This Video

- The Canvas Node.
- JavaFX Animation.

Canvas Control

- See: http://docs.oracle.com/javafx/2/canvas/jfxpub-canvas.htm
- Canvas is a Node, is not visible by itself, but is not a Pane type Node so it cannot contain other nodes.
- Hint: Make sure the Canvas has a size and is where you think it is – otherwise you will not see anything!

Canvas Control, Cont.

- All drawing to the Canvas will take place through its GraphicsContext object:

```java
Canvas canvas = new Canvas(300, 250); // width, height
GraphicsContext gc = canvas.getGraphicsContext2D();
```

- (Also has an empty, default constructor.)
- Draw by setting colours first using:

```java
gc.setFill(Color.RED); // The fill colour
gc.setStroke(Color.BLUE); // The line colour
```

Canvas Control, Cont.

- Set the line thickness using:

```java
gc.setLineWidth(5); // 5 pixels
```

- See the GraphicsContext API docs for all the drawing methods. A method starting with “stroke…” draws the outline of the shape. Starting with “fill…” draws a filled shape.

Canvas Control, Cont.

- For example:

```java
to draw a line from (x1, y1) to (x2, y2) in pixels as doubles:
gc.strokeLine(x1, y1, x2, y2);
```
- Don’t forget that the (0, 0) pixel position of a Canvas is the top, left corner of the canvas.

Canvas Control, Cont.

- A filled oval:

```java
gc.fillOval(x, y, width, height);
```
- (x, y) is the upper left corner of a rectangle enclosing the oval of size width by height.
- (For a circle, width equals height…)

Prof. Alan McLeod
Canvas Control, Cont.

- You can draw many other shapes and curves including Bezier curves. Can also draw text.

- Effects such as linear and radial colour gradients can be applied by supplying LinearGradient and RadialGradient objects to the setStroke and setFill objects instead of just a single colour.

- The Canvas object itself can be translated, rotated and otherwise transformed in any way you can imagine.

Canvas Demo

- See FXCanvasDemo.

- Draws filled circles – pretty exciting!

- Check out the effect of colour transparency in addition to the RGB values.

- Ignore the animation aspect of the demo, for now.

JavaFX Animation

- Lots of videos and tutorials “out there”. A couple of links to use as a starting point:

  https://docs.oracle.com/javafx/2/animations/basics.htm
  and:

  http://www.java2s.com/Tutorials/Java/JavaFX/1010__JavaFX_Timelin e_Animation.htm

- Look at two different ways to animate: Transition and Timeline.
JavaFX Transition Animation

- Transitions can be used to animate single nodes or many nodes in parallel.

- A property (color, rotation, position, etc.) is changing from a start value to an end value over a specified length of time.

- Fairly easy to set up and useful for simpler node animations.

- See a code "snippet" on the next slide that moves a red rectangle 300 pixels in the x direction over 3 seconds, and back and forth, forever...

JavaFX Transition Animation, Cont.

```java
Rectangle rect = new Rectangle(10, 10, 100, 100);
rect.setFill(Color.RED);
TranslateTransition translate = new TranslateTransition(Duration.millis(3000), rect);
translate.setFromX(10);
translate.setToX(310);
translate.setAutoReverse(true);
translate.setCycleCount(Timeline.INDEFINITE);
translate.play();
```

JavaFX Transition Animation, Cont.

- See FXSimpleAnimation.

- Many other transitions are possible – fades, scaling, rotations, movement along a specific path, etc:
  - FadeTransition
  - ScaleTransition
  - RotateTransition
  - PathTransition
  - FillTransition
  - StrokeTransition
  - TranslateTransition
  - ParallelTransition (plays transitions at the same time)
  - SequentialTransition (plays transitions one after the other)

JavaFX Timeline Animation

- A `Timeline` object is constructed with a series of `KeyFrame` objects that are individual frame specifications along the animation.

- A `KeyFrame` is constructed using at least a Duration object and at least two `KeyValue` objects. Optionally, an `EventHandler` object can be used to execute some code when the `KeyFrame` is finished.

JavaFX Timeline Animation, Cont.

- A `KeyValue` object consists of a single state for a mutable node property value.

- You can also add an `Interpolator` object, which specifies how you interpolate between `KeyValues`.

- The default `Interpolator` is Interpolator.LINEAR.

- You can specify others or define your own, which is handy for specifying a specific path if your property is a position.
Imitating javax.swing.Timer

- This class creates its own thread and fires an event at specified intervals.
- JavaFX does not "like" this separate timer thread since it already has its own animation thread running.
- You can use a Timeline object to imitate a Timer object in JavaFX.
- Build the object using only two KeyFrame objects:

```java
Timeline timeline = new Timeline(
    new KeyFrame(Duration.ZERO, actionEvent -> drawWhatever()),
    new KeyFrame(Duration.millis(1000 / desiredFrameRate)));
```

- If desiredFrameRate is 60 then, this timeline object will invoke the drawWhatever() method every 17 milliseconds.

Aside – Maximum Frame Rate

- Common monitor refresh rates are 60 Hz, for example.
- There is no point in trying to update frames at any rate higher than this!
- So, the lowest time between frames would be:
  - 1000 / 60 or about 17 milliseconds.
- A gamer would be very happy if his game updated at 60 fps. What is a more realistic expectation? How slow can an animation get before it gets "choppy"?

Back to the Canvas Demo

- Look at a very simple implementation of a Timeline animation.
- The filled ovals ("blobs") are collected in an ArrayList and re-displayed with an animation, one per second.

Simple Animation Technique

- If you have a simple enough animation, you don’t need to do anything fancy:
  - Wipe or just redraw the blank canvas.
  - Figure out the current location of all element in your drawing.
  - Draw your figure at its current position.
  - Repeat the process when your timer says the required interval has passed.
- Your eye cannot respond fast enough to see the canvas being blanked and re-drawn. The animation should appear fairly smooth.