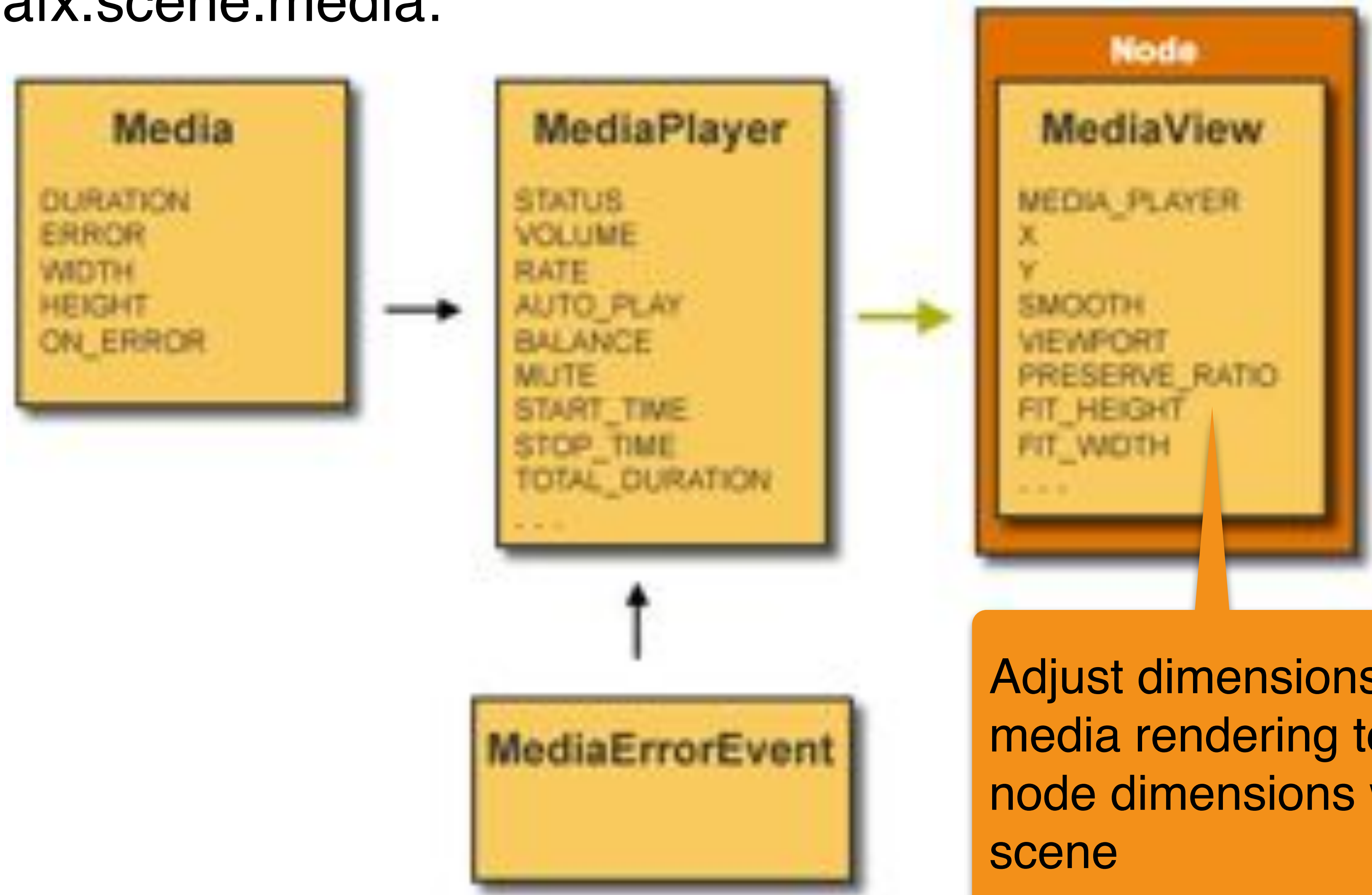
# 9 Programming with Video

Literature:

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

## Video Playback with JavaFX

javafx.scene.media:



Adjust dimensions of media rendering to view node dimensions within scene

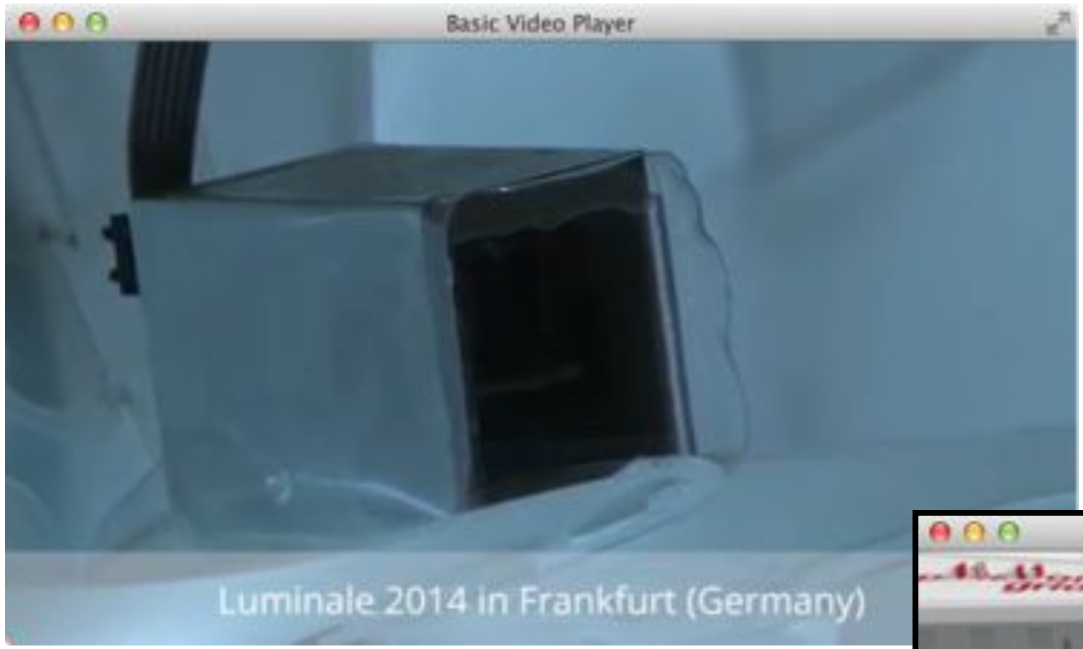
# Basic Video Playback Application

```
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

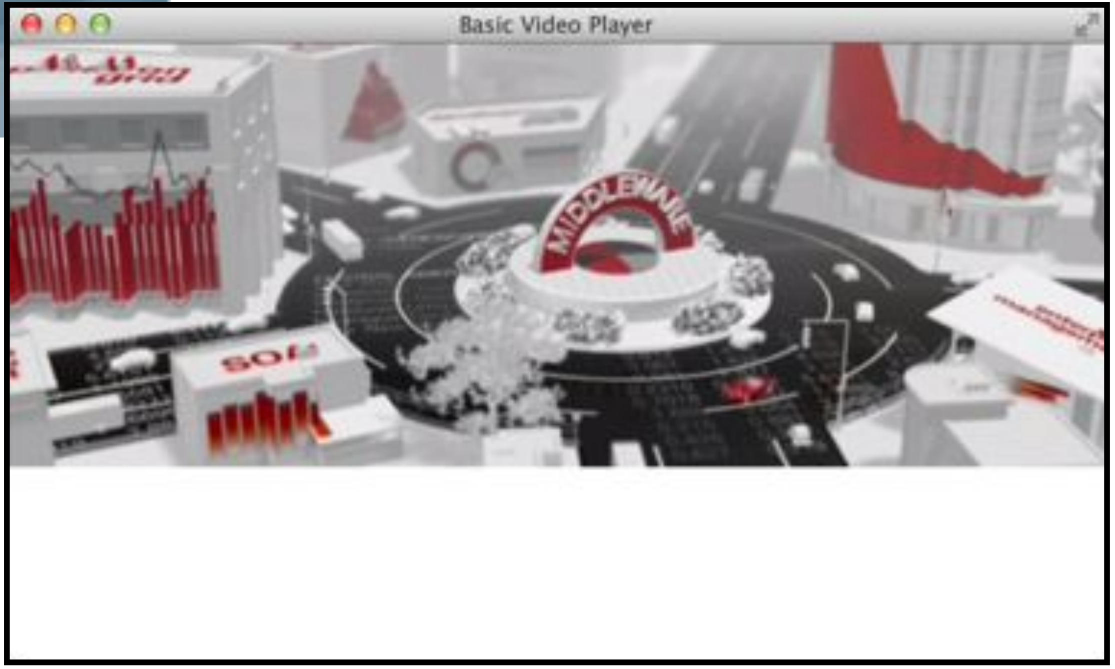
```
FileChooser fileChooser = new FileChooser();
fileChooser.setTitle("Please select video file");
File file = fileChooser.showOpenDialog(primaryStage);
if (file != null) {
  String mediaURI = file.toURI().toString();
  try {
   Media media = new Media(mediaURI);
    MediaPlayer mediaPlayer = new MediaPlayer(media);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

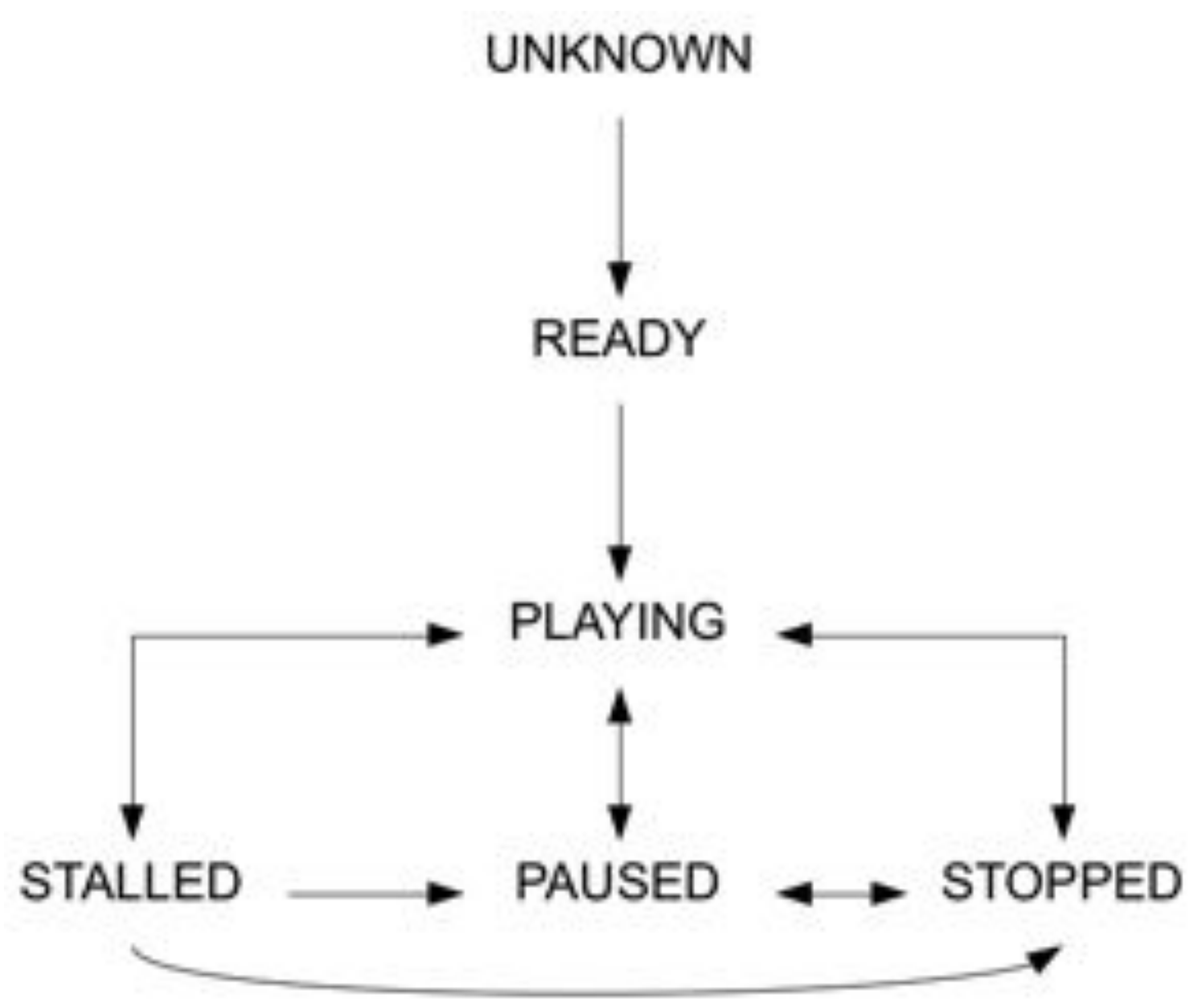


Video aspect ratio
equal to scene
(640x360 = 16:9)

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.

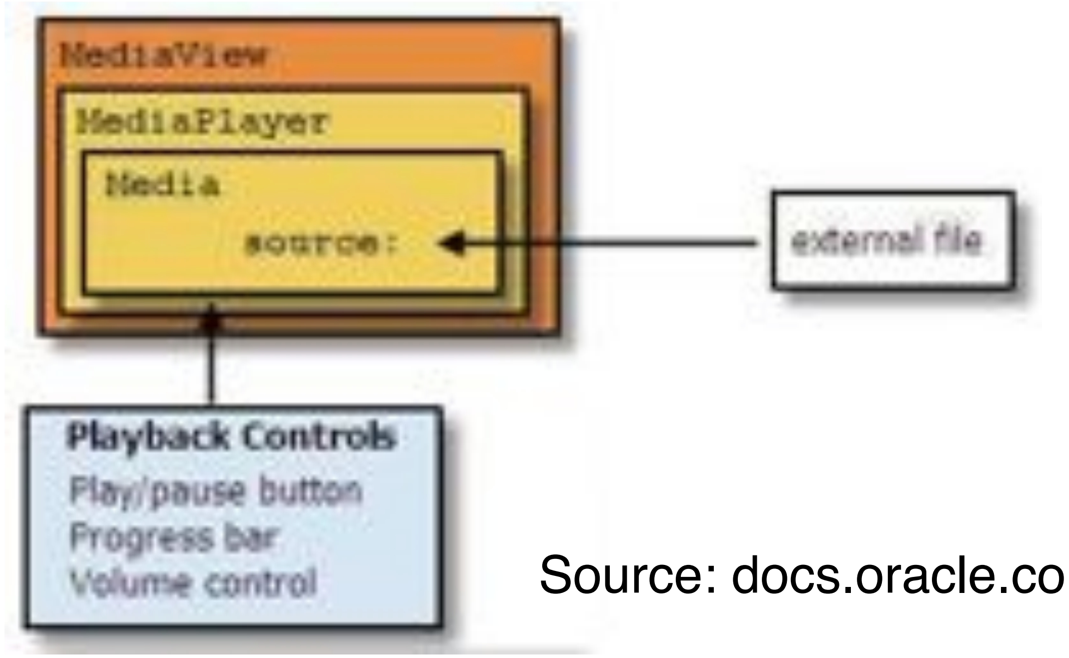
QUIZ:
How can we adapt our display to media aspect ratio?

## State Transition Listener

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

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

*Alternative property of MediaView getBoundsInLocal():*
The rectangular bounds of this Node in the node's untransformed local coordinate space. […]
This property will always have a non-null value.
Note that boundsInLocal is automatically recomputed whenever the geometry of a node changes.

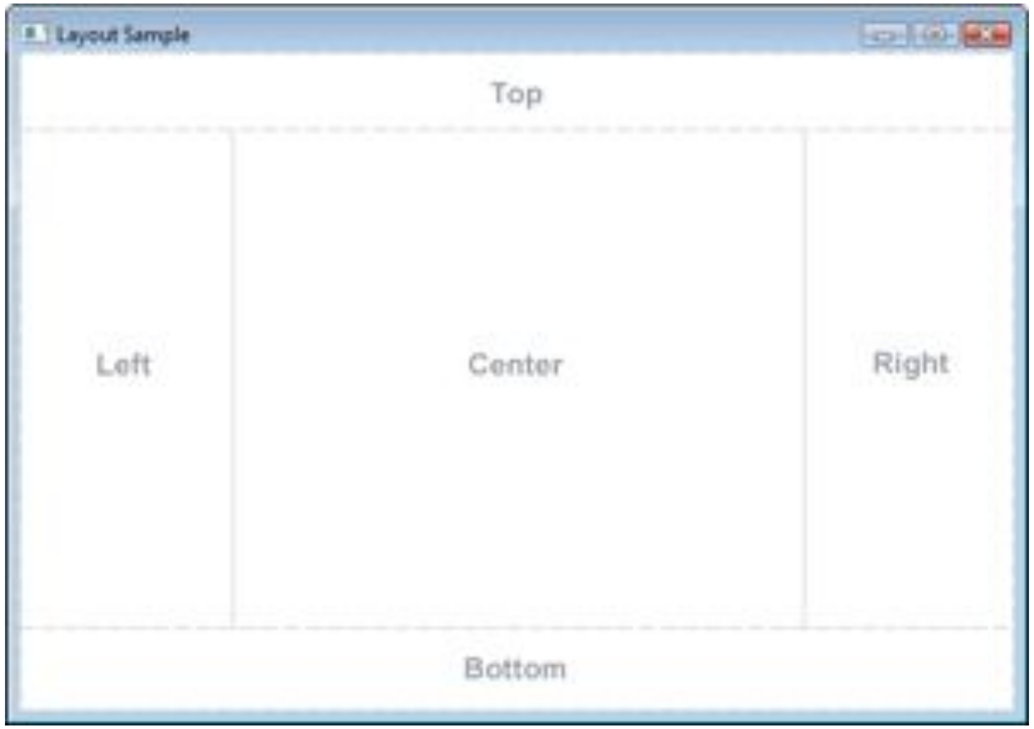
## Controlling Media Playback



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

# BorderPane Node Layout



```
BorderPane borderpane = new BorderPane();
ToolBar toolbar = new ToolBar();
HBox statusbar = new HBox();
Node appContent = new AppContentNode();
borderPane.setTop(toolbar);
borderPane.setCenter(appContent);
borderPane.setBottom(statusbar);
```

Source: docs.oracle.com

# Control Bar for Video Playback

```
BorderPane root = new BorderPane();
Scene scene = new Scene(root);
…
root.setCenter(mediaView);
MediaControl mediaControl = new MediaControl(mediaPlayer);
root.setBottom(mediaControl);
```

```
public class MediaControl extends HBox {
    private MediaPlayer mp;
    … // local UI controls

    public MediaControl(final MediaPlayer mp) {
        this.mp = mp;
        setStyle("-fx-background-color: grey");
        setAlignment(Pos.CENTER);
        …

    protected void updateValues() {…}
}
```

## Variants of UI Control Elements

- Example: Control buttons
  - Similar for progress bar, volume slider etc.

(a) Control buttons integrated into pre-fabricated playback component

(Possibly) adjustable by parameters or skins

(b) Pre-fabricated control button components

Flash: Video components “PauseButton”, “PlayButton” etc.

(c) Use of standard UI control components (used here)

Customized for specific purpose

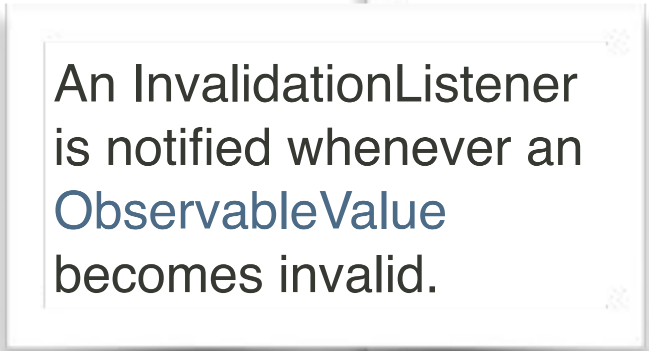
(d) Use of pre-designed UI components

Bought from external source

Loaded from external library

# Regular Updates During Playback

- For keeping UI up to date:
  - Update progress bar, elapsed time display etc.
- Similar to “tick” event in timeline
  - but video playback has intrinsic timeline
- Platform-dependent
- JavaFX (one possible solution):
  - Observe property “currentTime” of media player (invalidation)
  - React “lazily” in regular time interval, not eagerly at change time!


An InvalidationListener is notified whenever an ObservableValue becomes invalid.

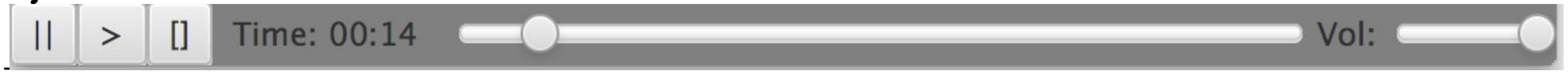
```
mp.currentTimeProperty().addListener(
  new InvalidationListener() {
    public void invalidated(Observable ov) {
        updateValues();
    }
});
```

## Designing a Playback Control Bar (1)

```
public class MediaControl extends HBox {
    private MediaPlayer mp;
    private Slider timeSlider;
    private Label playTime;
    private Slider volumeSlider;
    private Duration totalDuration;

    public MediaControl(final MediaPlayer mp) {…
      // Add buttons
      final Button pauseButton  = new Button("||");
      final Button playButton  = new Button(">");
      final Button stopButton  = new Button("[]");
      this.getChildren().add(pauseButton);
      this.getChildren().add(playButton);
      this.getChildren().add(stopButton);
      // Add Time label
      Label timeLabel = new Label("  Time: ");
      this.getChildren().add(timeLabel);
      // Add Play label
      playTime = new Label("00:00");
      playTime.setPrefWidth(50);
      playTime.setMinWidth(50);
      this.getChildren().add(playTime);  … contd
```

## Designing a Playback Control Bar (2)

```
public class MediaControl extends HBox {
    private MediaPlayer mp;
    private Slider timeSlider;
    private Label playTime;
    private Slider volumeSlider;
    private Duration totalDuration;
    …
    public MediaControl(final MediaPlayer mp) {…
      // Add time slider
      timeSlider = new Slider();
      HBox.setHgrow(timeSlider, Priority.ALWAYS);
      this.getChildren().add(timeSlider);
      // Add Volume label
      Label volumeLabel = new Label("Vol: ");
      this.getChildren().add(volumeLabel);
      // Add Volume slider
      volumeSlider = new Slider();
      volumeSlider.setPrefWidth(70);
      volumeSlider.setMinWidth(30);
      this.getChildren().add(volumeSlider);
    }
}
```



# Handling User Input Events

```
pauseButton.setOnAction(new EventHandler<ActionEvent>() {
  public void handle(ActionEvent e) {
    mp.pause();
  }
}); …

volumeSlider.valueProperty().addListener(new
   InvalidationListener() {
  public void invalidated(Observable ov) {
    if (volumeSlider.isValueChanging()) {
      mp.setVolume(volumeSlider.getValue() / 100.0);
    }
  }
});

timeSlider.valueProperty().addListener(new
    InvalidationListener() {
  public void invalidated(Observable ov) {
    if (timeSlider.isValueChanging()) {
      mp.seek(totalDuration.multiply(
        timeSlider.getValue() / 100.0));
    }
  }
});
```

## Update Interface View

```
protected void updateValues() {
    Duration currentTime = mp.getCurrentTime();
    totalDuration = mp.getMedia().getDuration();
    int currentSecs = (int)Math.floor(currentTime.toSeconds());
    int minutes = currentSecs / 60;
    int seconds = currentSecs - minutes * 60;
    playTime.setText(
      String.format("%02d:%02d", minutes, seconds));
    timeSlider.setValue(
      currentTime.toMillis()/totalDuration.toMillis()*100);
    volumeSlider.setValue(
      (int)Math.round(mp.getVolume() * 100));
}
```

# 9 Programming with Video

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

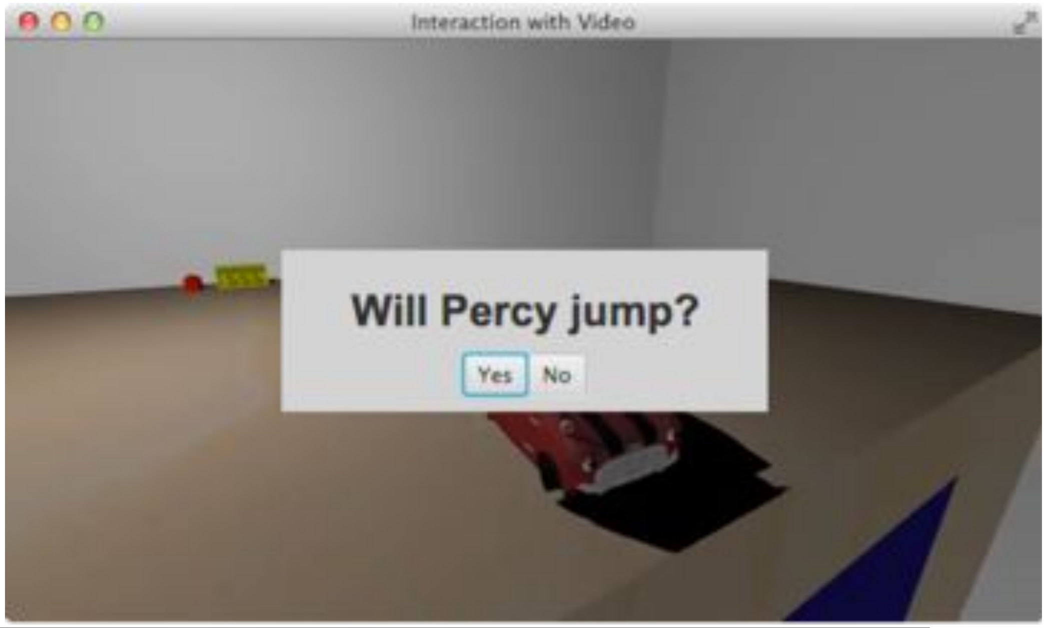
- A *cue point* marks a specific point in time during media playback.
  - Specification by *time stamp* relative to media start time
  - Flash/ActionSript: “cue point”
  - JavaFX: “Media marker”
- Internal cue point: Embedded into movie file
  - Supported by some video formats
- External cue point: Defined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

# 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

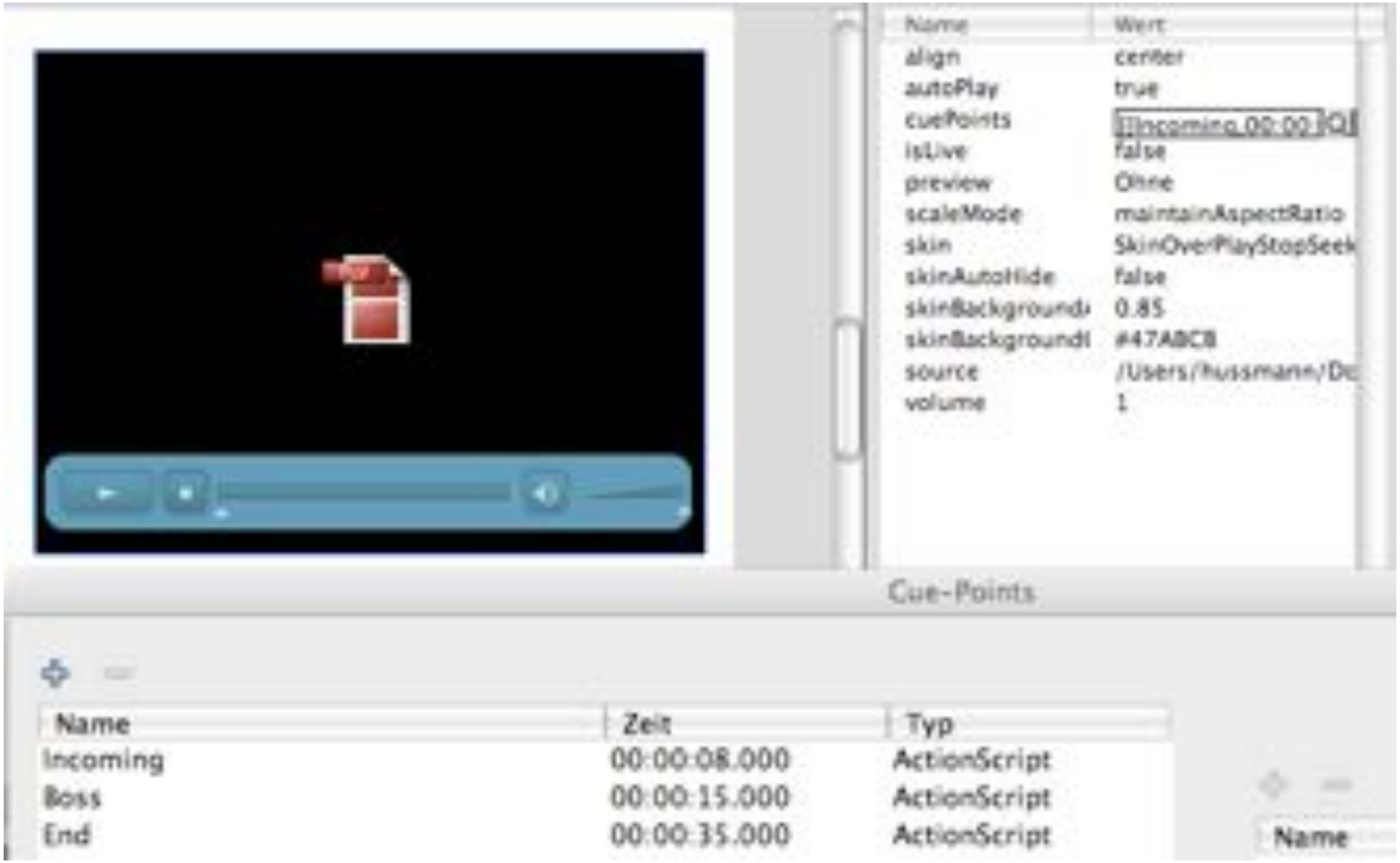
```
final BorderPane dialogBox = new BorderPane();
final Label prompt = new Label();
prompt.setStyle("-fx-font: 20pt 'sans-serif'");
dialogBox.setCenter(prompt);
dialogBox.setStyle("-fx-background-color: lightgrey;");
dialogBox.setMaxHeight(BOXHEIGHT);
dialogBox.setMaxWidth(BOXWIDTH);
dialogBox.setPadding(new 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);
```



# 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);
    }
});
```

# Adobe Flash Cue Points (1)

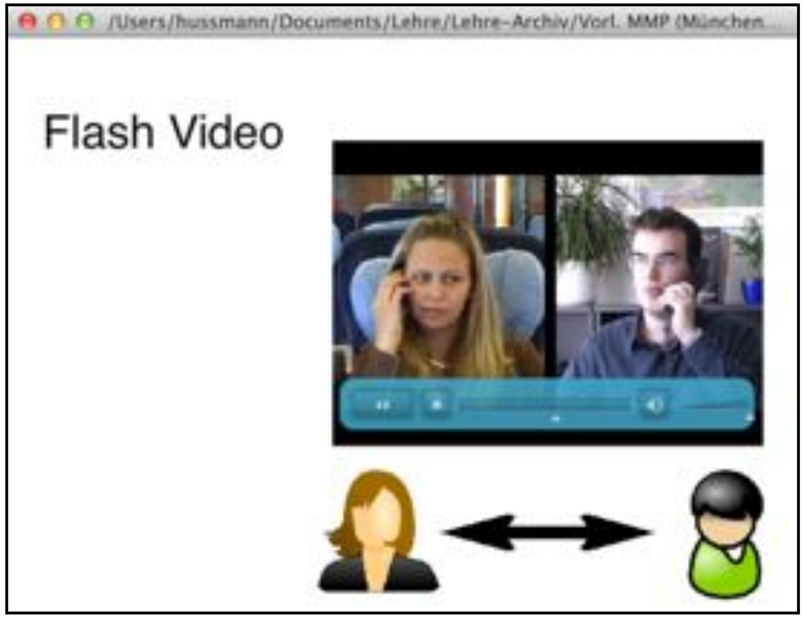


| Name | Wert |
|---|---|
| align | center |
| autoPlay | true |
| cuePoints | [{Incoming 00:00 |
| isLive | false |
| preview | Ohne |
| scaleMode | maintainAspectRatio |
| skin | SkinOverPlayStopSeek |
| skinAutoHide | false |
| skinBackground/ | 0.85 |
| skinBackgroundI | #47ABCB |
| source | /Users/hussmann/Dc |
| volume | 1 |

Cue-Points

| Name | Zeit | Typ |
|---|---|---|
| Incoming | 00:00:08.000 | ActionScript |
| Boss | 00:00:15.000 | ActionScript |
| End | 00:00:35.000 | ActionScript |

# Adobe Flash Cue Points (2)


/Users/hussmann/Documents/Lehre/Lehre-Archiv/Vorl. MMP (München...
Flash Video

- Video Player component
  - Registration of cue points
- ActionScript 3.0
  - Registration of cue point event handlers
- Example: Animation synchronized with video

```
videoplayer.addEventListener(MetadataEvent.CUE_POINT, cp_listener);
function cp_listener(e:MetadataEvent):void {
      if (e.info.name == "Incoming") {
            sekr.visible = true;
            pfeil1.visible = true;
      }
      if (e.info.name == "Boss") {
            sekr.visible = false;
            boss.visible = true;
            pfeil2.visible = true;
            pfeil1.visible = 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)

## Application Examples (1)

**#01: Gipsy Voices**

Ein Hauptmenü in Form einer leeren Bühne lädt den Nutzer dazu ein, spielerisch zu erkunden, welche Musiker in der Band "Gipsy Voices" spielen. Klickt der Anwender eine Person an, öffnet sich ein Fenster, das mehrere Videos zur jeweiligen Person bietet.

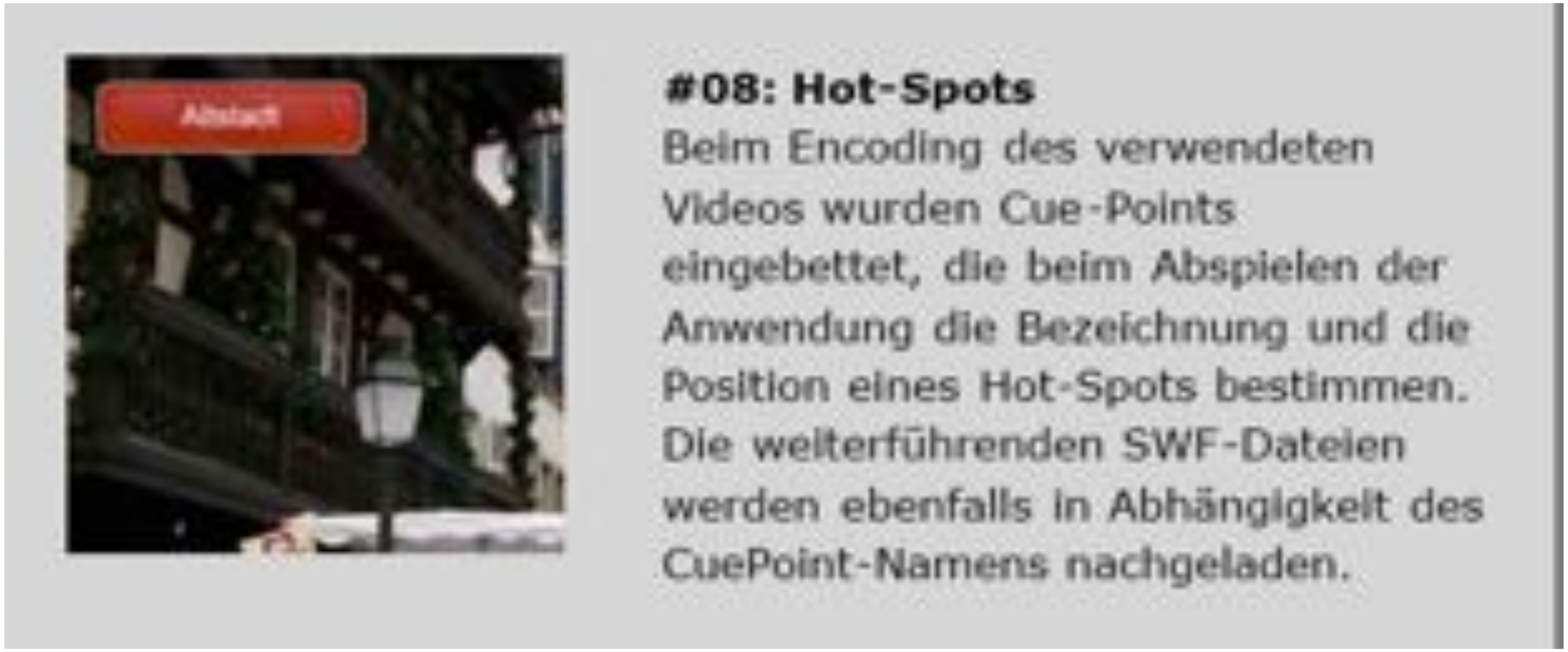
www.video-flash.de

## Application Examples (2)

**#02: Hutshop**
Dieser Entwurf soll einen Eindruck vermitteln, wie eine Produktpräsentation mit Videos im Internet ausssehen könnte. Es werden freigestellte Videos verwendet.

www.video-flash.de

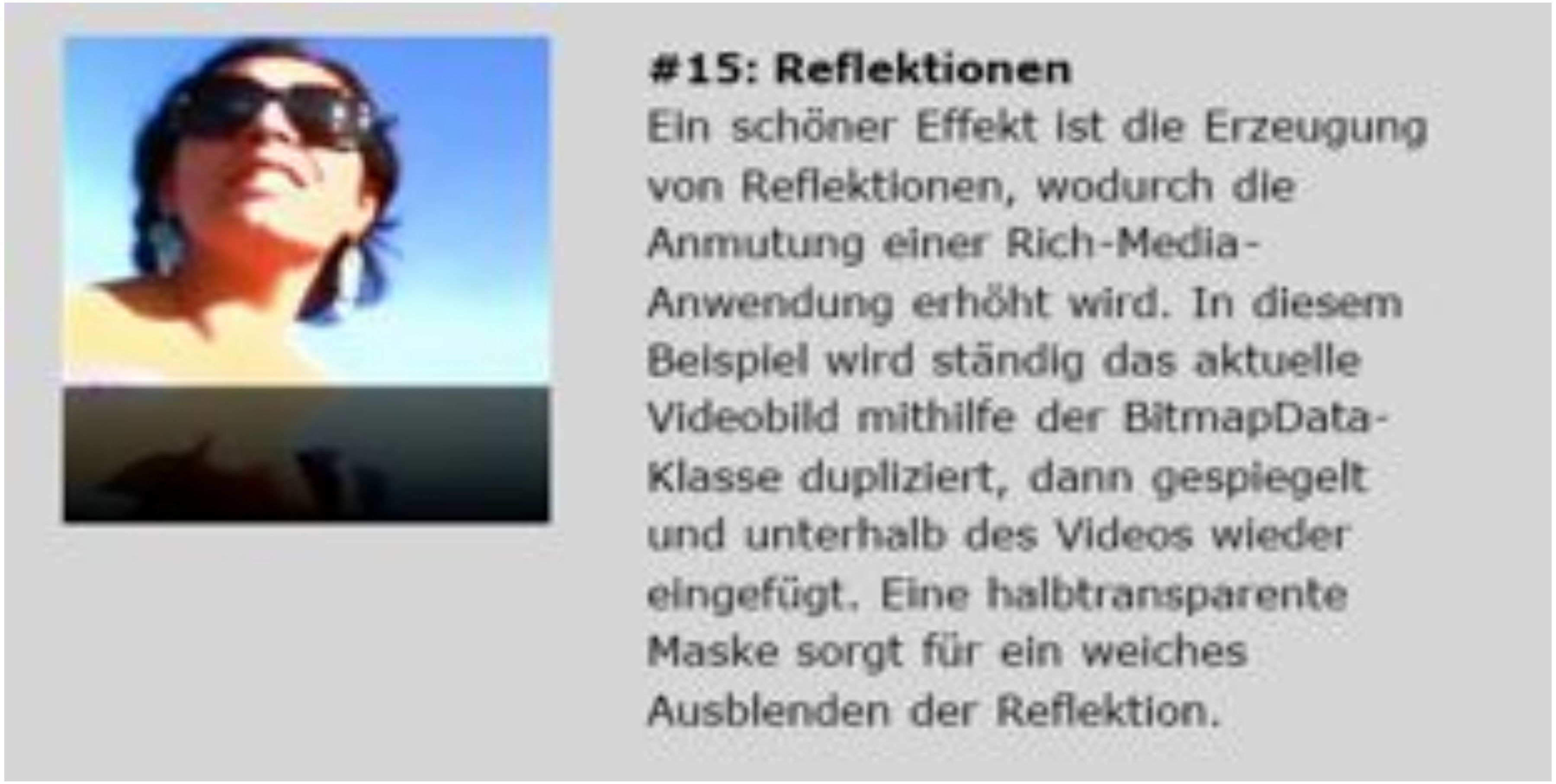
## Application Examples (3)



**#08: Hot-Spots**
Beim Encoding des verwendeten Videos wurden Cue-Points eingebettet, die beim Abspielen der Anwendung die Bezeichnung und die Position eines Hot-Spots bestimmen. Die weiterführenden SWF-Dateien werden ebenfalls in Abhängigkeit des CuePoint-Namens nachgeladen.

www.video-flash.de

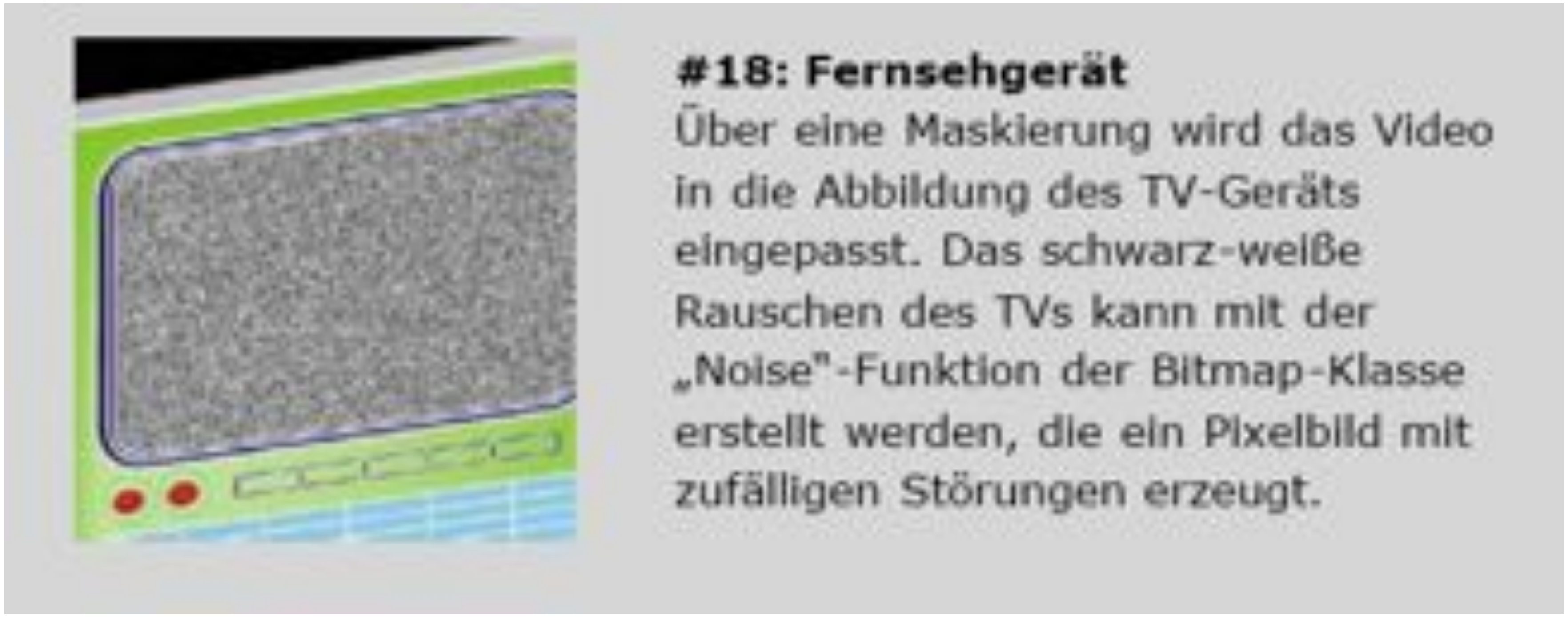
# Application Examples (4)



**#15: Reflektionen**
Ein schöner Effekt ist die Erzeugung von Reflektionen, wodurch die Anmutung einer Rich-Media-Anwendung erhöht wird. In diesem Beispiel wird ständig das aktuelle Videobild mithilfe der BitmapData-Klasse dupliziert, dann gespiegelt und unterhalb des Videos wieder eingefügt. Eine halbtransparente Maske sorgt für ein weiches Ausblenden der Reflektion.

www.video-flash.de

## Application Examples (5)



**#18: Fernsehgerät**
Über eine Maskierung wird das Video in die Abbildung des TV-Geräts eingepasst. Das schwarz-weiße Rauschen des TVs kann mit der „Noise"-Funktion der Bitmap-Klasse erstellt werden, die ein Pixelbild mit zufälligen Störungen erzeugt.

www.video-flash.de

# 9 Programming with Video

Literature:

Clemens Szyperski: Component Software - Beyond Object-Oriented Programming. 2nd ed. Addison-Wesley 2002

M.D. McIlroy: Mass produced software components. In: Naur/Randell (eds.), Software Engineering, NATO Scientific Affairs Div. 1969 (http://www.cs.dartmouth.edu/~doug/components.txt)

# Software Components

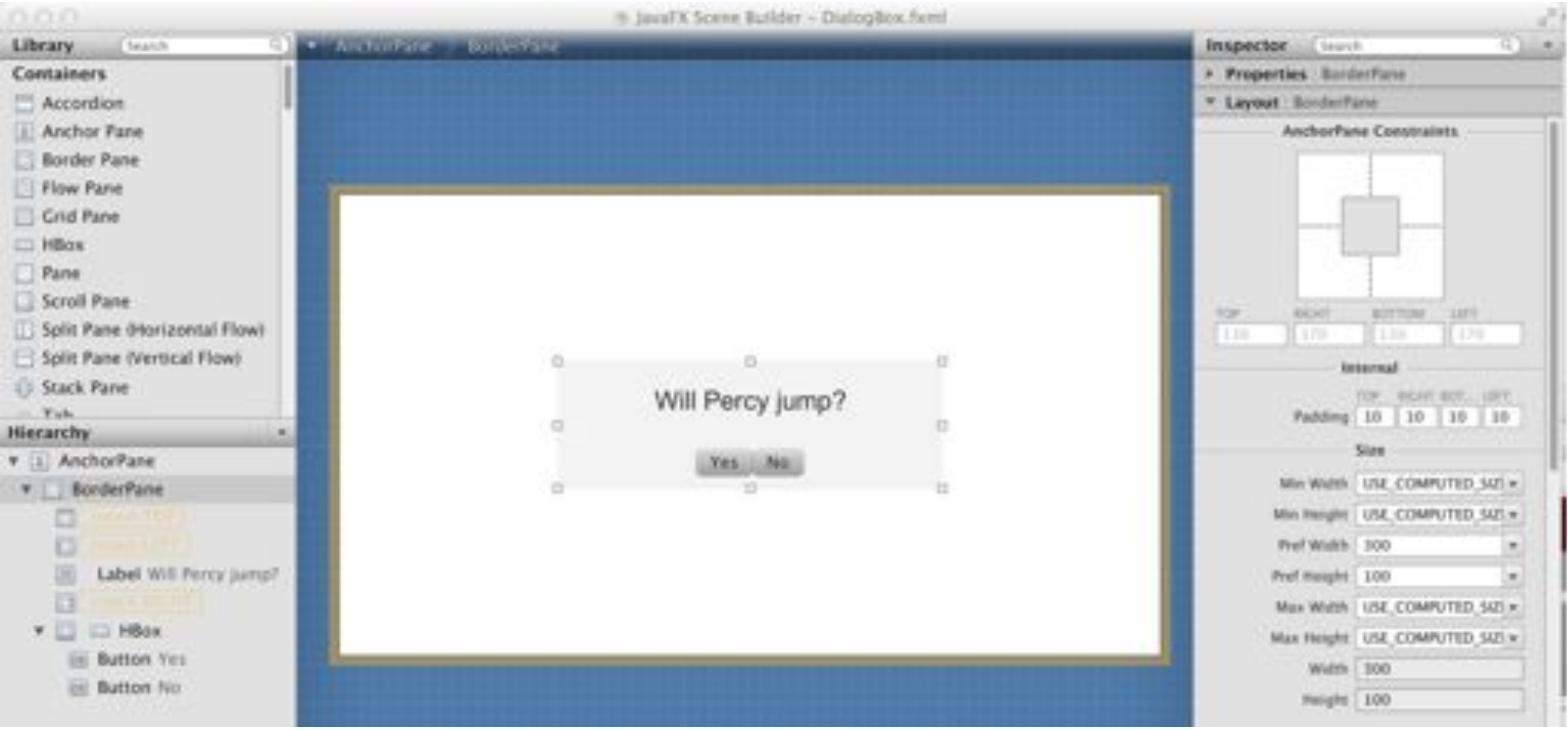
- *Software component:* “A ***software component*** is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties.”

  ECOOP 1996, Workshop on Component-oriented Programming
- Components for visual development environments:
  - Provide well-defined functionality
  - Can be dragged from palette to working area (creating an instance)
  - Can be adjusted by setting parameter values with a *component inspector*
  - Can, in some environments, be connected to other components using visual metaphors
    - » Connecting input and output “ports” with “lines”
- Component is also accessible through programming (API):
  - Parameters can be manipulated by program code (getter, setter)
- There is a marketplace for components
  - Custom components
  - Building blocks for software

## Software Components for Java: Java Beans

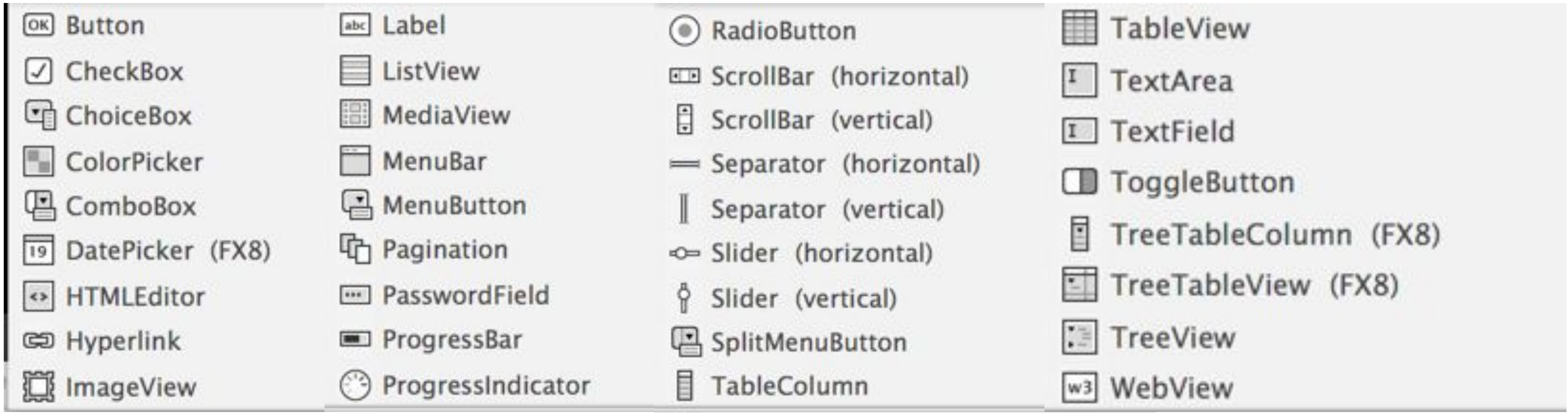
- Simple conventions make a Java class a Java “Bean”
  - Public constructor without parameters
  - Serializability
  - Public getter/setter methods for all local properties
    - » *type* get*Prop*()
    - » void set*Prop*(*type* v)
- All Swing and JavaFX controls are Java Beans
- Developer tools can easily manipulate beans and their properties

# Dialog Box as Component Composition

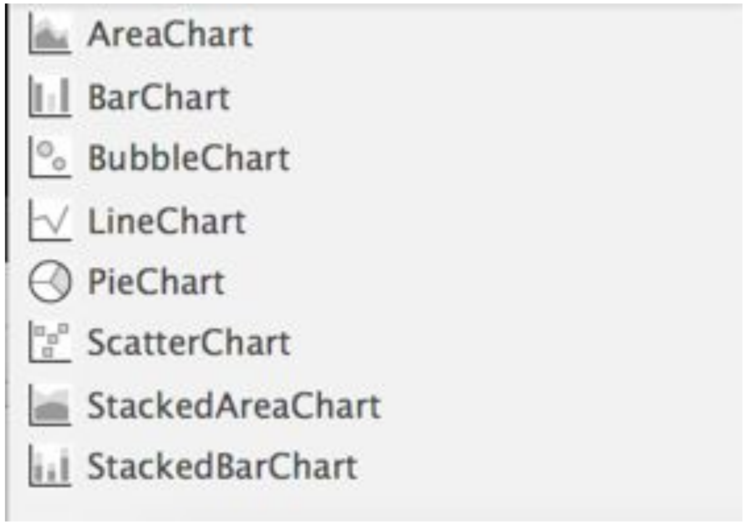


Padding 10 10 10 10
Min Width USE_COMPUTED_SIZE
Min Height USE_COMPUTED_SIZE
Pref Width 300
Pref Height 100
Max Width USE_COMPUTED_SIZE
Max Height USE_COMPUTED_SIZE
Width 300
Height 100

# JavaFX Controls (Java Beans)



| | | | |
|---|---|---|---|
| Button | Label | RadioButton | TableView |
| CheckBox | ListView | ScrollBar (horizontal) | TextArea |
| ChoiceBox | MediaView | ScrollBar (vertical) | TextField |
| ColorPicker | MenuBar | Separator (horizontal) | ToggleButton |
| ComboBox | MenuButton | Separator (vertical) | TreeTableColumn (FX8) |
| DatePicker (FX8) | Pagination | Slider (horizontal) | TreeTableView (FX8) |
| HTMLEditor | PasswordField | Slider (vertical) | TreeView |
| Hyperlink | ProgressBar | SplitMenuButton | WebView |
| ImageView | ProgressIndicator | TableColumn | |



- AreaChart
- BarChart
- BubbleChart
- LineChart
- PieChart
- ScatterChart
- StackedAreaChart
- StackedBarChart

## “Custom Controls” = Marketplace

CONTROLS FX

JFXtras

A set of high quality controls and add-ons for JavaFX.

fexperience.com/controlsfx
jfxtras.org
www.mrlonee.com