Objective

Introduction to JavaFX Animation

- Rotate Transition
- Translate Transition
- Scale Transition
- Sequential Transition
- Parallel Transition
- Animation Timer

JavaFX Animation

Animation in computer is displaying consecutive frames in which an object is transformed a little comparing to its previous frame. This creates an illusion of animation.

In JavaFX, the package javafx.animation contains all the classes for animation. All the classes of this package extend the class javafx.animation.Animation.

JavaFX provides many classes such as RotateTransition, ScaleTransition, TranslateTransition, AimationTimer, etc.

https://www.javatpoint.com/javafx-animation

RotateTransition

Lets start with a simple example:

Refer to RotationFX.java

1. Create a shape for animation

```
Rectangle rect = new Rectangle(100,100,50,50);
 rect.setFill(Color.BLUE);
```

2. Create an instance of the transition class

```
RotateTransition rotate = new RotateTransition();
```

3. Set the desired properties like duration

```
rotate.setDuration(Duration.millis(2000)); // 2 seconds
```

4. set related parameters

```
rotate.setByAngle(270);  // rotate angle in degree
```

RotateTransition

5. set Cycles and autoreverse parameters

```
rotate.setCycleCount(4);   // number of the cycles of the transition
rotate.setAutoReverse(true);   // autoreverse the transition
```

6. Add the shape to a group

```
Group group = new Group(rect);
```

7. Set the target node on which the transition will be applied.

```
rotate.setNode(group);
```

8. Add the group to the scene

```
Scene scene = new Scene(group, 600, 600);
```

9. Finally, play the transition using the play() method.

```
rotate.play();
```

TranslateTransition

Example:

Refer to TranslateFX.java

1. Create a shape for animation

```
Rectangle rect = new Rectangle(100,100,50,50);
rect.setFill(Color.BLUE);
```

2. Create an instance of the transition class

```
TranslateTransition translate = new TranslateTransition();
```

3. Set the desired properties like duration

```
translate.setDuration(Duration.millis(1500)); // 1.5 seconds
```

4. set related parameters

```
translate.setByX(300);
translate.setByY(100)
```

TranslateTransition

5. set Cycles and autoreverse parameters

```
translate.setCycleCount(4);   // number of the cycles of the transition
translate.setAutoReverse(true);   // autoreverse the transition
```

6. Add the shape to a group

```
Group group = new Group(rect);
```

7. Set the target node on which the transition will be applied.

```
translate.setNode(group);
```

8. Add the group to the scene

```
Scene scene = new Scene(group, 600, 600);
```

9. Finally, play the transition using the play() method.

```
translate.play();
```

ScaleTransition

Refer to ScaleFX.java

Lets start with a simple example:

1. Create a shape for animation

```
Rectangle rect = new Rectangle(100,100,50,50);
rect.setFill(Color.BLUE);
```

2. Create an instance of the transition class

```
ScaleTransition scale = new ScaleTransition();
```

3. Set the desired properties like duration

```
scale.setDuration(Duration.millis(2000)); // 2 seconds
```

4. set related parameters

```
scale.setByX(3);
scale.setByY(1.5);
```

ScaleTransition

5. set Cycles and autoreverse parameters

```
    scale.setCycleCount(4);   // number of the cycles of the transition
    scale.setAutoReverse(true);   // autoreverse the transition
```

7. Add the shape to a group

```
     Group group = new Group(rect);
```

Set the target node on which the transition will be applied.

```
    scale.setNode(group);
```

8. Add the group to the scene

```
     Scene scene = new Scene(group, 600, 600);
```

9. Finally, play the transition using the play() method.

```
      scale.play();
```

Other Animation Classes

JavaFX provides many animation classes such as:

FadeTransition, FillTransition, PathTransion, ect,

All follows the same routine except the parameters.

The parameters for each type of transitions are different.

Sequential/Parallel Transition

SequentialTransition and ParellelTransition are two classes that puts everything together, one sequentially and the other in parallel.

Sequential

SequentialTransition : animation will perform sequentially, from left to right

Refer to SequentialAmimation.java

```
SequentialTransition seq = new SequentialTransition();
seq.getChildren().addAll(rotate, translate, scale);
seq.play();
Group group = new Group(rect);
Scene scene = new Scene(group, 600, 600);
```

Parallel

ParallelTransition : animations perform in parallel.

In the example provided, you will see inconsistency during animation. It is due to the different duration and cycles, modify them, and you will get a better result.

Refer to ParallelFX.java

```
ParallelTransition par = new ParallelTransition();
par.getChildren().addAll(rotate, translate, scale);
seq.play();
Group group = new Group(rect);
Scene scene = new Scene(group, 600, 600);
```

AnimationTimer Class

General Animation

For a general and free animation, we use the abstract **AnimationTimer** class.

- It contains two implemented methods start/stop that do just that
- It contains one abstract method that we must implement
    - **`handle(long now)`**

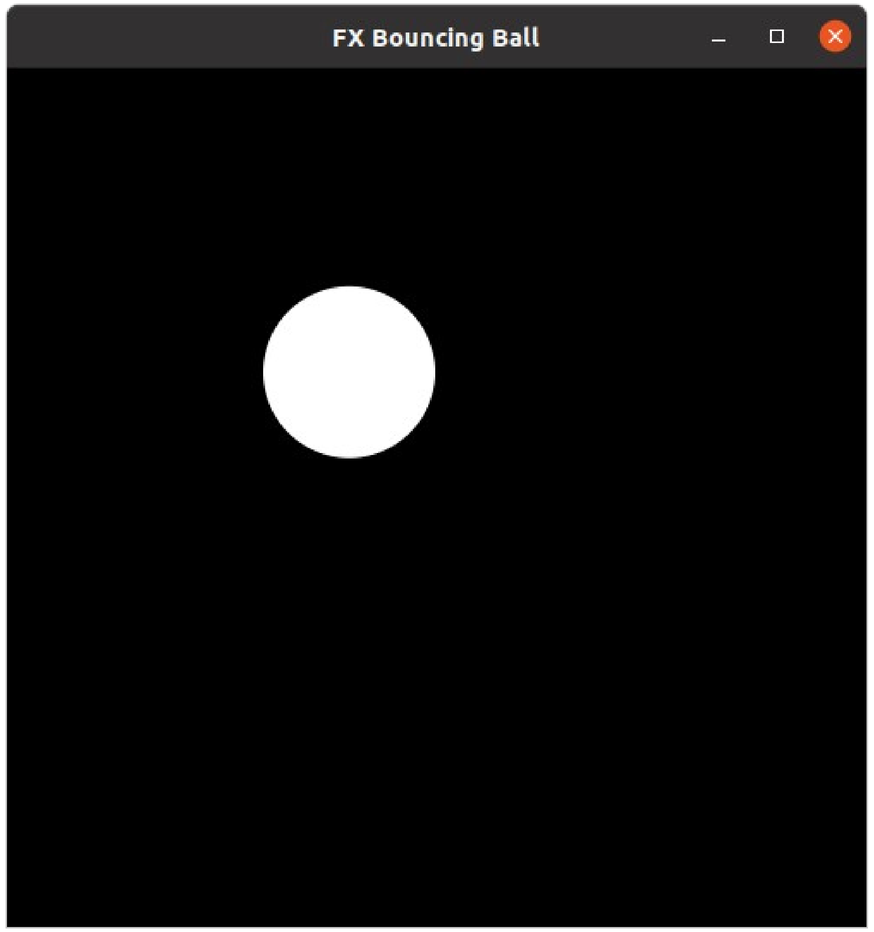
When the timer is running, handle will be called for each frame that needs to be drawn.

Bouncing Ball Animation

```
private BallAnimation animation;
private Ellipse ball;
private int xVelocity = 2;
private int yVelocity = 3;

@Override
public void start(Stage primaryStage) {
   Pane root = new Pane();

   Rectangle background = new Rectangle(0, 0, 500, 500);
   background.setFill(Color.BLACK);

   ball = new Ellipse(250,250,50,50);
   ball.setFill(Color.WHITE);

   animation = new BallAnimation();
   animation.start();

   root.getChildren().addAll(background, ball);
   Scene scene = new Scene(root, 500, 500);
   primaryStage.setTitle("FX Bouncing Ball");
   primaryStage.setScene(scene);
   primaryStage.show();
}
```

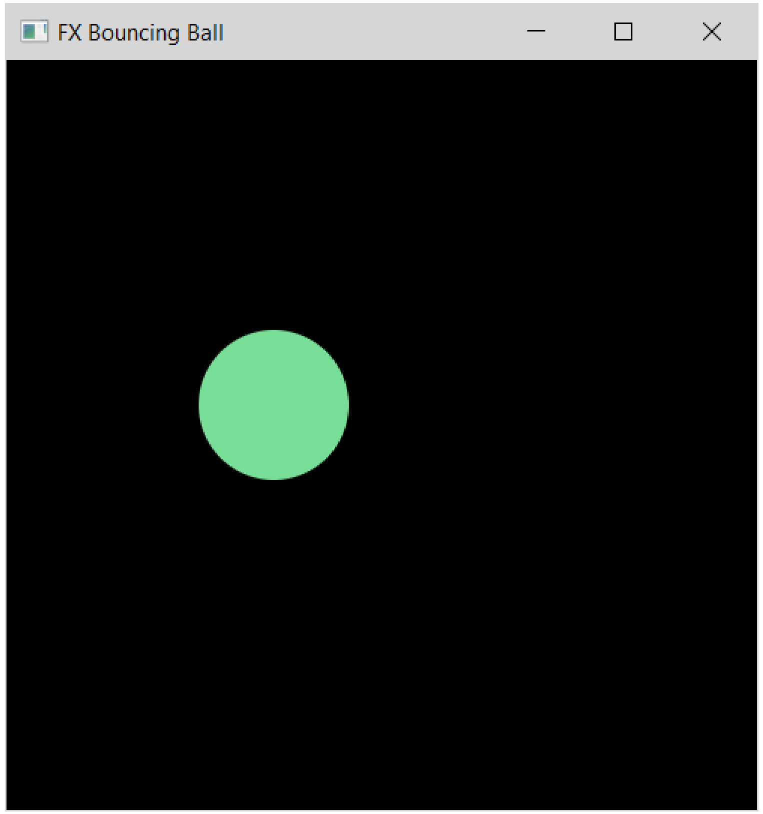
Bouncing Ball Animation

```
private class BallAnimation extends AnimationTimer {
  @Override
  public void handle(long now) {
    double x = ball.getCenterX();
    double y = ball.getCenterY();

    if (x + xVelocity > 450 || x + xVelocity < 50) {
      xVelocity *= -1;
    }

    if (y + yVelocity > 450 || y + yVelocity < 50) {
      yVelocity *= -1;
    }

    x += xVelocity;
    y += yVelocity;

    ball.setCenterX(x);
    ball.setCenterY(y);

  }
}
```



Bouncing Ball Animation

```
private int colorChangeCountdown = 60;

private class BallAnimation extends AnimationTimer {
  @Override
  public void handle(long arg0) {
    double x = ball.getCenterX();
    double y = ball.getCenterY();

    if (x + xVelocity > 450 || x + xVelocity < 50) {
      xVelocity *= -1;
    }

    if (y + yVelocity > 450 || y + yVelocity < 50) {
      yVelocity *= -1;
    }

    x += xVelocity;
    y += yVelocity;

    ball.setCenterX(x);
    ball.setCenterY(y);

    colorChangeCountdown--;
    if (colorChangeCountdown == 0) {
      ball.setFill(randomColor());
      colorChangeCountdown = 60;
    }

  }
}
```



Refer to BouncingBallFX.java