255);
  int blue = random.nextInt(255);
  color = Color.rgb(red, green, blue, .70);
}
```

Überblick über die TODOs in Controller.java

1. Instanzvariablen: MediaPlayer Setup
2. Initialisierung des AudioSpectrumListener
   → `initialize(...)`
3. Zufällige Farbe generieren, wenn in die GUI geklickt wird
   → `changeColor()`
4. Wenn die Leertaste gedrückt wird, soll das Audio je nach aktuellem Status entweder abgespielt oder pausiert werden.
   → `togglePlaying()`
5. Dialog um neue Audio-Datei zu öffnen
   → `handleFileOpen()`

Schritt 4: Play / Pause Callback

```
mediaPlayer.setOnReady(() -> {
  mediaPlayer.play();
  mediaPlayer.setOnPlaying(() -> {
    isPlaying = true;
  });
  mediaPlayer.setOnPaused(() -> {
    isPlaying = false;
  });
});
```

Schritt 4: KeyboardListener

```
scene.setOnKeyReleased(event -> {
  switch (event.getCode()) {
    case SPACE:
      togglePlaying();
      break;
  }
});
```

Schritt 4: togglePlaying()

```
private void togglePlaying() {

  if (isPlaying) {
    mediaPlayer.pause();
  } else {
    mediaPlayer.play();
  }

}
```

Überblick über die TODOs in Controller.java

1. Instanzvariablen: MediaPlayer Setup
2. Initialisierung des AudioSpectrumListener
   → `initialize(...)`
3. Zufällige Farbe generieren, wenn in die GUI geklickt wird
   → `changeColor()`
4. Wenn die Leertaste gedrückt wird, soll das Audio je nach aktuellem Status entweder abgespielt oder pausiert werden.
   → `togglePlaying()`
5. Dialog um neue Audio-Datei zu öffnen
   → `handleFileOpen()`

Schritt 5: Datei öffnen

```
protected void handleFileOpen() {

  mediaPlayer.pause();

  try {
    FileChooser fileChooser = new FileChooser();
    fileChooser.getExtensionFilters()
               .add(new FileChooser.ExtensionFilter("MP3", "*.mp3"));
    fileChooser.getExtensionFilters()
               .add(new FileChooser.ExtensionFilter("WAV", "*.wav"));

    File loadedFile = fileChooser.showOpenDialog(null);

    String imgURL = loadedFile.toURI().toURL().toString();
    audioFile = new Media(imgURL);
    mediaPlayer.pause();
    mediaPlayer = new MediaPlayer(audioFile);

    initialize(null, null);
  } catch (Exception ignored) {
  }
}
```

Wrap-Up Quiz

1. Was ist ein Frequenzband?
2. Was versteht man unter “Diskretisierung”?
3. Was besagt das psychoakustische Modell?
4. Warum ist das Spektrum in unserer Visualisuerungs-GUI so weit links angesiedelt?
5. Wo befindet sich der Ankerpunkt eines Rectangle Objekts in JavaFX?

Vielen Dank!

WELCHE FRAGEN HABT IHR?

Bitte füllt diesen Feedbackbogen
zur Übung aus, damit wir den Übungsbetrieb
weiter verbessern können

http://goo.gl/forms/hl5crlgJKJCtj68F3

THAT'S IT FOR THIS YEAR!