Java Programming

Arthur Hoskey, Ph.D.
Farmingdale State College
Computer Systems Department

Today's Lecture

- JavaFX Graphics
  - Animations

Animations in JavaFX

- Allows us to change property values over time.
- JavaFX has prebuilt classes that simplify animation.
- We will cover the following types of animations:
  - Transition animations
  - Timeline animations

Animations

Transition

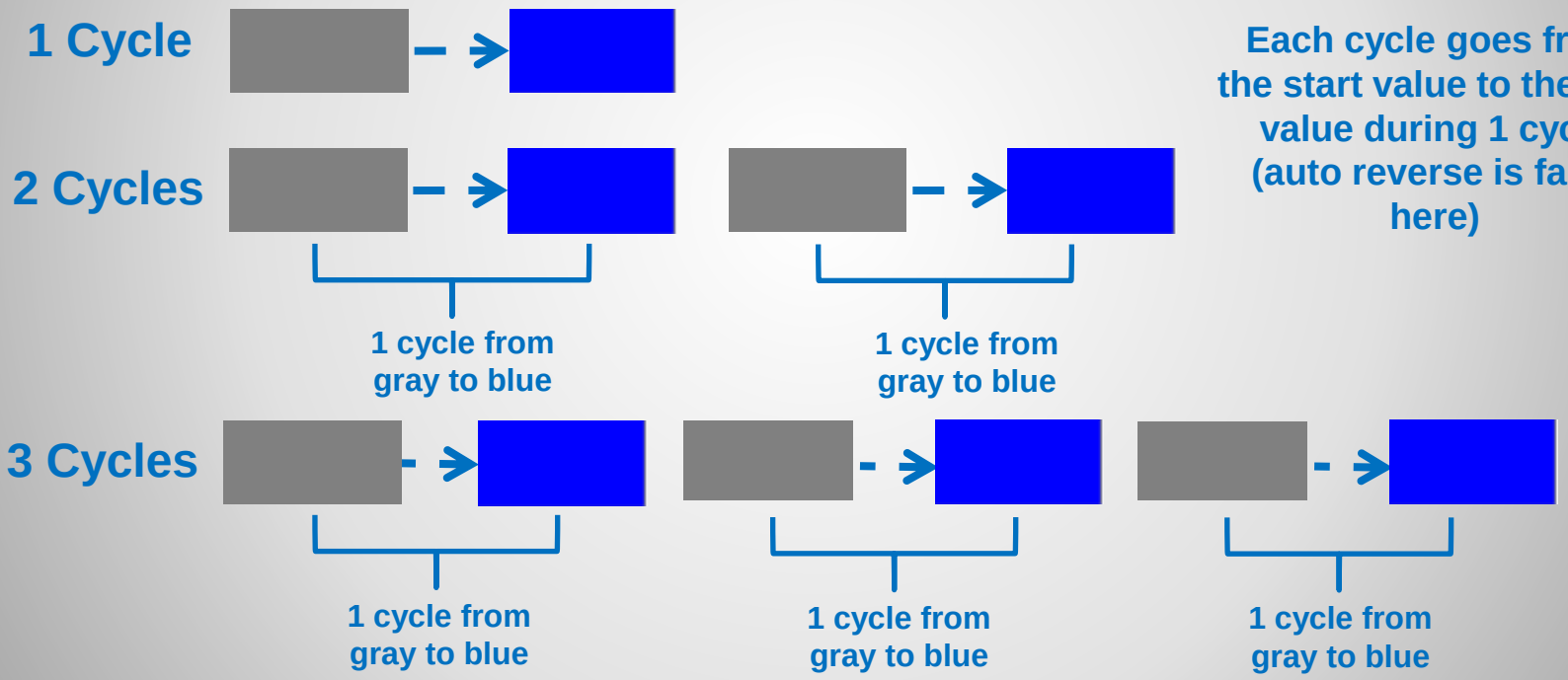
- Transition – Change a property value smoothly over a time period.
- The example below is a fill transition.
- It changes the color from gray to blue over a time period. (changes the fill color).

Color gradually changes from gray to blue over a defined period of time

Transition

Transition – Cycle

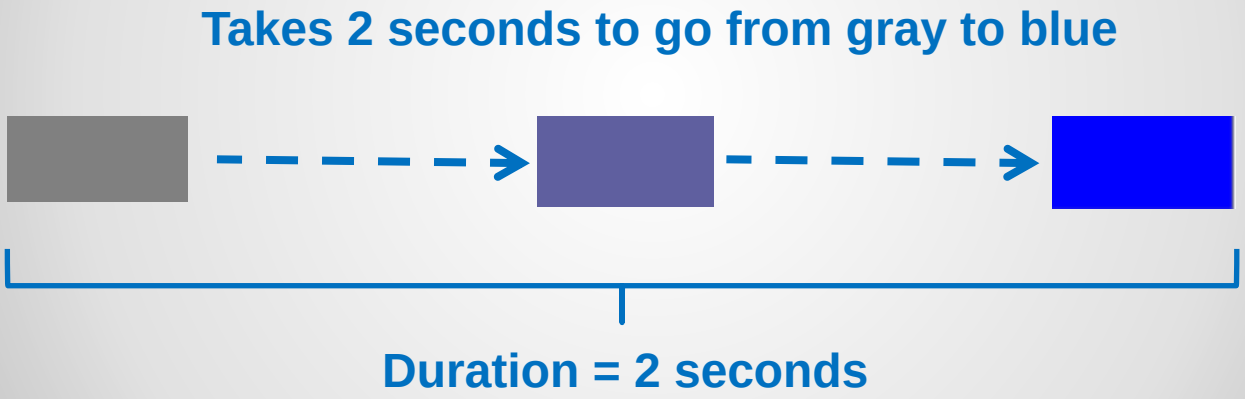
- One direction of the animation.
- Animations can have multiple cycles.
- Note: Auto reverse is false in this example (more on auto reverse on upcoming slides).



Transition - Cycle

Transition – Duration

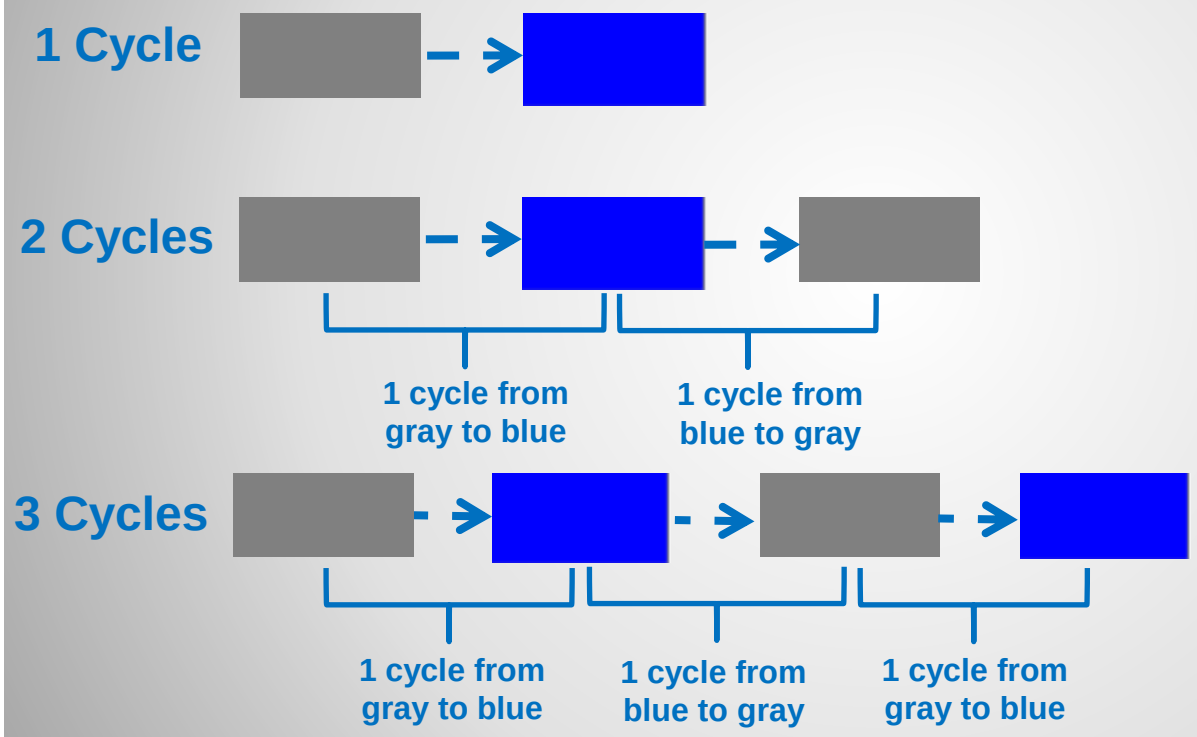
- The time it takes for one cycle of the animation.
- For example, if the duration is set to 2 seconds, then it will take 2 seconds to go from gray to blue.



Transition - Duration

Transition – Auto Reverse

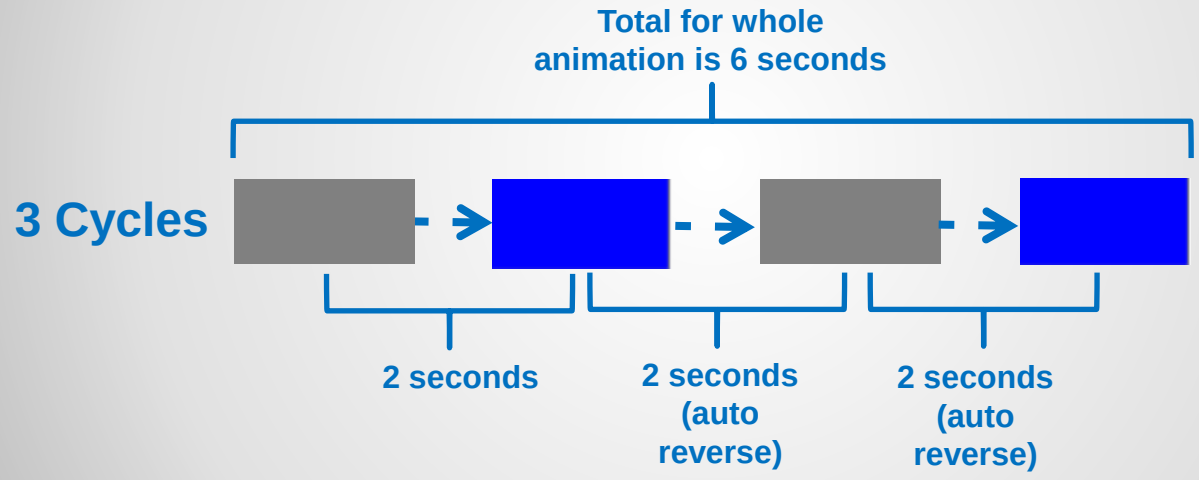
- Auto reverse will make the animation go in the opposite direction during the next cycle.



Transition – Auto Reverse

Transition – Example

- Duration = 2 seconds
- Cycles = 3
- Auto Reverse = true
- Total animation time will be 6 seconds (3*2=6).



Transition – Example

Transition Animations in JavaFX

- Prebuilt classes for specific property transition. For example:
  - FillTransition
  - StrokeTransition
  - FadeTransition
  - RotateTransition
  - TranslateTransition
  - PathTransition (moves object along a path)
  - ScaleTransition (changes object size)
- Classes to run multiple transitions
  - ParallelTransition (animations done at same time)
  - SequentialTransition (animation done one after another)

Transition Animations

FillTransition (StrokeTransition is similar)

- Changes the fill color to a new color over time.
- Note: Use StrokeTransition class to change the stroke.

```
@FXML
private Rectangle rectangle;

FillTransition fillTransition =
   new FillTransition(Duration.seconds(2), rectangle);

fillTransition.setToValue(Color.BLUE);

fillTransition.setCycleCount(4);

fillTransition.setAutoReverse(true);
fillTransition.play();
```

Duration of one transition cycle will be 2 seconds

Do the transition on the rectangle instance

Change color to BLUE

Number of times to do the cycle (4 times in this case)

The even cycles will transition back to the original color.
Cycle 1 goes to BLUE
Cycle 2 goes to original
Cycle 3 goes to BLUE
Cycle 4 goes to original

FillTransition

Fill Transition - Cycles

- A cycle goes in one direction.
- In this code the animation goes from red to blue once. It takes 2 seconds for it to happen.

```
// Assume starting color is red
FillTransition fillTransition = new FillTransition(Duration.seconds(2), rectangle);
fillTransition.setToValue(Color.BLUE);
fillTransition.setCycleCount(1);
```

- You can do more than one cycle. The following code does two cycles. It will go from red→blue over the course of 2 seconds and then go from red→blue again taking another 2 seconds. On the second cycle there is no transition back to red, it immediately goes to red and transitions to blue again. The total duration will end up being 4 seconds.

```
FillTransition fillTransition = new FillTransition(Duration.seconds(2), rectangle);
fillTransition.setToValue(Color.BLUE);
fillTransition.setCycleCount(2);
```

FillTransition - Cycles

Fill Transition - Auto Reverse

- Auto reverse causes the animation to go back to its starting value.
- The first cycle will go from red to blue taking 2 seconds. The second cycle will go from blue to red taking two seconds. The total duration will be 4 seconds and the fill color will be red when it is done (the original color).

```
// Assume starting color is red
FillTransition fillTransition = new FillTransition(Duration.seconds(2), rectangle);
fillTransition.setToValue(Color.BLUE);
fillTransition.setCycleCount(2);
fillTransition.setAutoReverse(true);
```

- Note: The cycle count needs to be greater than 1 to see the effect of auto reverse. For example, in the following code it does not go back to the original color because the cycle count is 1.

```
FillTransition fillTransition = new FillTransition(Duration.seconds(2), rectangle);
fillTransition.setToValue(Color.BLUE);
fillTransition.setCycleCount(1); // Cycle count too low for auto reverse
fillTransition.setAutoReverse(true);
```

Fill Transition - Auto Reverse

FadeTransition

- Changes the opacity of the shape (opaque vs transparent).
- Opaque - Cannot see through the object.
- Transparent – Can see through the object.
- For example:

```
FadeTransition fadeTransition =
  new FadeTransition(Duration.seconds(2), rectangle);

fadeTransition.setFromValue(1.0);
fadeTransition.setToValue(0.0);
fadeTransition.setCycleCount(2);
fadeTransition.setAutoReverse(true);
fadeTransition.play();
```

Starting as opaque (1.0 means you CANNOT see through object)

Ending as transparent (0.0 means you CAN see through object)

FadeTransition

- Now on to handling multiple transitions…

Handling Multiple Transitions

Handling Multiple Transitions

- JavaFX allows you can coordinate multiple transitions.
- ParallelTransition – Do multiple transitions simultaneously.
- SequentialTransition – Do one transition after another in sequence.

Handling Multiple Transitions

ParallelTransition

- Do multiple transitions simultaneously (in parallel).
- For example:

```
ParallelTransition parallelTransition = new
  ParallelTransition(fillTransition, fadeTransition);

parallelTransition.play();
```

Simultaneously run the fill and fade transitions (you can add more to the list if you want)

Run the parallel transition.

IMPORTANT!!! Make sure to NOT call play on the fill or fade transitions. The parallel transition will call it for both.

OR

```
ParallelTransition parallelTransition = new ParallelTransition();
parallelTransition.getChildren().add(fillTransition);
parallelTransition.getChildren().add(fadeTransition);

parallelTransition.play();
```

You can also add transitions after calling new

ParallelTransition

SequentialTransition

- Do multiple transitions one after another (in sequence).
- Each transition will run to completion and then the next will start.

Run the fill, fade, and stroke transitions in sequence (you can add more to the list if you want)

```
SequentialTransition sequentialTransition = new
SequentialTransition(fillTransition, fadeTransition, strokeTransition);
sequentialTransition.play();
```

Run the transitions one after another

IMPORTANT!!! Make sure to NOT call play on the fill or fade transitions. The parallel transition will call it for both.

SequentialTransition

Running a Method After an Animation Finishes

- Use the setOnFinished method.
- Here is an example using a ParallelTransition (setOnFinished works on any Transition type):

```
ParallelTransition parallelTransition = new ParallelTransition();
// other code to setup the transition here…

parallelTransition.setOnFinished(e -> someMethod());
parallelTransition.play();
```

- The someMethod() method will run after the animation finishes.
- Note: The -> is the lambda operator. If you do not understand lambda, all you need to do is replace someMethod with the name of whatever method you want to run.

Running a Method After an Animation Finishes

- Now on to TimeLine animations…

Timeline Animations

Timeline Animations in JavaFX

- Timeline animations can be used on any Node property that can be changed.
- Gives more control and has more options than transitions.

- Uses the following classes:
  - KeyValue – Defines a property and target value for that property.
  - KeyFrame – Holds a set of KeyValues. It sets a time when the KeyValues should reach their target values.
  - Timeline – Holds a set of KeyFrames. It will play the animation. It controls each of its KeyFrames reaching moving towards their target values in their respective times.

Timeline Animations

Timeline Animations

Use circle's translateX property

Target is 100

```
KeyValue keyValue1 = new KeyValue(circle.translateXProperty(), 100);
KeyFrame keyFrame1 = new KeyFrame(Duration.seconds(1), keyValue1);
```

Setup keyFrame1

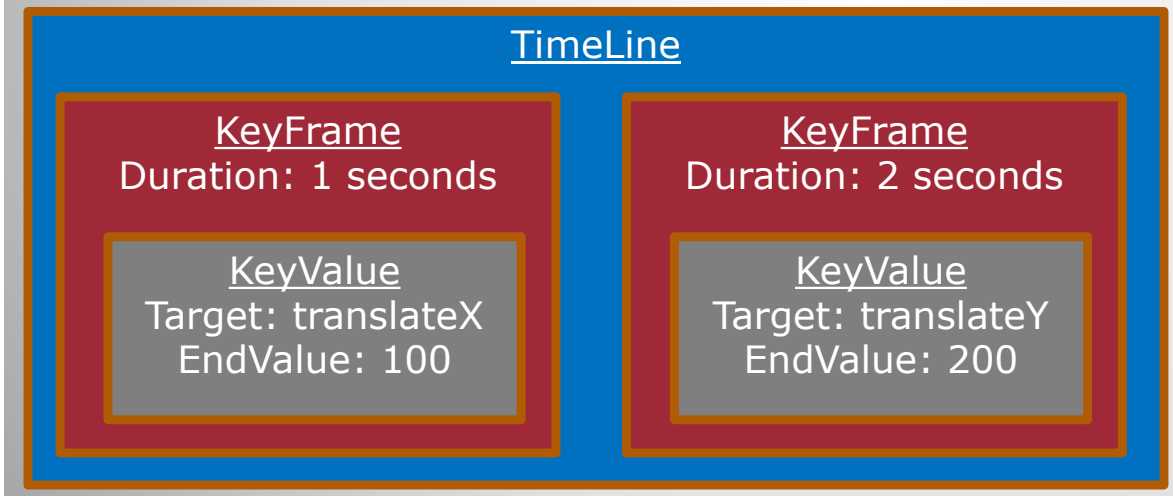
```
KeyValue keyValue2 = new KeyValue(circle.translateYProperty(), 200);
KeyFrame keyFrame2 = new KeyFrame(Duration.seconds(2), keyValue2);
```

Setup keyFrame2

```
Timeline timeline1 = new Timeline(keyFrame1, keyFrame2);
timeline1.play();
```

Add KeyFrames to Timeline

Run animation



Timeline will run all of its KeyFrames simultaneously. In this case keyFrame2 will take longer to finish than keyFrame1 (two seconds instead of one).

Timeline Animations

Sequential Timeline Animation

- You can run animations sequentially if you want.
- Use a SequentialTransition to play the animation (instead of calling play on the Timeline).

```
KeyValue keyValue1 = new KeyValue(circle.translateXProperty(), 100);
KeyFrame keyFrame1 = new KeyFrame(Duration.seconds(1), keyValue1);
Timeline timeline1 = new Timeline(keyFrame1);

KeyValue keyValue2 = new KeyValue(circle.translateYProperty(), 200);
KeyFrame keyFrame2 = new KeyFrame(Duration.seconds(2), keyValue2);
Timeline timeline2 = new Timeline(keyFrame2);

SequentialTransition sequentialTransition =
   new SequentialTransition(timeline1, timeline2);
sequentialTransition.play();
```

Put keyFrame1 in its own Timeline (do not call play on Timeline)

Put keyFrame2 in its own Timeline (do not call play on Timeline)

Add Timelines to a SequentialTransition

Runs animations. When timeline1 completes it will run timeline2

Sequential Timeline Animation

End of Slides