Java2 Lab 11
(JavaFx3)

[Experimental Objective]

1. Understand and master the video, audio with JavaFx
2. Change the layout and style by writing a css file

[JavaFx music player]

Referring to the theoretical courseware, use javafx to write a music player that supports audio and video formats.

Use the Media to pass the video to the Url, you can use the local file or you can use the remote website, MP3, MP4 are supported.

Create a play button, use > to indicate, click on it to activate the play() behavior, and the button display content is changed to ||. If you click ||, pause() will be triggered to pause playback and display as >.

Create a replay button <<, click to trigger seek (Duration.ZERO), return to the starting position to replay.

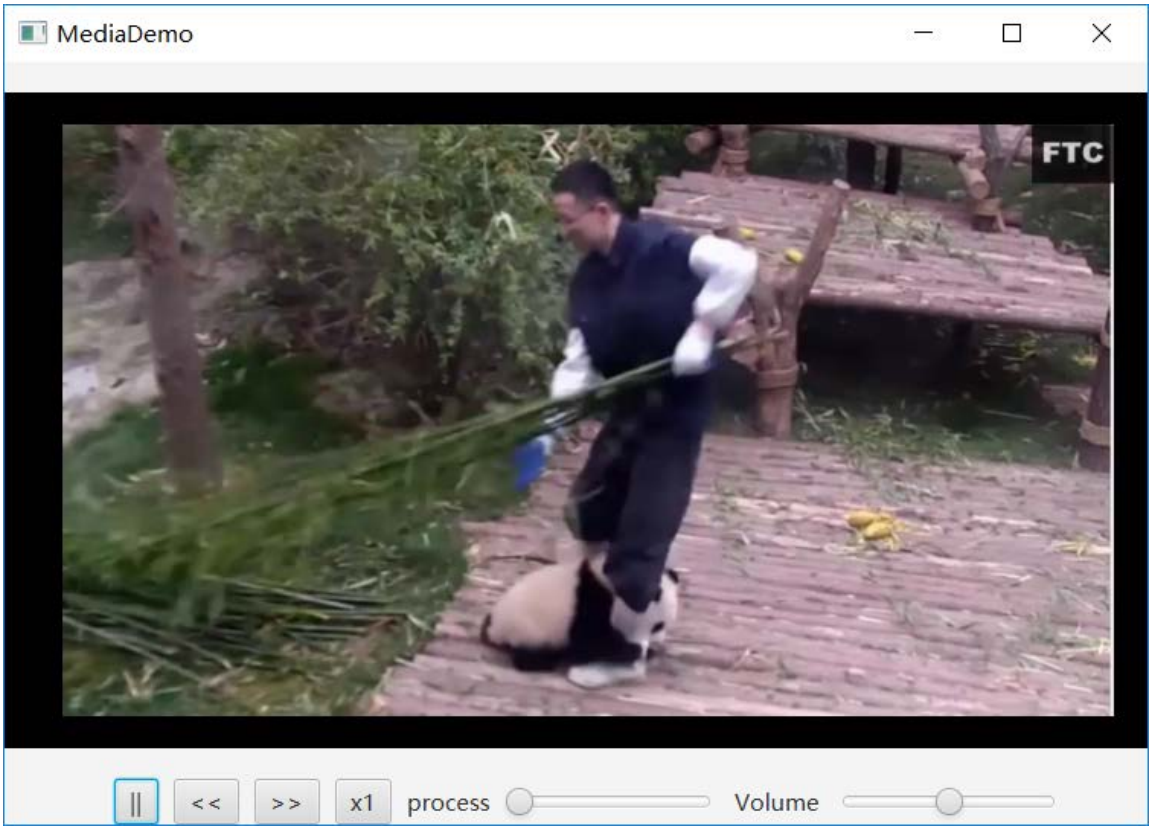
Finally, a volume sliding adjustment window is built to adjust the volume change according to the percentage of the sliding distance.

```
1.  public class testVideo1 extends Application {
2.
3.      private final String MEDIA_URL =
4.              "http://edu.konagora.com/video/TestVid.mp4";
5.
6.      @Override
7.      public void start(Stage primaryStage) {
8.
9.          Media media = new Media(MEDIA_URL);
10.         int width = media.widthProperty().intValue();
11.         int height = media.heightProperty().intValue();
12.         MediaPlayer mediaPlayer =
13.         new MediaPlayer(media);
14.         MediaView mediaView =
15.         new MediaView(mediaPlayer);
16.         Button playButton = new Button(">");
17.
18.         playButton.setOnAction(e -> {
19.             if (playButton.getText().equals(">")) {
20.             mediaPlayer.play();
21.             playButton.setText("||");
22.             } else {
23.             mediaPlayer.pause();
24.             playButton.setText(">");
25.             }
26.             });
```

```
27.             Button rewindButton = new Button("<<");
28.             rewindButton.setOnAction(e->
29.             mediaPlayer.seek(Duration.ZERO));
30.             Slider slVolume = new Slider();
31.
32.
33.
34.             slVolume.setPrefWidth(150);
35.             slVolume.setMaxWidth(Region.USE_PREF_SIZE);
36.             slVolume.setMinWidth(30);
37.             slVolume.setValue(50);
38.             mediaPlayer.volumeProperty()
39.             .bind(slVolume.valueProperty()
40.             .divide(100));
41.
42.             HBox hBox = new HBox(10);
43.             hBox.setAlignment(Pos.CENTER);
44.             hBox.getChildren().addAll(playButton,
45.             rewindButton,
46.             new Label("Volume"),
47.             slVolume);
48.
49.             BorderPane pane = new BorderPane();
50.
51.             pane.setCenter(mediaView);
52.             pane.setBottom(hBox);
53.             Scene scene = new Scene(pane, 750, 500);
54.             primaryStage.setTitle("MediaDemo");
55.             primaryStage.setScene(scene);
56.             primaryStage.show();
57.     }
58.
59.     public static void main(String[] args) {
60.         launch(args);
61.     }
62. }
```

【exercise 1】

Based on the above code, the following new features are added:
(1) Following the replay button <<, create a new quick end button>>, click to immediately jump to the end of the video and end playback.
(2) Imitate the play pause button >, ||, create a double speed change button, click to support x1, x2, x4, x6, x8 double speed playback, and display the current playback speed in the button.
(3) Follow the volume adjustment button to create a new progress bar adjustment button so that the progress bar can change the progress of the playback.



[JavaFx-Css]

For the sake of simplicity, we put the java file and the css file in the same directory, and bind the scene to the css file in java:

```
scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());
```

The default CSS style source in JavaFX 8 is a file called modena.css. The CSS file can be found in the JavaFX jar file jfxrt.jar, which is located in the Java directory. `/jdk1.8.x/jre/lib/ext/jfxrt.jar`。
Unzip jfxrt.jar and you should find modena.css in the com/sun/javafx/scene/control/skin/modena/ directory. The default style sheet

is always applied to JavaFX applications. By adding a custom style sheet, we can modify the style by overriding the default style in modena.css.

[css selector]

The selector locates the JavaFX nodes on the scene graph, and then we can style them using CSS style definitions.
The two types of selector types are id and class .
The id selector is a unique string name set on the scene node.
The class selector is a string name that can be added as a tag to any JavaFX node.
The class selector is independent of the concept of a Java class. Class selectors are used to group nodes to style them together with a CSS style definition.

[id selecctor]

id selector is a unique string name assigned to a node. When using the id selector, we will call the setId(String ID) method on the JavaFX node object to set its id.
For example, to locate a Button instance with an id of my-button, you will call the setId("my-button") method.
To set the button with the id my-button style, you will create a CSS style definition block selector#my-button declared with an id as follows:

```
#my-button {

    -fx-text-fill: rgba(17, 145,  213);

    -fx-border-color:  rgba(255, 255,  255,  .80);

    -fx-border-radius: 8;

    -fx-padding: 6  6  6  6;

}
```

This CSS style block will be applied to the button with the unique identifier of my-button.
The CSS selector name is preceded by a # symbol, and the id in the Java code does not use the # symbol.

类选择器

When using the type selector, we will call getStyleClass(). The add(String styleClass) method adds a selector to the node. This method allows us to have multiple style classes to style the nodes.

The getStyleClass() method returns an ObservableList, and we can add and remove style classes to dynamically update its appearance.

The following code locates a button with a style class containing a num-button via the getStyleClass().add("num-button") method.

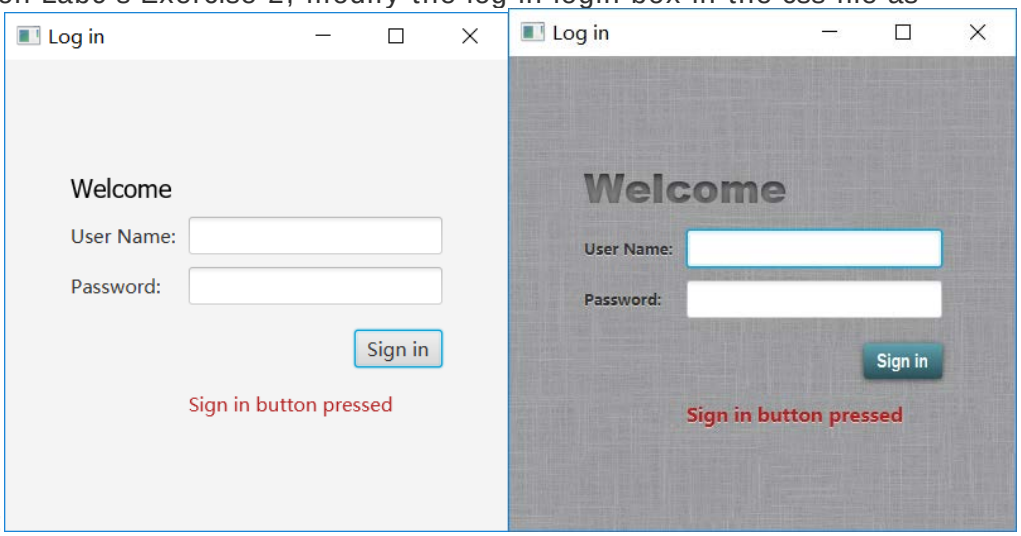
The following is a CSS style definition block declared with a class selector .num-button :

```
.num-button  {

    -fx-background-color: white, rgb(189,218,230), white;

    -fx-background-radius: 50%;

    -fx-background-insets: 0, 1, 2;

    -fx-font-family:  "Helvetica";
    -fx-text-fill:  black;
    -fx-font-size:  20px;
}
```

This CSS style block will be applied to buttons with the style class num-button. The CSS selector name is prefixed with a ., and the selector in the Java code is not used. symbol.

[Case 2]

Based on Lab9's Exercise 2, modify the log in login box in the css file as shown:



Requirements: The background requires the use of the following image (you can save the following image as jpg to the project). Modify the welcome font to Arial Narrow bold, modify the user name and other label gray bold to remove the shadow, click the user name and password text box when the blue bold border appears, modify the sign in button style and hover effect, appear when the button is clicked Action hint font bold

prompt:

Load the background image:

```
.root {
    -fx-background-image: url("background.jpg");
}
```

Modify the label size:

```
.label {
    -fx-font-size: 12px;
    -fx-font-weight: bold;
    -fx-text-fill: #333333;
    -fx-effect: dropshadow( gaussian , rgba(255,255,255,0.5) , 0,0,0,1 );
}
```

Button style settings:

```
.button {
    -fx-text-fill: white;
    -fx-font-family: "Arial Narrow";
    -fx-font-weight: bold;
    -fx-background-color: linear-gradient(#61a2b1, #2A5058);
    -fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0.6) , 5, 0.0 , 0 , 1 );
}
```

Button hover settings:

```
.button:hover {
    -fx-background-color: linear-gradient(#2A5058, #61a2b1);
}
```

Set the text text:

(1) Set the id in java

```
text.setId("welcome-text");
```

(2) Set the text style in css

```
    #welcome-text {
    -fx-font-size: 32px;
    -fx-font-family:"Arial Black";
    -fx-fill: #818181;
    -fx-effect: innershadow( three-pass-box , rgba(0,0,0,0.7) , 6,0.0,0,2 );
}
```

Action prompts are bold:

(1)Set the id in java:

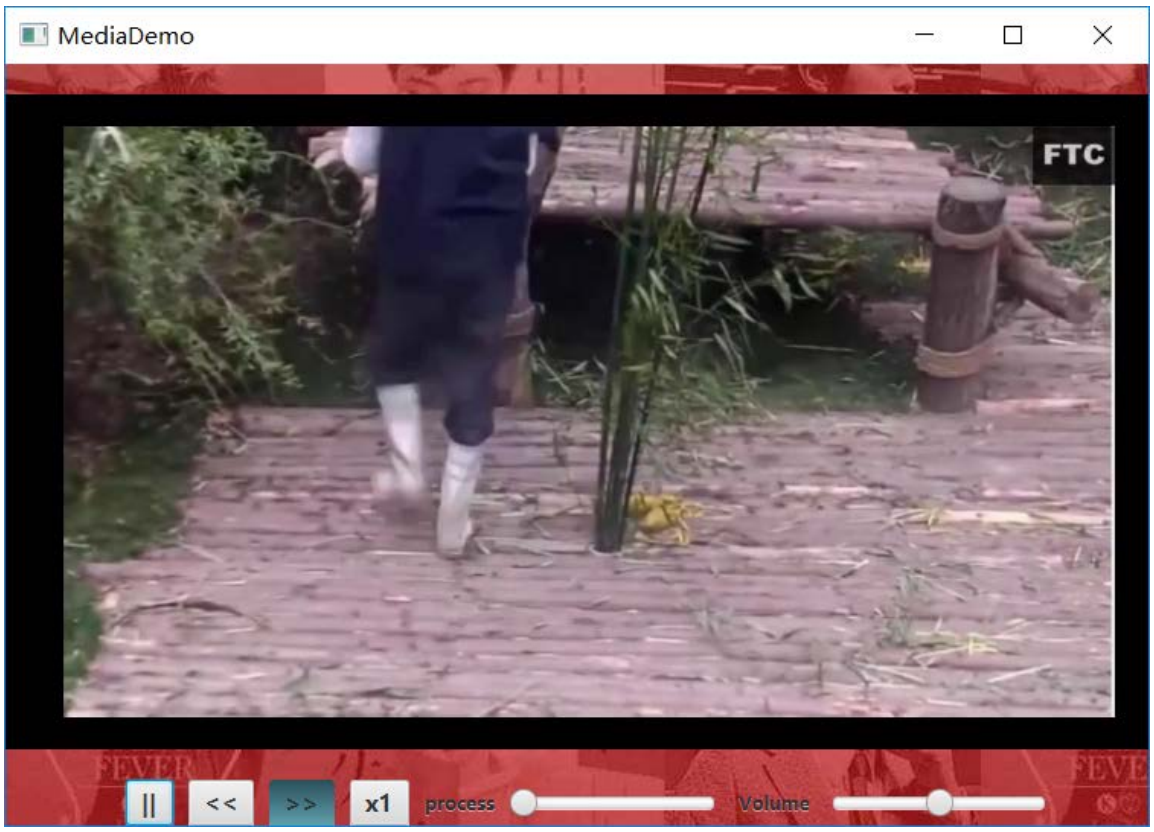
```
actiontarget.setId("actiontarget");
```

(2)implement action style in css

```
    #actiontarget {
     -fx-fill: FIREBRICK;
    -fx-font-weight: bold;
     -fx-effect: dropshadow( gaussian , rgba(255,255,255,0.5) , 0,0,0,1 );
}
```

【Exercise 2】

On the basis of Exercise 1, modify the player style with css, requiring:
(1) Modify the player background image. You can use the following to choose your own.
(2) Modify all buttons to display the fonts are bold, the label is shaded bold
(3) Button hover will change
color



11849062@mail.sustc.edu.cn