9 Programming with Video

9.1 Playing Video and Playback Control

9.2 Interactive Video

Literature:

James L. Weaver: Pro JavaFX 2: A Definitive Guide to Rich Clients with Java Technology, Apress 2012

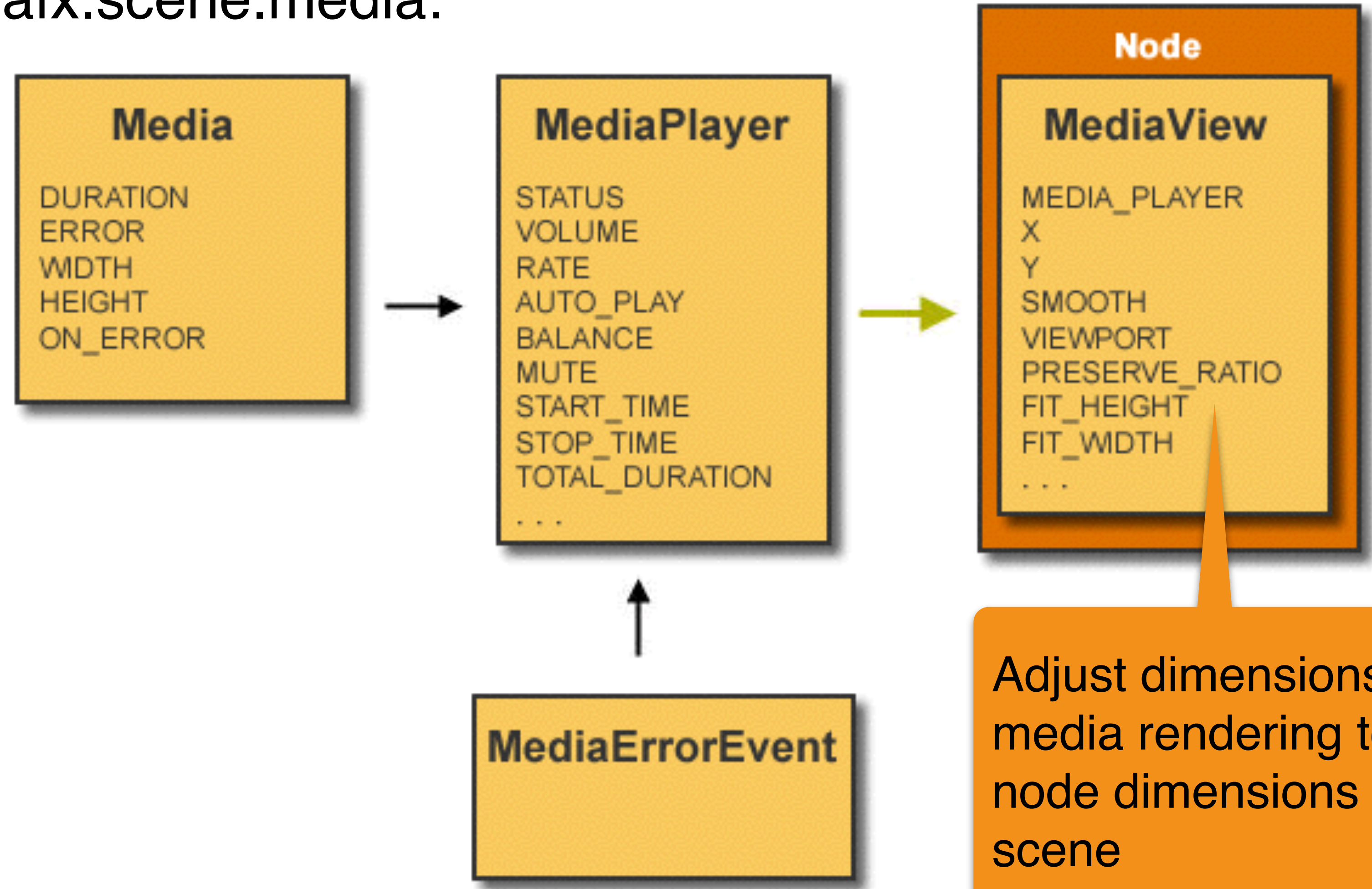
http://docs.oracle.com/javafx/2/media/playercontrol.htm

Video Playback in High-Level Frameworks

- Example Cocos2d-x (high-level game framework)
  - Built-in class VideoPlayer (experimental)
  - Subclass of UINode (therefore of scene graph Node)
  - Typical methods: play, pause, resume, stop
  - Event listeners based on state transitions
  - Support for full screen video
- Example JavaFX (high-level general multimedia framework)
  - Built-in classes MediaPlayer, MediaView
  - MediaView is subclass of scene graph Node
  - Built-in support for designing control UI elements
- Other platforms/frameworks
  - Either similar (e.g. in HTML5-based frameworks)
  - Or need to wrap platform-specific components

Video Playback with JavaFX

javafx.scene.media:



Adjust dimensions of media rendering to view node dimensions within scene

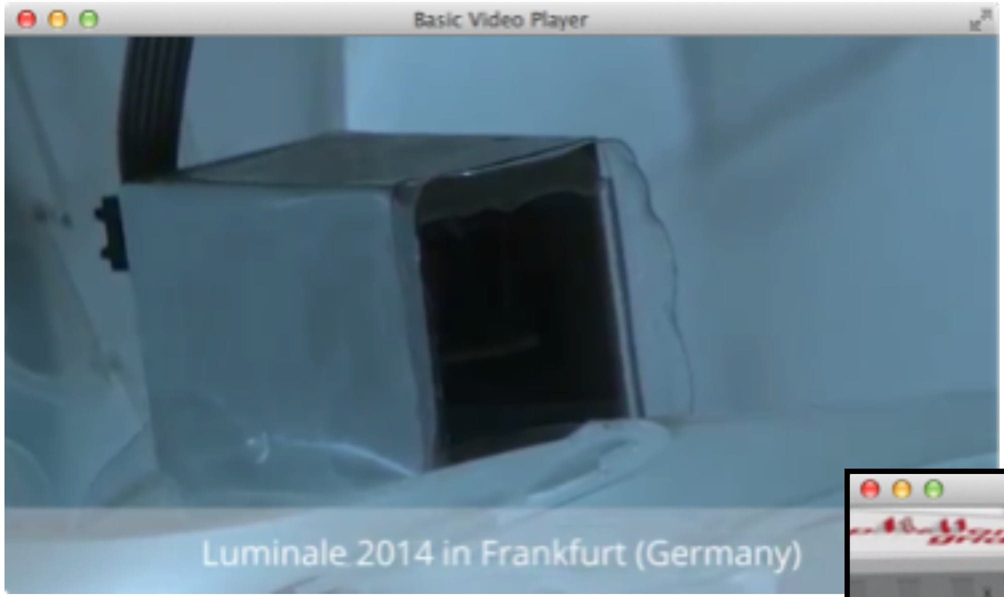
Basic Video Playback Application

```
private static final int SCWIDTH = 640;
private static final int SCHEIGHT = 360;

@Override
public void start(Stage primaryStage) {

  primaryStage.setTitle("Basic Video Player");
  Group root = new Group();
  Scene scene = new Scene(root);

  Media media = new Media(
     getClass().getResource("XXX.mp4").toString());
  MediaPlayer mediaPlayer = new MediaPlayer(media);
  MediaView mediaView = new MediaView(mediaPlayer);
  mediaView.setFitWidth(SCWIDTH);
  mediaView.setFitHeight(SCHEIGHT);

  root.getChildren().add(mediaView);
  primaryStage.setScene(scene);
  primaryStage.show();
  mediaPlayer.play();
}
```

Interactive Selection of Video Source File

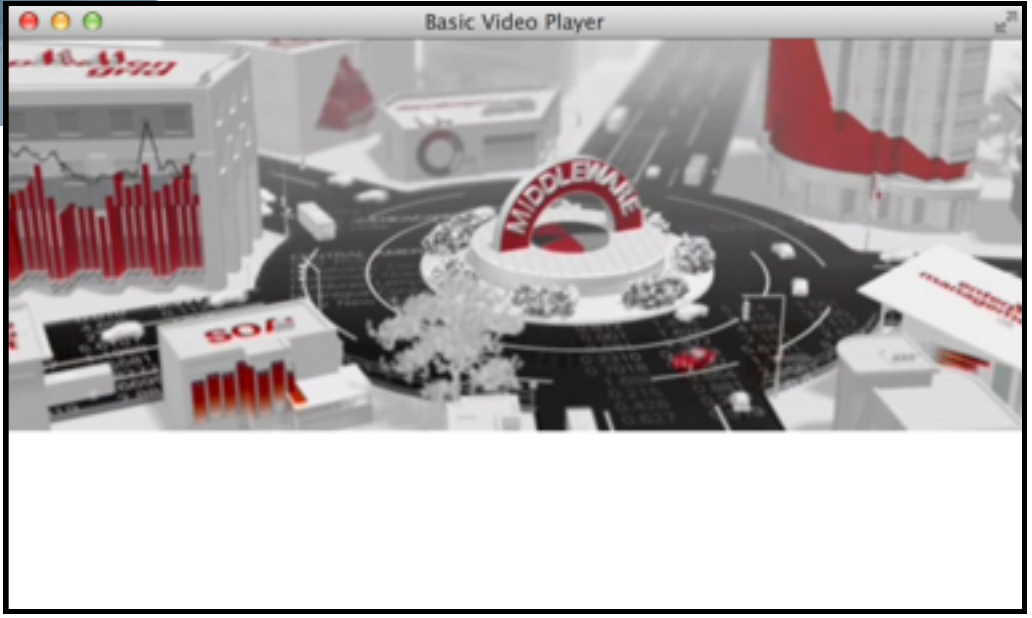
```
FileChooser fileChooser = new FileChooser();
fileChooser.setTitle("Please select video file");
File file = fileChooser.showOpenDialog(primaryStage);
if (file != null) {
  String mediaURI = file.toURI().toString();
  try {
    Media media = new Media(mediaURI);
    MediaPlayer mediaPlayer = new MediaPlayer(media);
    MediaView mediaView = new MediaView(mediaPlayer);
    mediaView.setPreserveRatio(true);
    mediaView.setFitWidth(SCWIDTH);

    root.getChildren().add(mediaView);
    primaryStage.setScene(scene);
    primaryStage.show();
    mediaPlayer.play();
  }
  catch (MediaException e) {
    System.out.println("Media Exception");
    System.exit(0);
  }
 }
```

Problem: Adaptation to Media Aspect Ratio

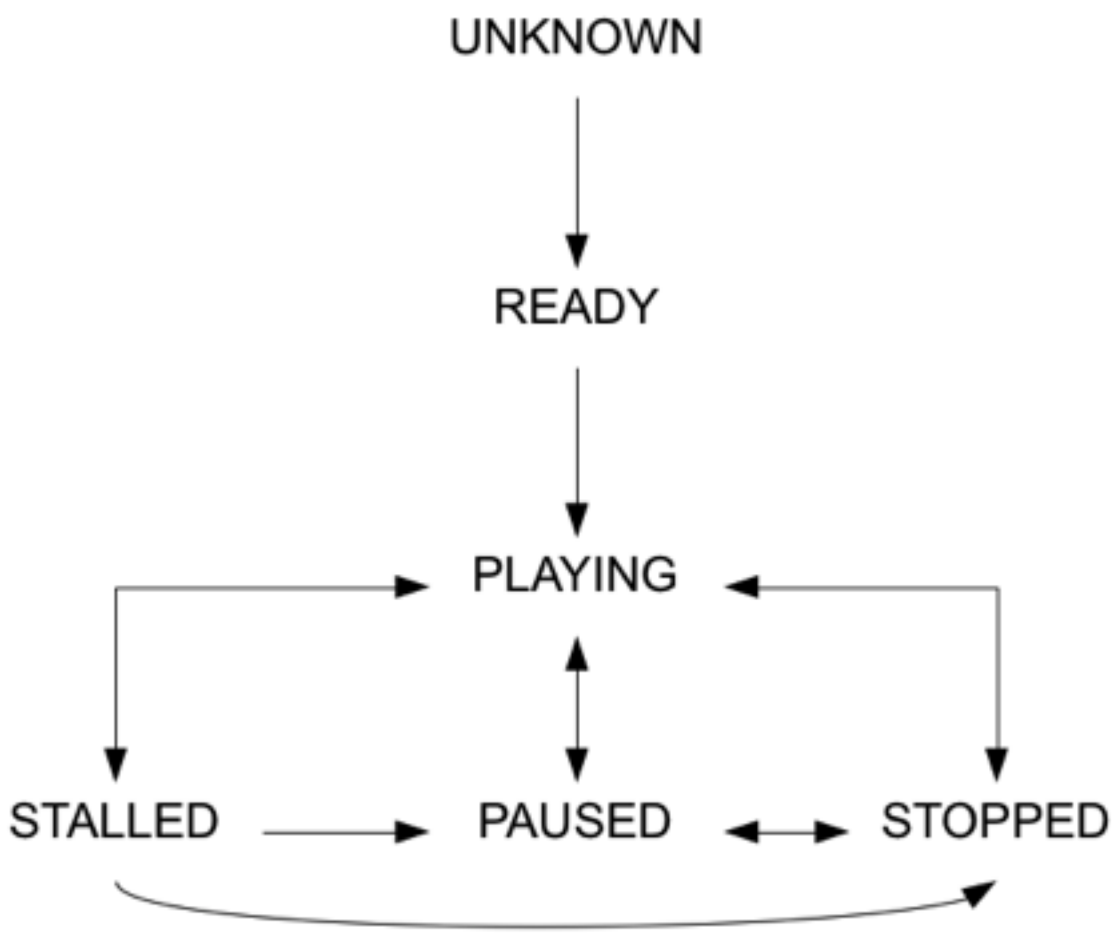


Video aspect ratio equal to scene (640x360 = 16:9)

Difference in aspect ratio (free space)



State Model for Media Playback System



Quote from
http://docs.oracle.com/javafx/2/api/:

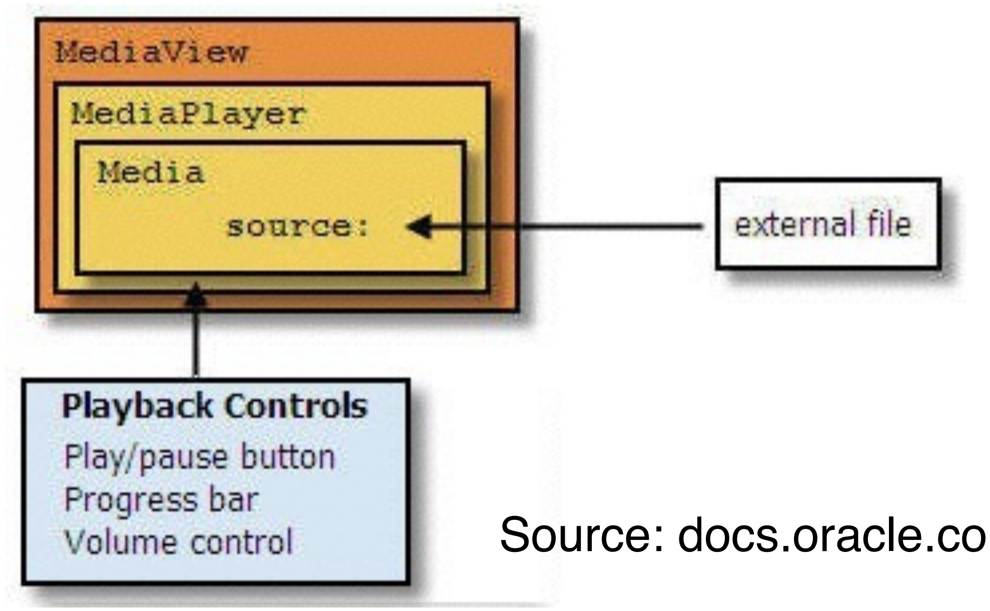
The media information is obtained asynchronously and so not necessarily available immediately after instantiation of the class. All information should however be available if the instance has been associated with a MediaPlayer and that player has transitioned to MediaPlayer.Status.READY status.

QUIZ:
How can we adapt our display to media aspect ratio?

State Transition Listener

```
mediaView.setPreserveRatio(true);
mediaView.setFitWidth(SCWIDTH);

mediaPlayer.setOnReady(new Runnable() {
  public void run() {
    mediaView.setFitHeight(
      mediaPlayer.getMedia().getHeight());
    primaryStage.sizeToScene();
  }
});
```

Controlling Media Playback



Source: docs.oracle.com

- Different properties (player state, media time, volume etc.)
- User-initiated control:
  - Start, pause, seek to position, set volume
- System feedback:
  - Player status, position in media, current volume
- Traditionally, control and feedback integrated into a single interface

9 Programming with Video

9.1 Playing Video and Playback Control

9.2 Interactive Video

9.3 Components for Multimedia Programming

Literature:

James L. Weaver: Pro JavaFX 2: A Definitive Guide to Rich Clients with Java Technology, Apress 2012

Events Generated by Media Components

- Various events are reported by Media Components to the surrounding application for flexible reaction:
  - User interaction like playback control
  - Media events like reaching end of media
  - User-defined events when reaching specific positions *(cue events)*

- Reaction to media events requires *EventListener* objects for media specific events, e.g.:

```
public final void
   setOnHalted(java.lang.Runnable value)
```

Cue Points / Media Markers

- A *cue point* marks a specific point in time during media playback.
  - Specification by *time stamp* relative to media start time
  - Flash/ActionSript: “cue point”
  - JavaFX: “Media marker”
- Internal cue point: Embedded into movie file
  - Supported by some video formats
- External cue point: Defined outside movie file
  - When reaching a cue point, a (script) event is fired

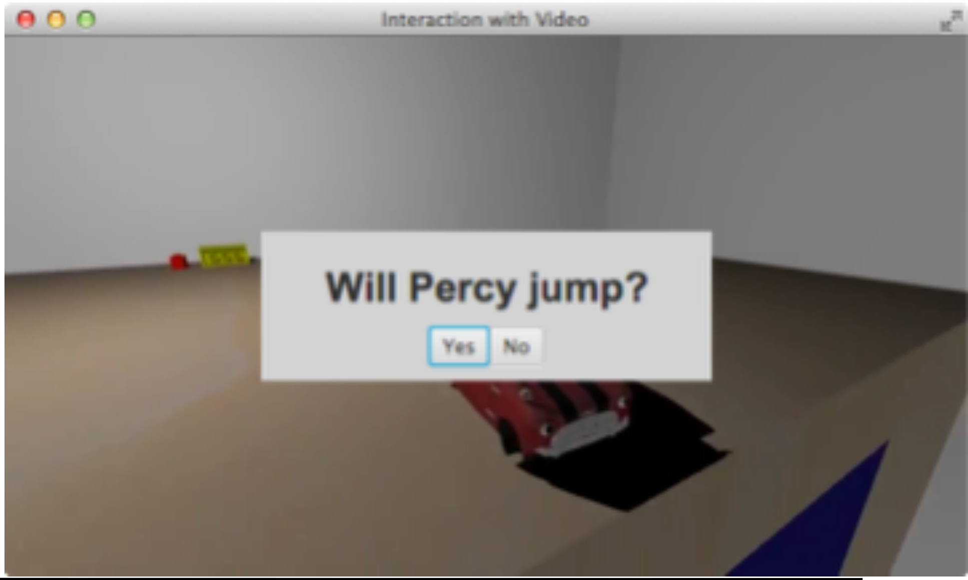
– Cue points can always be simulated by timers

Media Markers and Media Marker Events

```
Media media = new Media(getClass()
     .getResource”PercysPerfectPlan.mp4")
     .toString());
…
final ObservableMap<String, Duration> markers =
     media.getMarkers();
markers.put("onEdge", Duration.millis(33500));
markers.put("noJump", Duration.millis(40000));
markers.put("jump", Duration.millis(103000));

MediaPlayer mediaPlayer = new MediaPlayer(media);
…
mediaPlayer.setOnMarker(new EventHandler<MediaMarkerEvent>() {
    @Override
    public void handle(MediaMarkerEvent ev){
       if (ev.getMarker().getKey().equals("onEdge")) {
            mediaPlayer.pause();
            prompt.setText("Will Percy jump?");
            dialogBox.setVisible(true);
        }
    }
});
```

Popup Dialog Box for Video Interaction

```
final BorderPane dialogBox = new BorderPane();
final Label prompt = new Label();
prompt.setStyle("-fx-font: 20pt 'sans-serif'");
dialogBox.setCenter(prompt);
dialogBox.setStyle("-fx-background-color: lightgrey;");
dialogBox.setMaxHeight(BOXHEIGHT);
dialogBox.setMaxWidth(BOXWIDTH);
dialogBox.setPadding(new Insets(10));
final HBox buttons = new HBox();
final Button yesButton  = new Button("Yes");
final Button noButton  = new Button("No");
buttons.setAlignment(Pos.CENTER);
buttons.getChildren().add(yesButton);
buttons.getChildren().add(noButton);
dialogBox.setBottom(buttons);
```



Video credits:
Benno Kühnl, Christian Becker,
Sarah Torma, Thomas Burghart
WS 2012/13

User-Controlled Video Continuation

```
yesButton.setOnAction(new EventHandler<ActionEvent>() {
    public void handle(ActionEvent e) {
        mediaPlayer.seek(markers.get("noJump"));
        mediaPlayer.play();
        dialogBox.setVisible(false);
    }
});

noButton.setOnAction(new EventHandler<ActionEvent>() {
    public void handle(ActionEvent e) {
        mediaPlayer.seek(markers.get("jump"));
        mediaPlayer.play();
        dialogBox.setVisible(false);
     }
});
```

How to Realize Real Interaction in Video?

- Real interaction means:
  - Pointing to regions in video window identifies objects
  - Clicking on a region or symbol modifies video scene
- Scene needs to be *decomposed:*
  - Parts/objects of video playback can be made (in)visible by script code
  - Objects can be moved around in video
- Easy solution:
  - *Overlaying* of videos
- Two main techniques:
  - *Masking* cuts out specific parts from a video
    - » Prerequisite: Objects are easy to identify and do not move much
  - *Alpha channel* video overlays
    - » Prerequisite: External production of video with alpha channel
    - » Using video effect software (e.g. AfterEffects)