Graphic & Animation

Ekarat Rattagan

Android Graphics Programming

- There are many ways to do graphics programming in Android
  - 2D vs. 3D
  - static vs. dynamic
- Many of them require a lot of knowledge of the underlying graphics libraries
- We will look at the very simplest form of 2D graphics

Drawing on a Canvas

- Visible elements in an Android UI are called **Views**
- Each View has an associated **Canvas**
- When the View is shown, its ***onDraw*** method is automatically called by Android
- It uses the Canvas to render the different things it wants to display
- We can create our own View with our own *onDraw* method to display basic objects using the Canvas

Canvas and Paint

- **Canvas** has methods for drawing Arcs, Bitmaps, Circles, Lines, Ovals, Paths, Rectangles, etc.
- Also methods to rotate, scale, skew, translate
- **Paint** has methods for setting the alpha, color, shade, stroke, etc.

Let's Create a New Project!

- In Eclipse, go to File → New → Project
- Then select “Android Project”

- Name the project
- Specify the package
- Name the Activity class

- Next, create your own custom View class

Creating Your Own View Class

1. Create a new Java class that extends View

2. Implement the necessary constructors

3. Implement the *onDraw* method and use the Canvas parameter to draw using a Paint object

4. Add your View to the application's Layout

Create a DrawableView class

```
 1 package [your package];
 2
 3 public class DrawableView extends View {
 4
 5   // Second, you must implement these constructors!!
 6   public DrawableView(Context c) {
 7       super(c);
 8   }
 9   public DrawableView(Context c, AttributeSet a) {
10       super(c, a);
11   }
12

... continued on next slide ...
```

Still in the DrawableView class...

```
13   // Third, implement the onDraw method.
14   // This method is called when the View is displayed
15   protected void onDraw(Canvas canvas) {
16
17      // this is the “paintbrush”
18      Paint paint = new Paint();
19      // set the color
20      paint.setColor(Color.RED);
21
22      // draw Rectangle with corners at (40, 20) and (90, 80)
23      canvas.drawRect(40, 20, 90, 80, paint);
24
25      // change the color
26      paint.setColor(Color.BLUE);
27      // set a shadow
28      paint.setShadowLayer(10, 10, 10, Color.GREEN);
29
30      // create a “bounding rectangle”
31      RectF rect = new RectF(150, 150, 280, 280);
32      // draw an oval in the bounding rectangle
33      canvas.drawOval(rect, paint);
34   }
35
36 } // end of DrawableView class
```

Modify main.xml as follows

```
1 <?xml version="1.0" encoding="utf-8"?>
2
3  <[your package]
4   xmlns:android="http://schemas.android.com/apk/res/android"
5   android:layout_width="fill_parent"
6   android:layout_height="wrap_content"
7  />
8
```

Canvas object methods

- c.drawARGB(*alpha*, *r*, *g*, *b*); - fill window with color (rgb=0-255)
- c.drawArc(...); - draw a partial ellipse
- c.drawBitmap(*bmp*, *x*, *y*, null); - draw an image
- c.drawCircle(*centerX*, *centerY*, *r*, *paint*); - draw a circle
- c.drawLine(*x1*, *y1*, *x2*, *y2*, *paint*); - draw a line segment
- c.drawOval(*x1*, *y1*, *x2*, *y2*, *paint*); * *(requires Android 5.0)*
- c.drawOval(new RectF(*x1*, *y1*, *x2*, *y2*), *paint*); - draw oval/circle
- c.drawPoint(*x*, *y*, *paint*); - color a single pixel
- c.drawRect(*x1*, *y1*, *x2*, *y2*, *paint*); * *(requires Android 5.0)*
- c.drawRect(new RectF(*x1*, *y1*, *x2*, *y2*), *paint*); - draw rectangle
- c.drawRoundRect(*x1*, *y1*, *x2*, *y2*, *rx*, *ry*, *paint*); * *(requires Android 5.0)*
- c.drawRoundRect(new RectF(*x1*, *y1*, *x2*, *y2*), *rx*, *ry*, *paint*);
- c.drawText("*str*", *x*, *y*, *paint*); - draw a text string
- c.getWidth(), c.getHeight() - get dimensions of drawing are

Bitmap images

Draw an image (such as .png or .jpg) using the `Bitmap` class.

```
Bitmap name = BitmapFactory.decodeResource(
getResources(), R.drawable.ID);

// example: draw heart.png on screen at (0, 0)
Bitmap bmp = BitmapFactory.decodeResource(
getResources(), R.drawable.heart);
canvas.drawBitmap(bmp, 0, 0, null);

// you can also read a Bitmap from an input
stream
URL url = new
URL("http://example.com/myImage.jpg");
Bitmap bmp = BitmapFactory.decodeStream(
url.openStream());
```

Typeface

In Android, a font is called a Typeface. Set a font inside a Paint.
You can create a Typeface based on a specific font name:

`Typeface.create("font name", Typeface.STYLE)`

styles: NORMAL, BOLD, ITALIC, BOLD_ITALIC

- Or based on a general "font family":
  `Typeface.create(Typeface.FAMILY_NAME, Typeface.STYLE)`
  - family names: DEFAULT, MONOSPACE, SERIF, SANS_SERIF
- Or from a file in your `src/main/assets/` directory:
  `Typeface.createFromAsset(getAssets(), "filename")`

```
// example: use a 40-point monospaced blue font
Paint p = new Paint();
p.setTypeface(
 Typeface.create(Typeface.MONOSPACE, Typeface.BOLD));
p.setTextSize(40);
p.setARGB(255, 0, 0, 255);
```

Detecting User Interaction and Touch Events

Detecting Touch Events

- When the user touches/clicks on the View, Android invokes the View's ***onTouchEvent*** method
- A **MotionEvent** object is automatically generated and is passed to the method
- From the MotionEvent, you can determine:
  - the type of Action (down, up/release, move)
  - where the event occurred (x/y coordinate)
  - the time at which the event occurred

Modifying the DrawableView

1. In your DrawableView class, modify *onDraw* so that the color of the rectangle is randomized

2. Then add an *onTouchEvent* method that looks for an “up” action and calls *this.invalidate* if the touch is within the bounds of the rectangle

DrawableView class

```
1  package [your package];
2
3  public class DrawableView extends View {
4
5    // these constructors shouldn't change
6    public DrawableView(Context c) {
7        super(c);
8    }
9    public DrawableView(Context c, AttributeSet a) {
10       super(c, a);
11   }
12
```

Modify *onDraw* as follows

```
13   // This version of onDraw randomly chooses a color
14   // to use when drawing the rectangle
15   protected void onDraw(Canvas canvas) {
16
17      // this is the "paintbrush"
18      Paint paint = new Paint();
19
20      // set the color randomly
21      int whichColor = (int)(Math.random() * 4);
22      if (whichColor == 0) paint.setColor(Color.RED);
23      else if (whichColor == 1) paint.setColor(Color.GREEN);
24      else if (whichColor == 2) paint.setColor(Color.BLUE);
25      else paint.setColor(Color.YELLOW);
26
27      // draw Rectangle with corners at (40, 20) and (90, 80)
28      canvas.drawRect(40, 20, 90, 80, paint);
29
30
31   }
```

Add an *onTouchEvent* method

```
32   // this method is called when the user touches the View
33   public boolean onTouchEvent(MotionEvent event) {
34
35      // if it's an up ("release") action
36      if (event.getAction() == MotionEvent.ACTION_UP) {
37
38         // get the coordinates
39         float x = event.getX();
40         float y = event.getY();
41
42         // see if they clicked on the box
43         if (x >= 40 && x <= 90 && y >= 20 && y <= 80) {
44
45             // redraw the View... this calls onDraw again!
46             invalidate();
47         }
48      }
49      // indicates that the event was handled
50      return true;
51   } // end of onTouchEvent
52 } // end of DrawableView class
```