JavaOne™

JavaOne™

JavaFX Extreme GUI Makeover

Simon Ritter, Angela Caicedo
Technology Evangelists

MAKE THE FUTURE JAVA

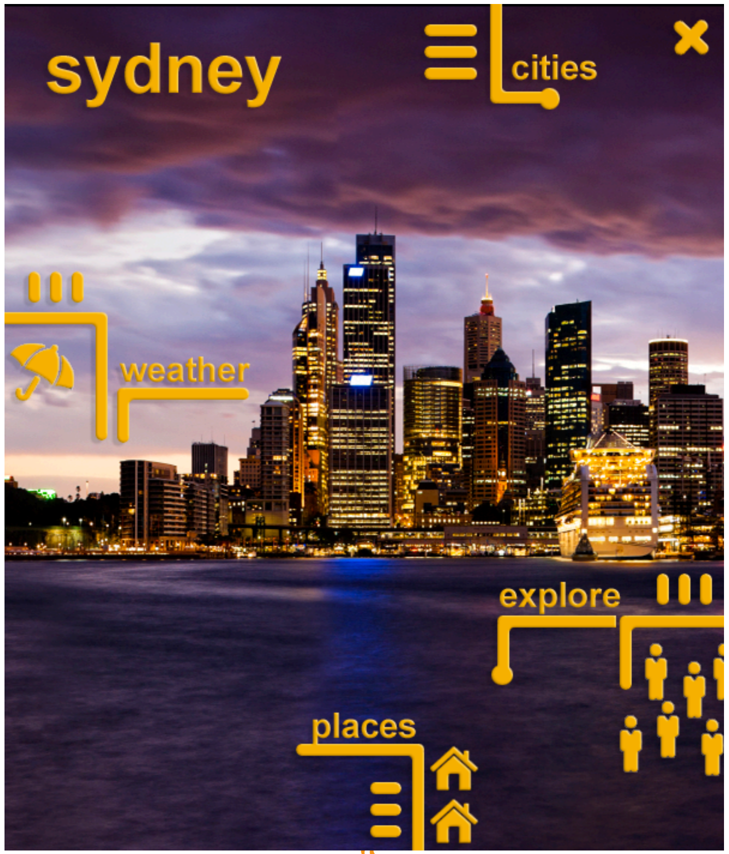
ORACLE®

Program Agenda

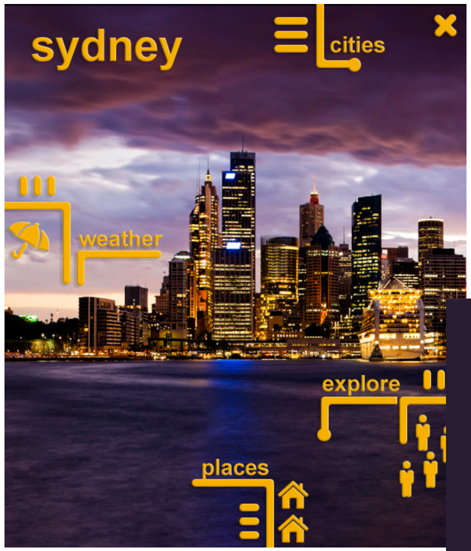
- Demo: Let’s see what we can build
- JavaFX and CSS
- JavaFX effects, animations and other cool features.
- Two approaches, two applications, you choose.
- Tools and demos

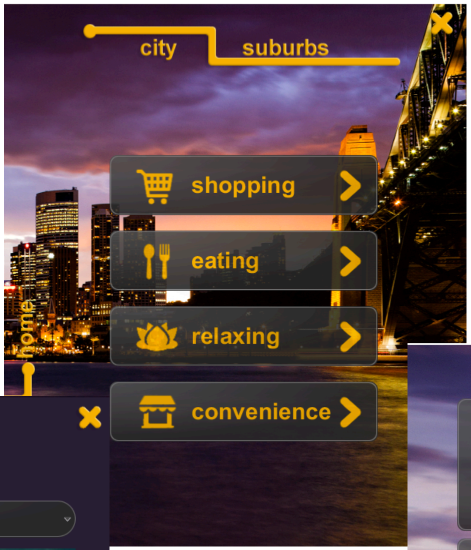
City Explorer Demo

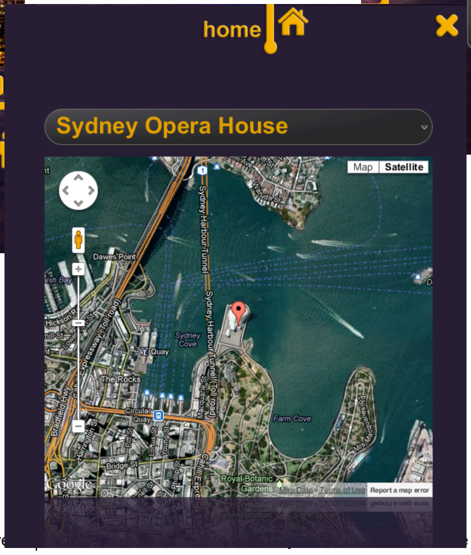
- Non-conventional interfaces
- Modern feeling
- Dynamic content
- Intuitive and easy to use
- Lots of animations
- Combinations of technologies: Java, JavaScript, HTML5.

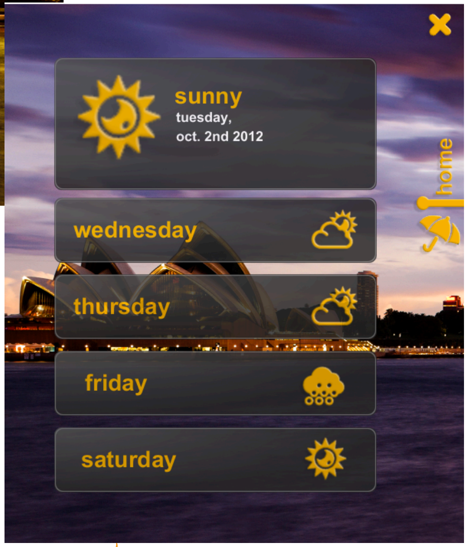


City Explorer Demo





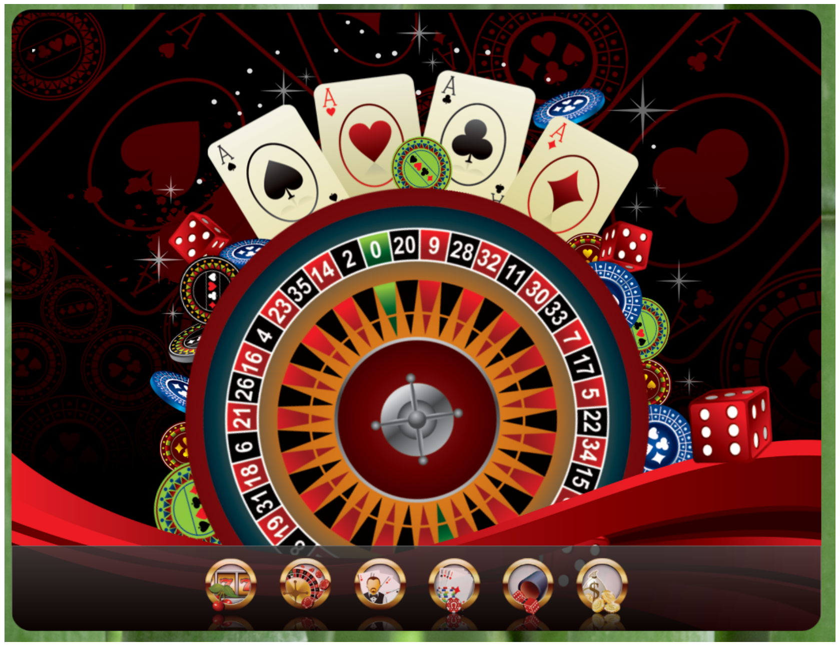




Casino Application for the Real World

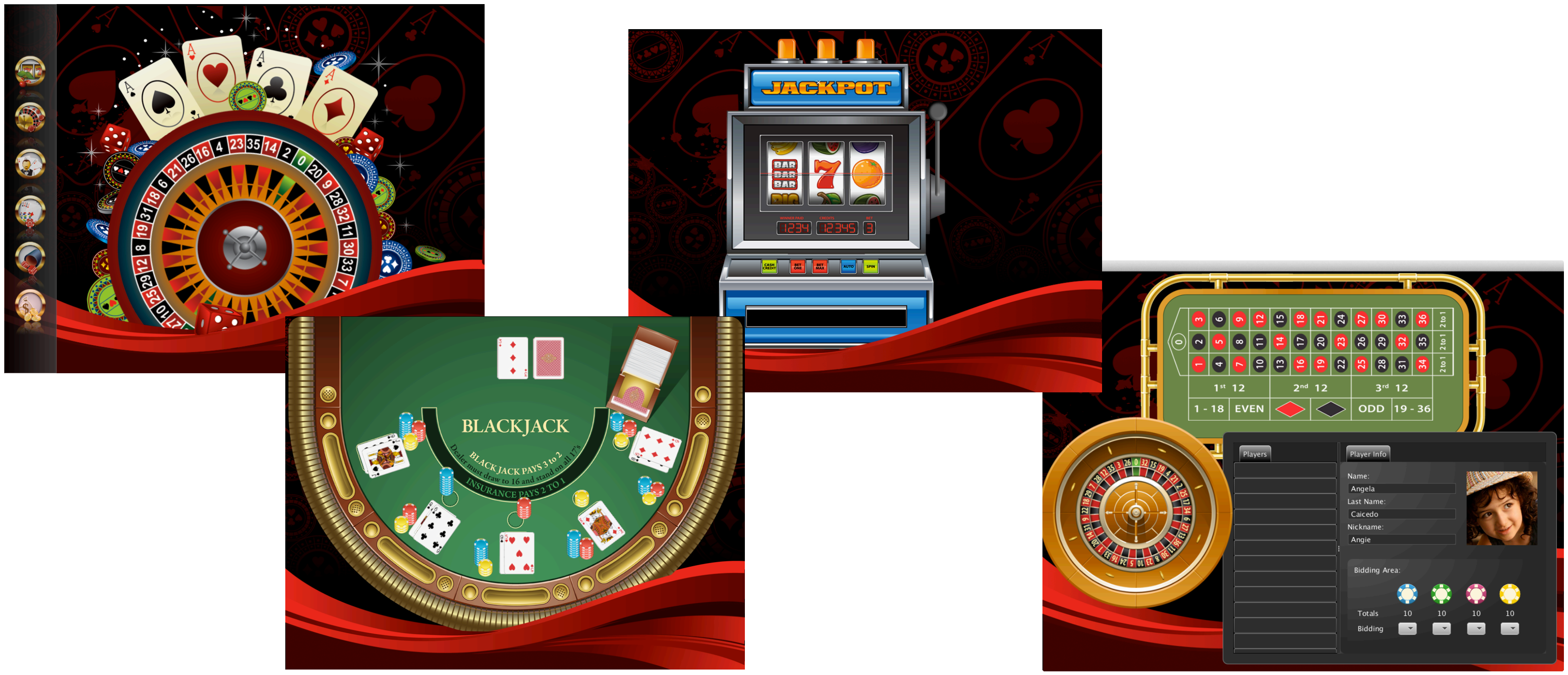
Join our session **CON5352, Tuesday 3pm**

- Dynamic slide-in menus
- Semi-transparent bar
- Customized buttons:
  - Reflection
  - Zoom in feature
  - Pushed effect
- Human interaction:
  - Gesture recognition
  - Neurosky (mind reader)



Casino Application for the Real World

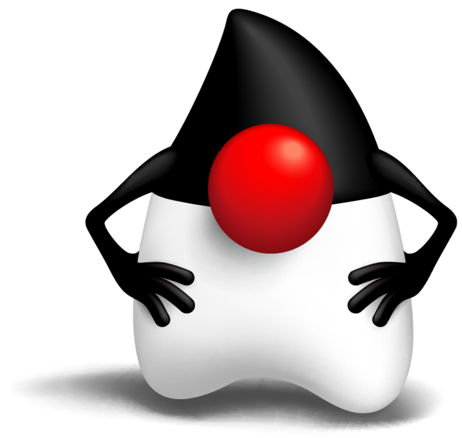
Join our session **CON5352, Tuesday 3pm**



Name: Angela
Last Name: Caicedo
Nickname: Angie

Totals	10	10	10	10
Bidding				

Demo



JavaFX and CSS

Skinning JavaFX Application with CSS

- Create a custom look - > Skin
- Create style definitions that control the look of user interface elements
- CSS in JavaFX applications is similar to using CSS in HTML.
- JavaFX CSS are based on the W3C CSS version 2.1 specification
  - http://www.w3.org/TR/CSS21/
  - Some additions from current work on version 3 of the specification
  - Some extensions that support specific JavaFX features.
- Enables you to change the just by changing the style sheet used.

Why CSS?

- CSS is a domain specific language
  - Very good for declaring visual effects
- CSS empowers designers
- CSS is a standard
- CSS is widely adopted
- Interoperability

CSS and the JavaFX Scene Graph

- CSS styles are applied to nodes in the JavaFX scene graph
- Styles are first applied to the parent, then to its children.
- CSS styles are applied asynchronously.
- Each node in the scene graph has a **styleClass** variable, a List<String>.
- Each node in the scene graph has an **id** variable, a string. Styles for specific ids can be specified using the "#nodeid" selector syntax in a style sheet.

Creating your StyleSheet

- Create one or more sheets to override the default styles.
- Create your own styles
- Style sheets have an extension of .css

JavaFX and CSS

```
Stage stage = new Stage();
Label label = new Label();
Label.setText(“Hello World”);
Stage.getScene().getContent().add(label);
Stage.setVisible(true);
```

JavaFX and CSS

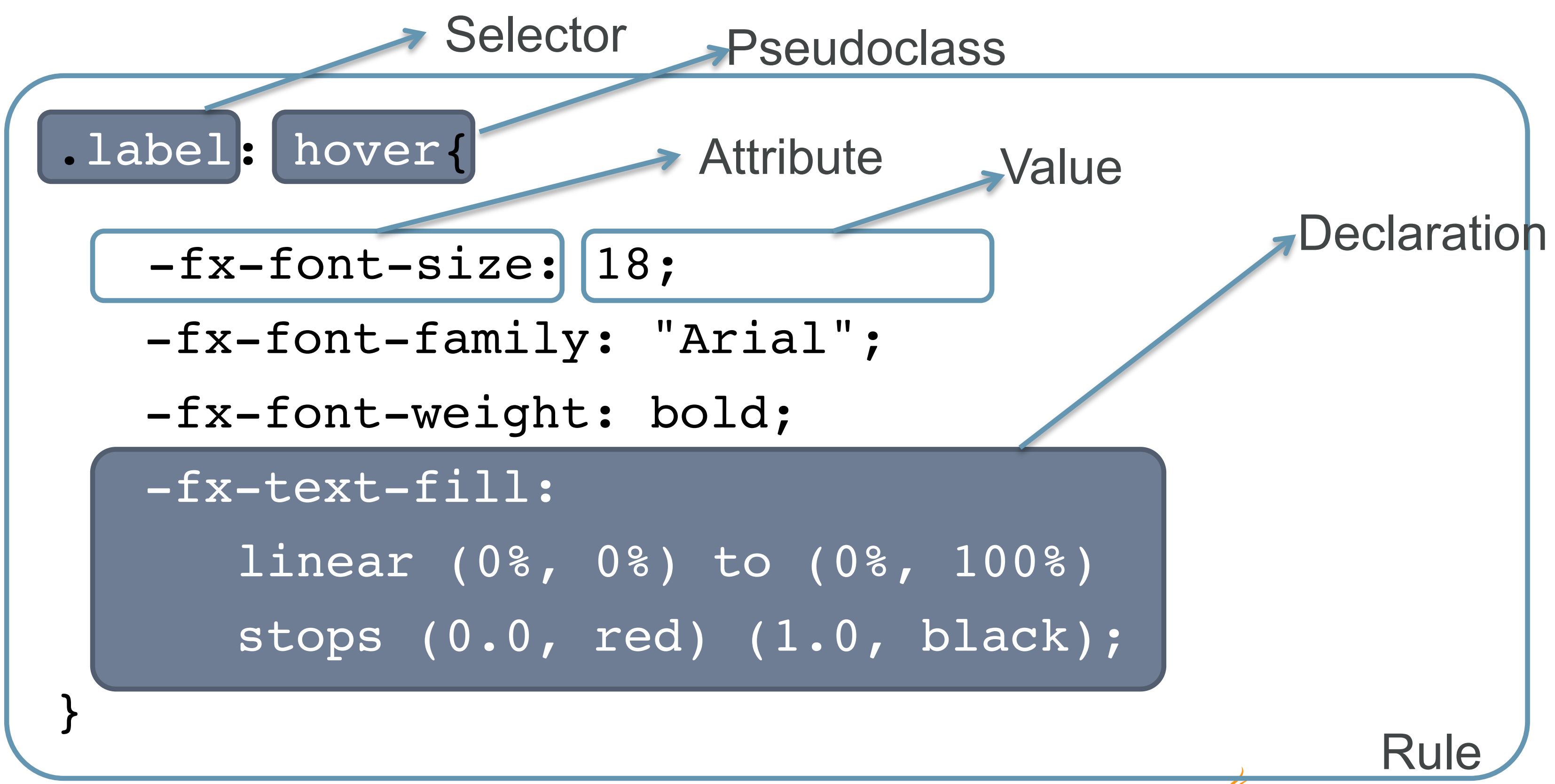
```
Stage stage = new Stage();
Label label = new Label();
label.setText(“Hello World”);
Scene scene = stage.getScene();
scene.getContent().add(label);
scene.getStylesheets().add(“/myCSS.css”);
stage.setVisible(true);
```

What is a “selector”?

- A pattern used to match a Node in the scene.
- Match against the Node’s class, styleClass, id, and pseudo-class state (hover, pressed, selected, focused, etc)

.label {…}	Matches any Node with styleClass “label”. Normally they correspond to class names. .button for Button, label for Label classes
#title {…}	Matches any Node with id “title”
* {…}	Matches any Node
.label:hover {…}	Matches any Node with styleClass “label” and “hover” equal to true
.check-box .label	Compound styles

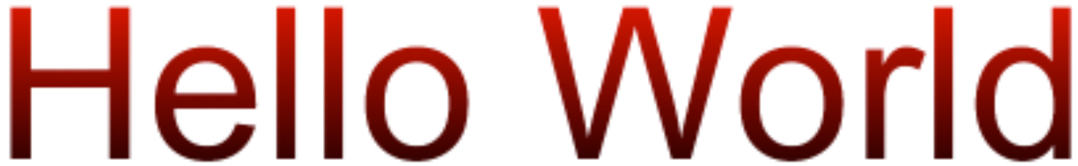
CSS Syntax



```
.label:hover{
   -fx-font-size: 18;
   -fx-font-family: "Arial";
   -fx-font-weight: bold;
   -fx-text-fill:
      linear (0%, 0%) to (0%, 100%)
      stops (0.0, red) (1.0, black);
}
```

myCSS.css

```
.label{
   -fx-font: “Amble”;
   -fx-fong-size: 18;
   -fx-text-fill:
     linear (0%, 0%) to (0%, 100%)
     stops (0.0, red) (1.0, black);
}
```

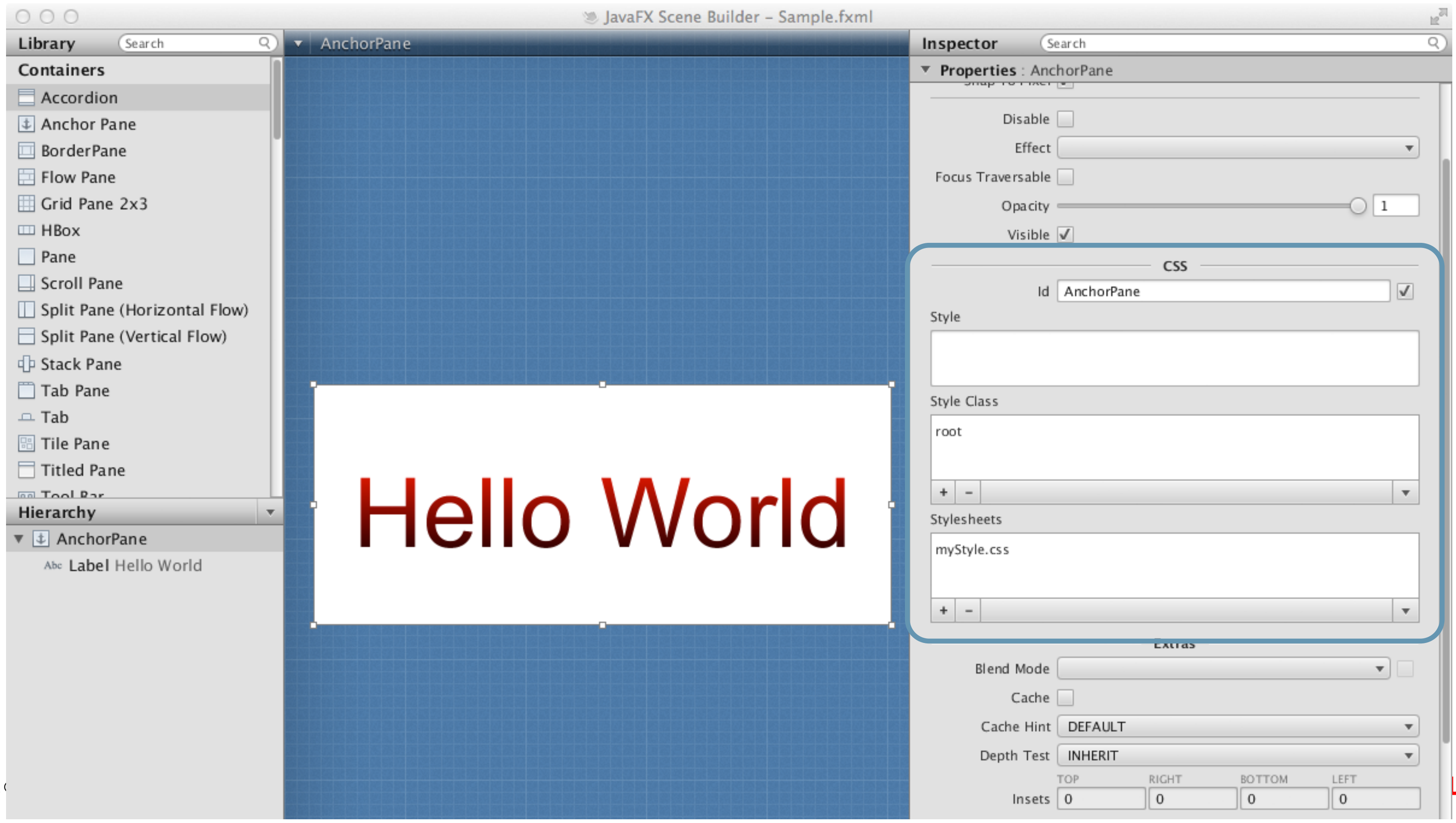

Hello World

myCSS.css

```
.my-label{
   -fx-font: “Amble”;
   -fx-fong-size: 18;
   -fx-text-fill:
     linear (0%, 0%) to (0%, 100%)
     stops (0.0, red) (1.0, black);
}
```

```
Label myStyledLabel = new Label(“Testing”);
myStyledLabel.getStyleClass().add(“my-label”);
```

JavaFX, CSS and Scene Builder



Disable
Effect
Focus Traversable
Opacity 1
Visible ✓

CSS

Id AnchorPane ✓
Style
Style Class
root
Stylesheets
myStyle.css

Extras

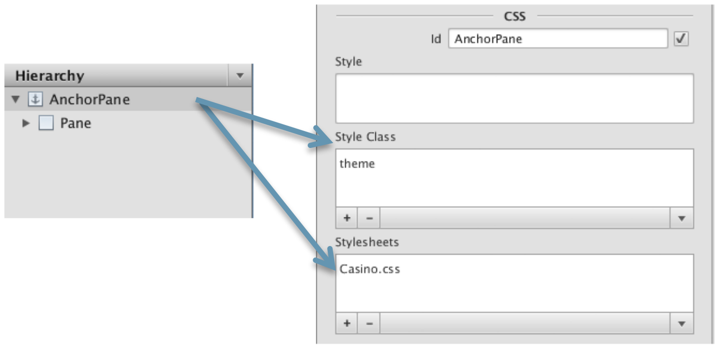
Blend Mode
Cache
Cache Hint DEFAULT
Depth Test INHERIT

	TOP	RIGHT	BOTTOM	LEFT
Insets	0	0	0	0

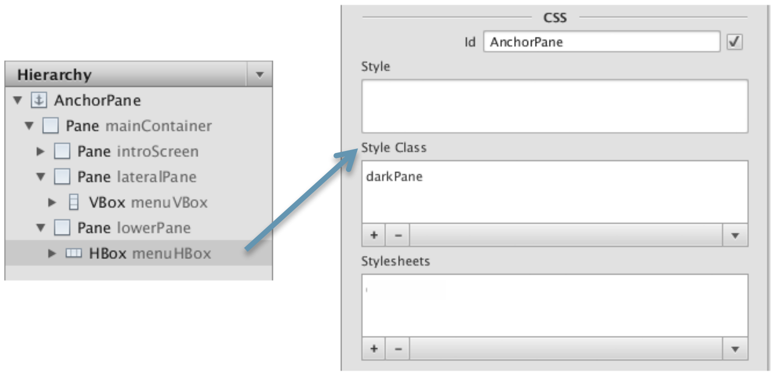
JavaFX with CSS

Setting the style in Scene Builder

Assign the stylesheet to the main container



Id: AnchorPane
Style Class: theme
Stylesheets: Casino.css



Id: AnchorPane
Style Class: darkPane

Set StyleClass for the component

|

Styling Directly in your Code

```
Button myStyldeButton = new Button(“Click Me”);
myStyledButton.setStyle(
          “-fx-background-color: blue;
          “-fx-border-color: yellow;)
```

Additions to HTML CSS

- Lookup
- Color Functions
  - derive
  - ladder
- Gradients
- Multiple background fills
- Multiple borders
- Effects
  - dropshadow
  - innershadow

Derive Function

```
derive( <color> , <number>% )
```

- Takes a color and computes a brighter or darker version of that color.
- Second parameter is the brightness offset
  - -100% to 100%.
  - Positive percentages indicate brighter colors.
    - 100% completely white.
  - Negative percentages indicate darker colors.
    - -100% completely black
    - 0% means no change in brightness

Ladder Function

```
ladder(<color> , <color-stop> [, <color-stop>]+)
```

- Interpolates between colors.
- The effect is as if a gradient is created using the stops provided, and then the brightness of the provided <color> is used to index a color value within that gradient.
  - 0% brightness, the color at the 0.0 end of the gradient is used
  - 100% brightness, the color at the 1.0 end of the gradient is used
  - 50% brightness, the color at the midway point of the gradient, is used.
  - No gradient is rendered -> single color result.

Gradient Function

- Linear gradients

```
linear-gradient( [ [from <point> to <point>] | [ to
<side-or-corner>], ]? [ [ repeat | reflect ], ]?
<color-stop>[, <color-stop>]+)
```

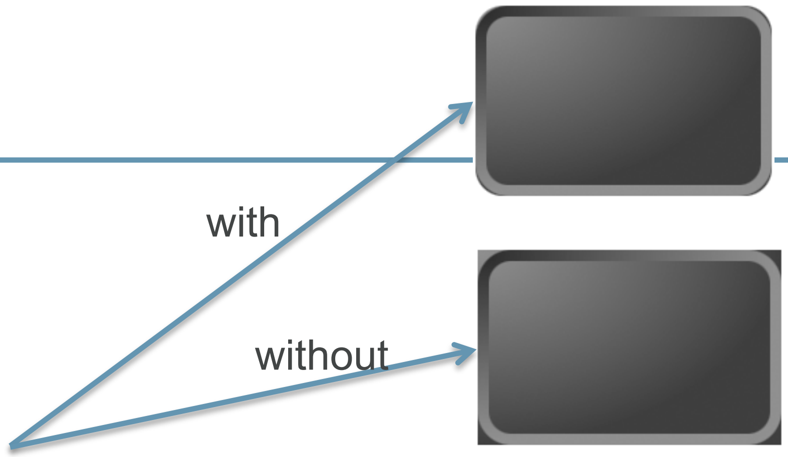
- Radial gradients

```
radial-gradient([ focus-angle <angle>, ]? [ focus-
distance <percentage>, ]? [ center <point>, ]? radius
[ <length> | <percentage> ] [ [ repeat |
reflect ], ]? <color-stop>[, <color-stop>]+)
```

JavaFX with CSS

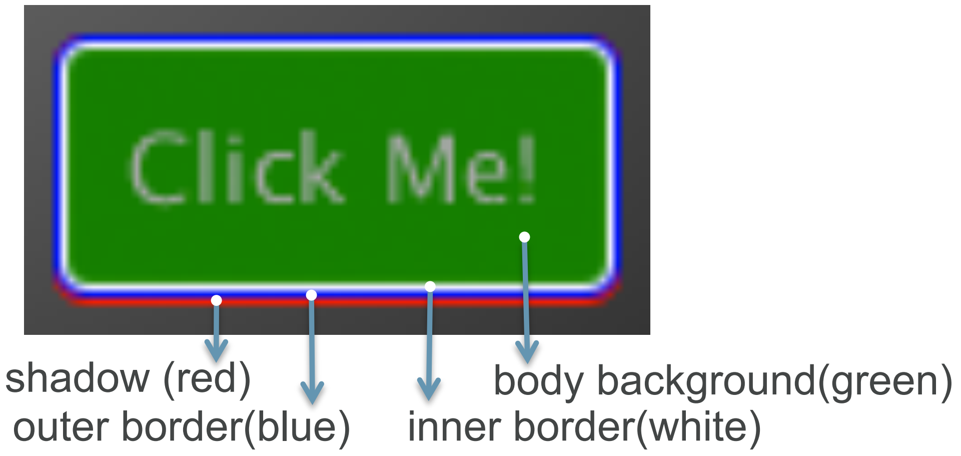
```
.root{
    master-color: black;
}

.my-gradientpane {
   -fx-background-radius: 20;
   -fx-border-radius: 20;
   -fx-border-width: 8;
   -fx-border-color: radial-gradient(radius 100%,
             derive(master-color,20%), derive(master-color,80%));
   -fx-background-color: radial-gradient(radius 100%,
             derive(master-color,80%), derive(master-color,30%));
}
```



JavaFX with CSS

```
.button {
    -fx-background-color: red, blue, white, green;
    -fx-text-fill: derive(master-color,110%);
}
```

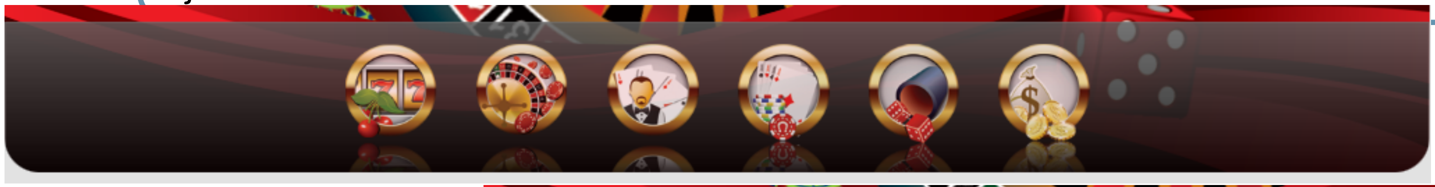


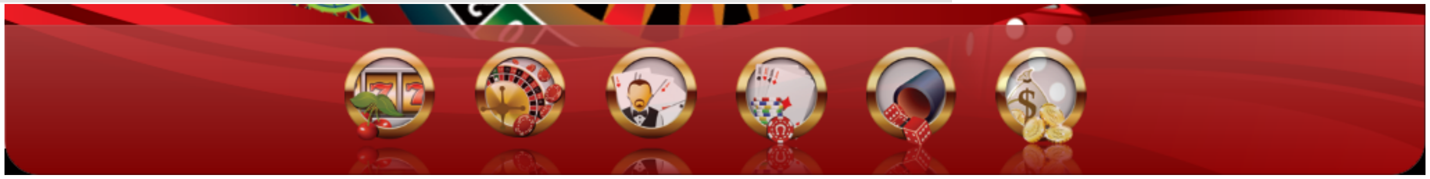
JavaFX with CSS

CSSFile.css

```
.theme{
    master-color: black;   //darkred
}

.darkPane {
    -fx-background-color:
            linear-gradient(to top, master-color,
                                  derive(master-color, 40%));
    -fx-border-color:
            derive(master-color, 70%),
            derive(master-color, 70%),
            transparent,
            derive(master-color, 70%);
}
```

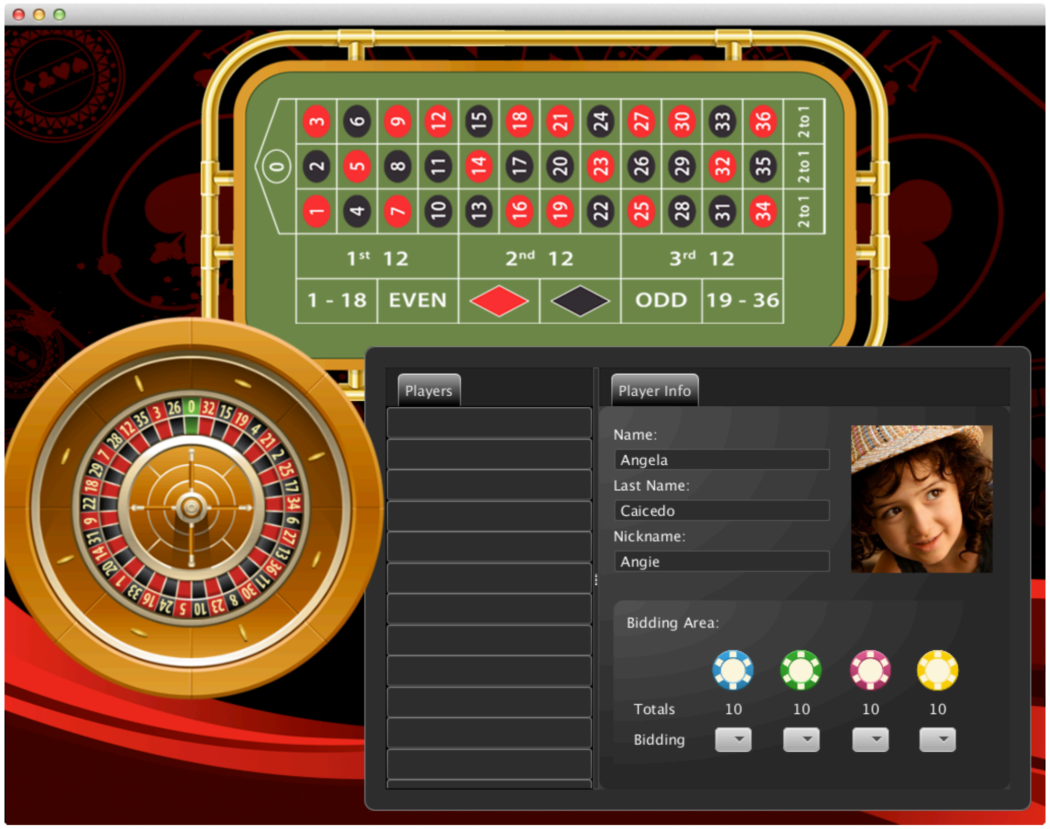




Red Panel

```
.myPane{
    -fx-effect: dropshadow(three-pass-box, rgba(0,0,0,0.4), 25, 0.0, 10,10);
    -fx-background-color:
        linear (0%, 0%) to (0%,100%) stops (0%, darkred) (100%, red),
        #f66883,
        linear (0%, 0%) to (0%,100%) stops (0%, red) (100%, darkred);
    -fx-background-insets: 0,1,2;
    -fx-background-radius: 15, 14, 13;
    -fx-padding: 20px;
}
```

Mastering CSS



Name: Angela

Last Name: Caicedo

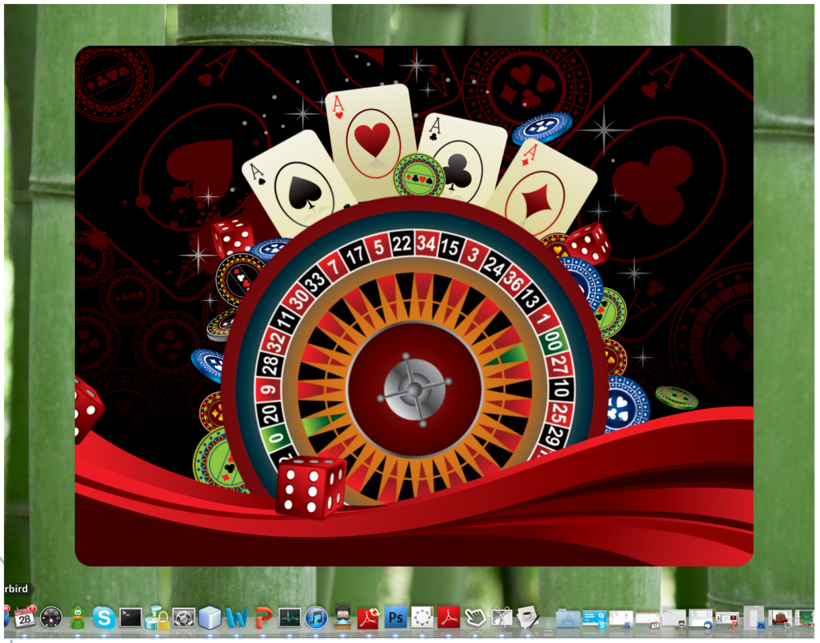
Nickname: Angie

Totals	10	10	10	10

JavaFX effects, animations and other cool features

JavaFX for Modern Interfaces

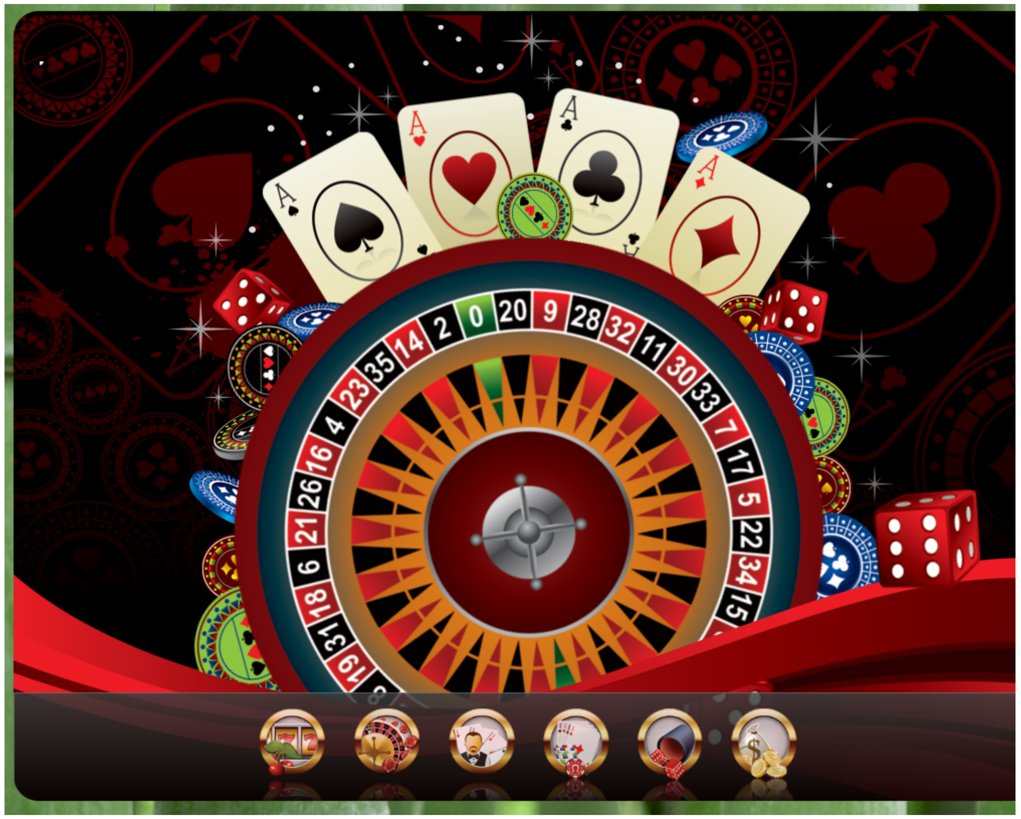
- Borderless applications
  - Blend nicely with the background
  - Set Scene's filling to null
  - Use StageStyle.TRANSPARENT



```
Scene scene = new Scene(root, 1024, 786);
scene.setFill(null);
stage.setScene(scene);
stage.initStyle(StageStyle.TRANSPARENT);
stage.show();
```

JavaFX for Modern Interfaces

- Dynamic slide-in menus
- Semi-transparent bar
- Customized buttons:
  - Reflection
  - Zoom in feature
  - Pushed effect

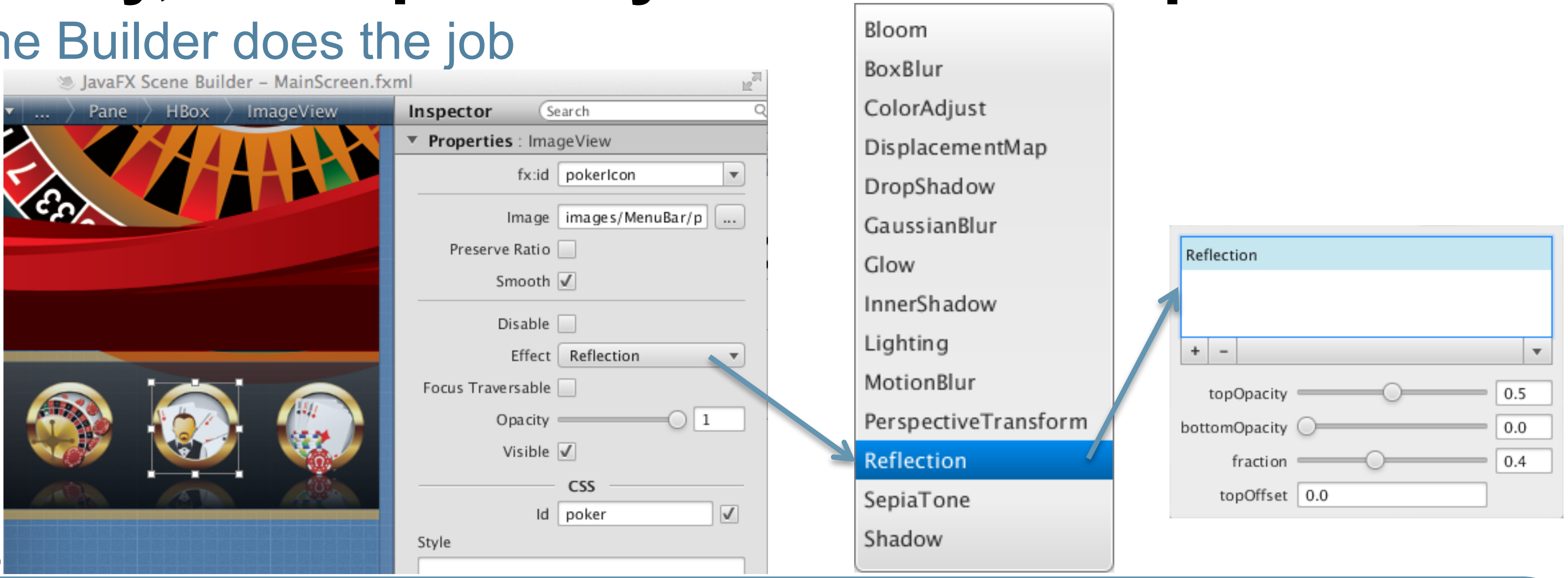


Component Transparency

```
Scene scene = new Scene(root, 1024, 786);
scene.getStylesheets().add("path/CSSFile.css");
...
Hbox menuH= HBoxBuilder.create().
                  alignment(Pos.TOP_CENTER).
                  opacity(0.85).
                  build();
menuH.getStyleClass().add(“darkPane");
```

Opacity, Transparency and other Properties

Scene Builder does the job



fx:id pokerIcon
Image images/MenuBar/p
Preserve Ratio ☐
Smooth ✓
Disable ☐
Effect Reflection
Focus Traversable ☐
Opacity 1
Visible ✓
Id poker ✓
Style

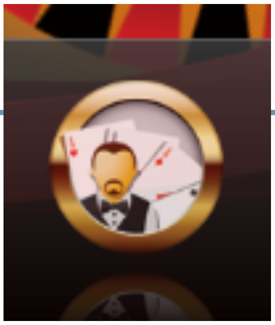
topOpacity 0.5
bottomOpacity 0.0
fraction 0.4
topOffset 0.0

FXML

```
<HBox id="MenuHBox" fx:id="menuH" alignment="TOP_CENTER" opacity="0.85"  ...>
    <children>
      <ImageView id="roulette" fx:id="rouletteIcon" scaleX="0.7" scaleY="0.7">
         <effect>
           <Reflection fraction="0.4" topOpacity="0.5"/>
         </effect>
          ...
```

Opacity, Transparency and other Properties

Programmatically

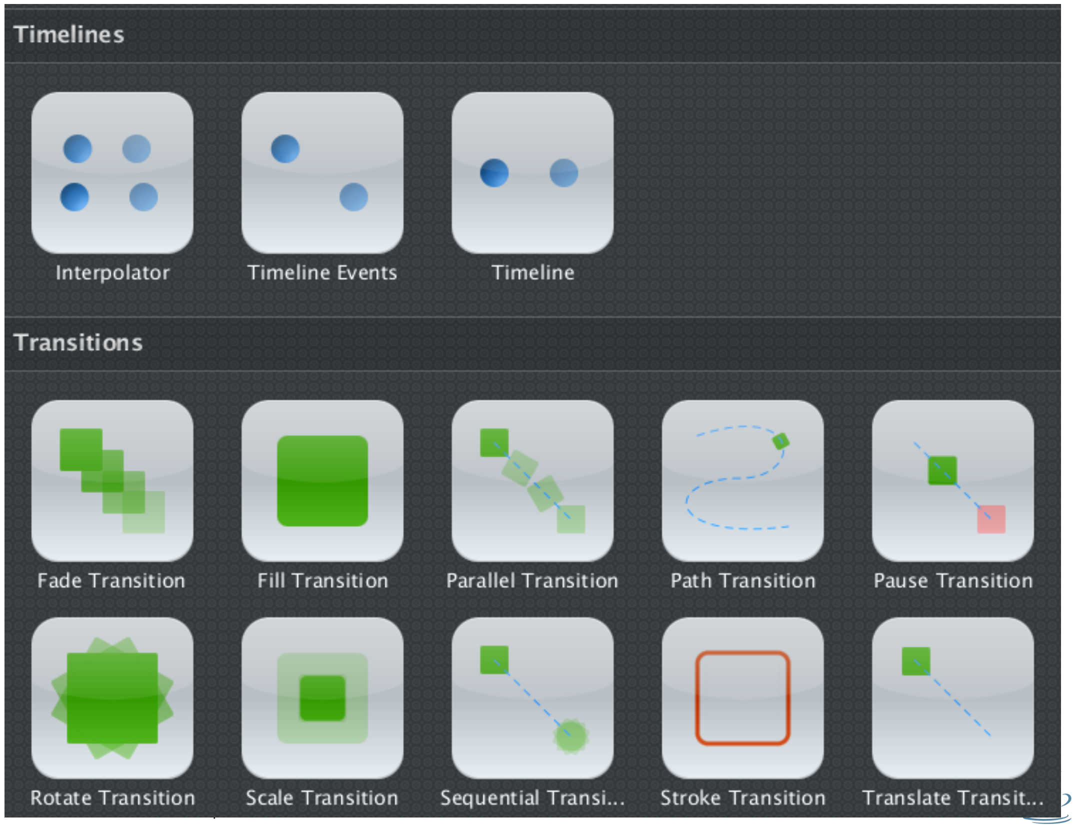


```
HBox menuH = HBoxBuilder.create().
                alignment(Pos.TOP_CENTER).
                opacity(0.85).
                build();
ImageView rouletteIcon  = new ImageView(new Image(Borrar.class.
                                 getResourceAsStream("roulette.png")));
rouletteIcon.setScaleX(0.7);
rouletteIcon.setScaleY(0.7);
Reflection reflection = new Reflection();
reflection.setFraction(0.4);
reflection.setTopOpacity(0.5);

rouletteIcon.setEffect(reflection);
```

Animations in JavaFX

Timelines and Transitions



Timeline-Based Animation

- Timeline
  - Modifies values of variables specified by KeyFrames
  - Doesn’t necessarily do any animation itself
- KeyFrame: specifies that a variable should have...
  - A particular value
  - At a particular time
- KeyValue: key value to be interpolated for a particular interval
- How is animation actually done?
  - Arrange for a KeyFrame to modify an interesting Node variable
    - x, rotate, opacity, fill, ...

Timeline Animation Example

```
Circle circle = new Circle(25,25, 20, Color.web("1c89f4"));
circle.setEffect(new Lighting());
DoubleProperty xPos = circle.translateXProperty();

//create a timeline for moving the circle
Timeline moveTimeline = new Timeline(
        new KeyFrame(Duration.ZERO,
            new KeyValue(xPos, 0.0)),
        new KeyFrame(new Duration(4000),
            new KeyValue(xPos, 205.0)));
moveTimeline.setCycleCount(Timeline.INDEFINITE);
moveTimeline.setAutoReverse(true);

moveTimeline.play();
        //timeline.pause();
        //timeline.stop();
        //timeline.playFromStart();
```

Timeline Events Example

```
timeline = new Timeline();
timeline.setCycleCount(Timeline.INDEFINITE);
timeline.setAutoReverse(true);

KeyValue kValueX = KeyValue.keyValue(stack.scaleXModel(), 2);
KeyValue kValueY = KeyValue.keyValue(stack.scaleYModel(), 2);

Duration dur = Duration.valueOf(2000);
EventHandler<ActionEvent> onFinished = new EventHandler<ActionEvent>() {
  public void handle(ActionEvent t) {
    stack.setTranslateX(random()*200-100);
    i = 0;
  }
};

KeyFrame kFrame = new KeyFrame(dur, onFinished, kValueX, kValueY);
timeline.getKeyFrames().add(kFrame);
getChildren().add(stack);
timeline.play();
```

Animated Transitions

- Predefined, single-purpose animations
  - Fade, Path, Pause, Rotate, Scale, Translate
  - Can specify to, from, and by values
- Container transitions
  - Parallel, Sequential
  - Can be nested arbitrarily
- Transitions and Timelines have a similar ancestry
  - A timeline can be added to a Parallel / Sequential transition

Animated Transitions Example

Translate/Rotate/Scale/Fade...

```
Circle circle = new Circle(20, Color.CRIMSON);
getChildren().add(circle);
TranslateTransition translateTransition = TranslateTransitionBuilder.
        create().
        duration(new Duration(4000)).
        fromX(20).
        toX(380).
        cycleCount(Timeline.INDEFINITE).
        autoReverse(true).
        build();
translateTransition.play();
```

Animated Transitions Example

Parallel/Sequential...

```
parallelTransition = new ParallelTransition();
parallelTransition.getChildren().addAll(
      fadeTransition,
      translateTransition,
      rotateTransition,
      scaleTransition);
parallelTransition.setCycleCount(Timeline.INDEFINITE);
parallelTransition.play();
```

Dynamic slide-in menus

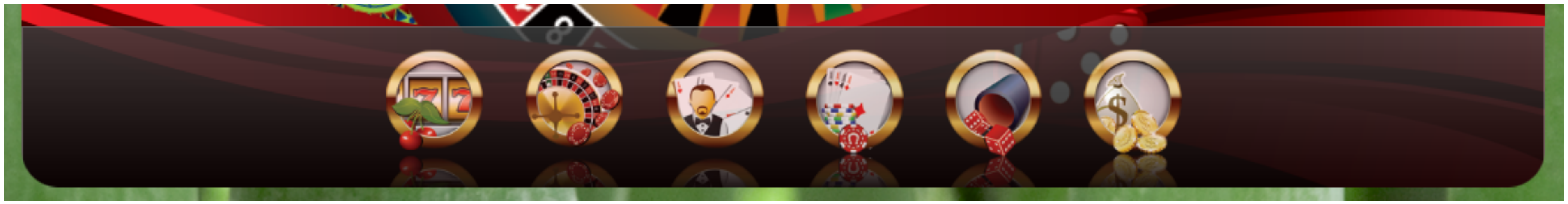
```
menuAnim = new TranslateTransition(Duration.millis(600),menuPane);
menuAnim.setFromY(MainScreenController.MENU_DOWN);
menuAnim.setToY(MainScreenController.MENU_UP);
```

Show

```
showMenuAnim.setRate(1);
showMenuAnim.play();
```

Hide

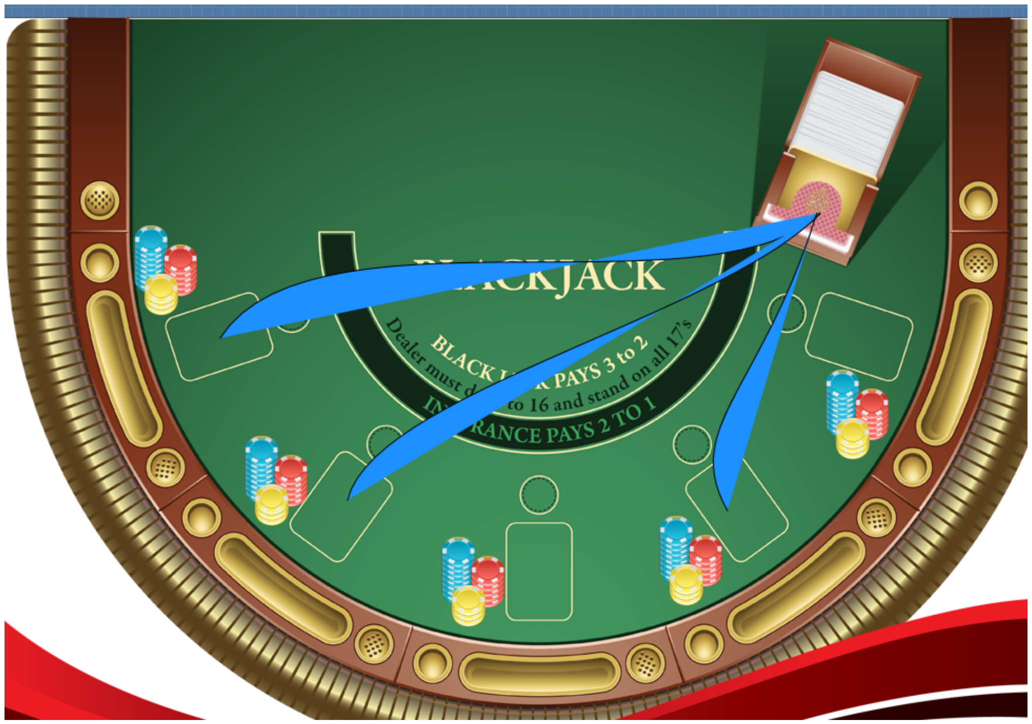
```
showMenuAnim.setRate(-1);
showMenuAnim.play();
```



Scene Builder

Amazing tool

- Visualize path for the cards animations



Lots and lots of Animations

- Serve player: parallel transition
  - Path transition:
    - Serve card + cover card
  - Rotate transition: position the card correctly
  - onFinished -> hide cover card
- Serve table: sequencial transition
  - Serve first card: player 1, player 2…. house.
  - Serve second card: player 1, player2… house

Lost of Animations

```
Path path = PathBuilder.create()
                .elements(new MoveTo(getDeckPosX(), getDeckPosY()),
                         new CubicCurveTo(coordX[0], coordY[0],
                           coordX[1], coordY[1],
                           coordX[2], coordY[2]))
                .build();
PathTransition pathTransitionCard = PathTransitionBuilder.create()
                .duration(Duration.seconds(Casino.SHORT_ANIM))
                .path(path)
                .node(card)
                .orientation(OrientationType.NONE)
                .build();
RotateTransition rotateCard = RotateTransitionBuilder.create()
                .toAngle(angle)
                .node(card)
                .duration(Duration.seconds(Casino.SHORT_ANIM))
                .build();
```

Lost of Animations

Option 1:

```
ParallelTransition playCard = new ParallelTransition(rotateCard, pathTransitionCard ,
                                                     rotateCover, pathTransitionCover);
playCard.setOnFinished(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent t) {
            FadeTransitionBuilder.create()
                  .duration(Duration.seconds(Casino.HALF_ANIM_TIME))
                  .toValue(Casino.NOT_VISIBLE)
                  .node(cover)
                  .build()
                  .play();
        }
    });
playCard.play();
```

Lost of Animations

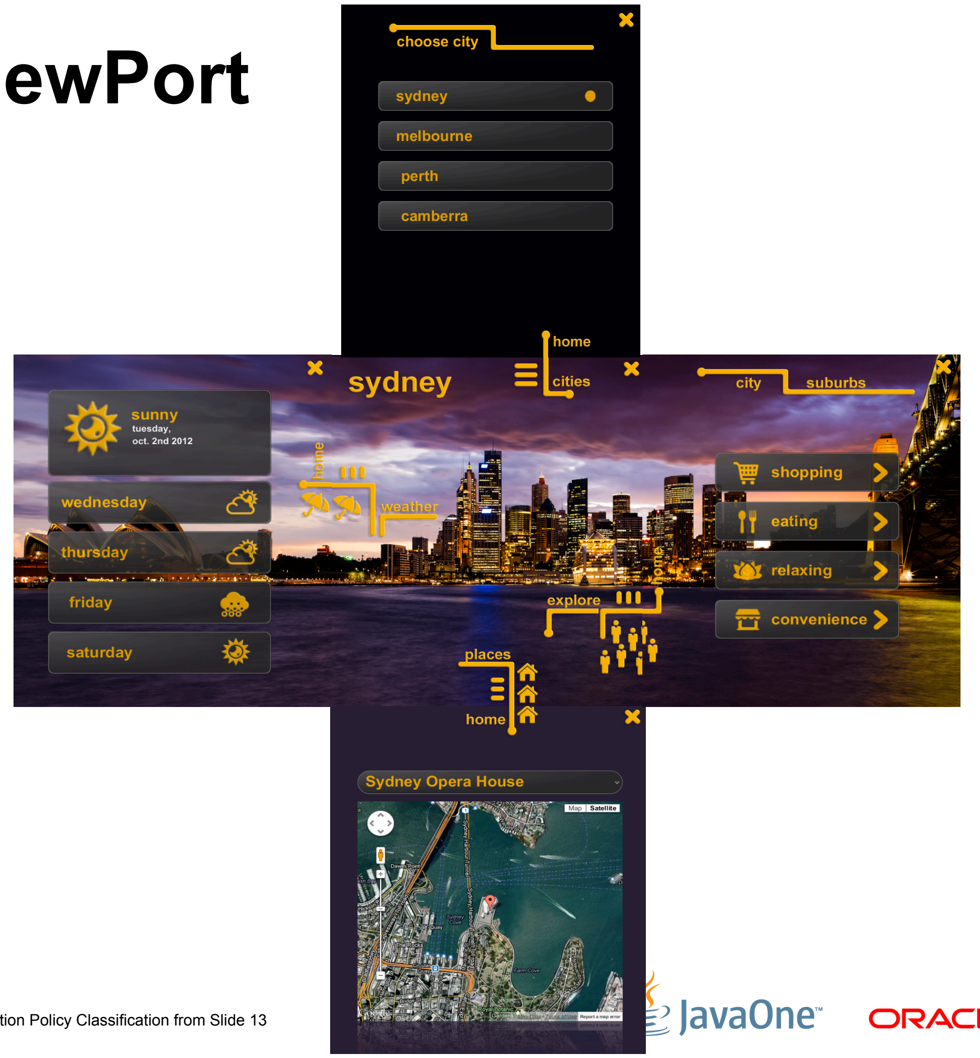
Option 2:

```
ParallelTransition playCard = new ParallelTransition(rotateCard, pathTransitionCard ,
                                                      rotateCover, pathTransitionCover);
FadeTransition hideCover = FadeTransitionBuilder.create()
                      .duration(Duration.seconds(Casino.HALF_ANIM_TIME))
                      .toValue(Casino.NOT_VISIBLE)
                      .node(cover)
                      .build()
                      .play();
            }
      });
SequentialTransition serveCard = new SequentialTransition(playCard, hideCover);
serveCard();
```

Two approaches, two applications, you choose

Working With a ViewPort

- Binding comes handy
- Good for small apps
- Only if scenegraph is flat

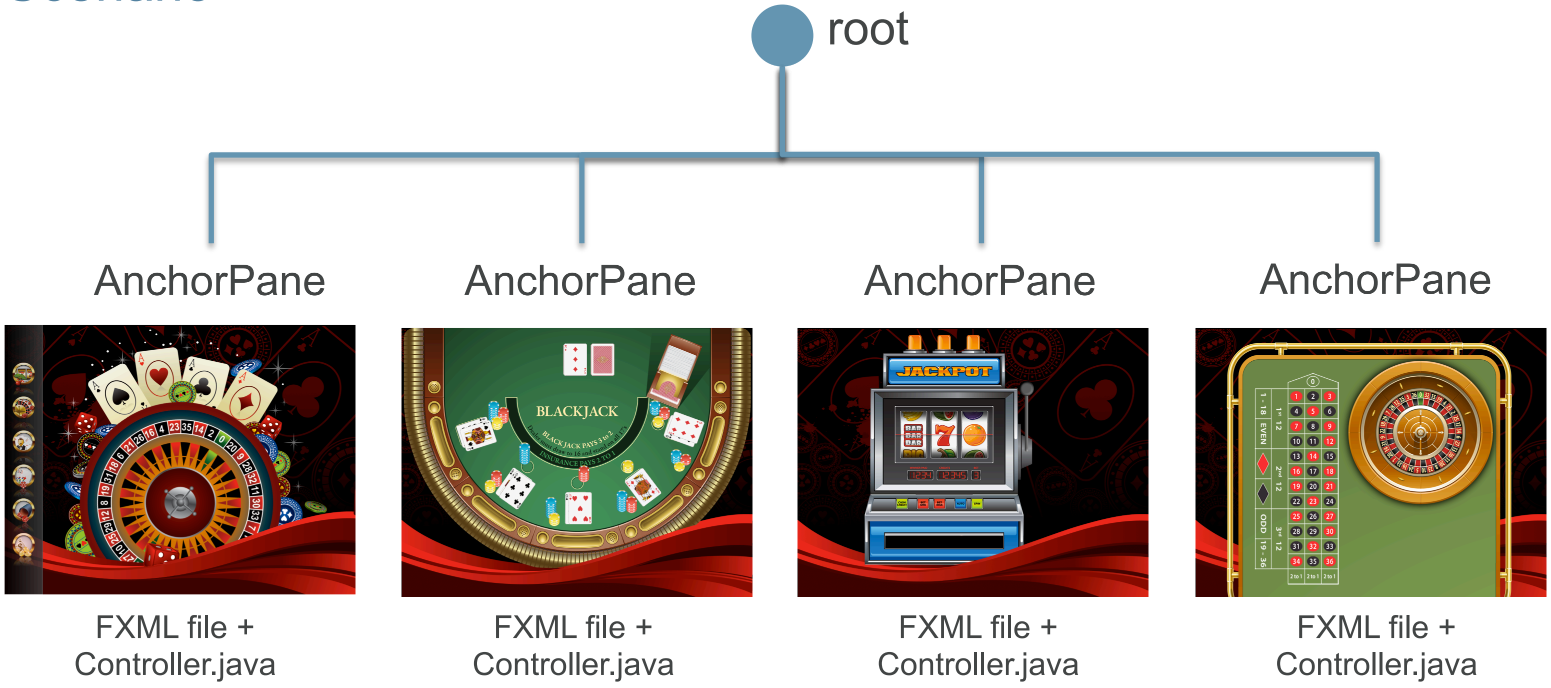


Multiple Screen Management

- Scenegraph should be as flat as possible
- SceneBuilder generate one file per screen, how to manage multiple screens?
- StackPane? Won’t solve the issue, scenegraph huge.
- Need to load and unload screens

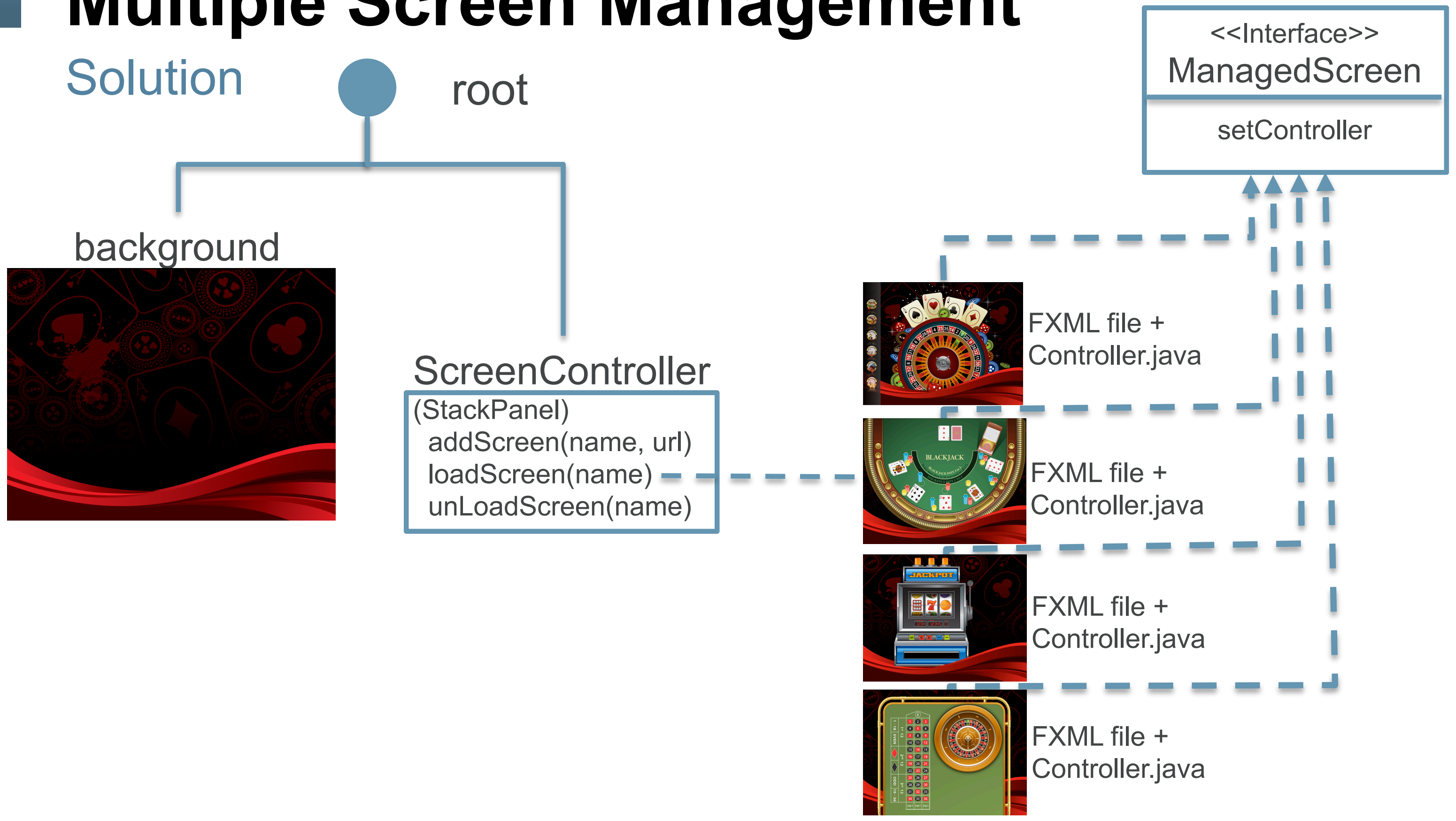
Multiple Screen Management

Scenario


root
AnchorPane
AnchorPane
AnchorPane
AnchorPane
BLACKJACK
JACKPOT
FXML file + Controller.java
FXML file + Controller.java
FXML file + Controller.java
FXML file + Controller.java

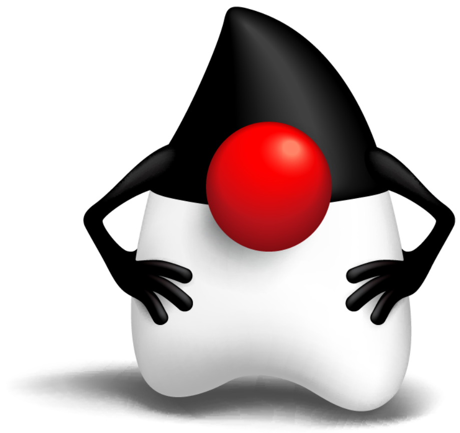
Multiple Screen Management

Solution



Tools and Demos

Demo



MAKE THE
FUTURE
JAVA

JavaOne™

ORACLE®

JavaOne™

- Dribbble.com
- Beautifulpixels.com
- Smashingmagazine.com
- Typography.com
- 960.gs

Textos con highlights

```
#login-dialog-label{
      -fx-text-fill: #717252;
      -fx-effect: dropshadow(one-pass-box, white, 0, 0.0, 0, 1);
}
```