Graphics

Programming the Android Platform

Topics

- 2D Graphics using
  - ImageView
  - Canvas
- ViewAnimation

Drawing 2D Graphics

- Two approaches
- Draw to a View
  - Simple graphics, no updating
- Draw to a Canvas
  - More complex graphics, with regular updates

Drawable

- Represents something that can be drawn, such as a bitmap, color, shape, etc.
- Examples:
  - ShapeDrawable
  - BitmapDrawable
  - ColorDrawable

Drawing to Views

- Can draw
  - Via XML
  - Programmatically

Drawing to Views via XML

```
<LinearLayout …
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:background="#FF444444"
   android:id="@+id/linear_layout” >
    <ImageView
       android:id="@+id/imageView1"
       android:layout_height="wrap_content"
       android:layout_width="wrap_content"
       android:src="@drawable/b128”/>
</LinearLayout>
```

Drawing to Views via XML (cont.)

```
public class BubbleActivity extends Activity {
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
  }
}
```

Drawing to Views in Code

```
public class BubbleActivity extends Activity {
  public void onCreate(Bundle savedInstanceState) {
    …
    setContentView(R.layout.main);
    LinearLayout ll = …;
    ImageView bubbleView= new ImageView(this);
    bubbleView.setBackgroundResource(R.drawable.b128);
    bubbleView (new LinearLayout.LayoutParams(
                        ViewGroup.LayoutParams.WRAP_CONTENT,
                        ViewGroup.LayoutParams.WRAP_CONTENT));
     ll.addView(bubbleView);
}
```

ShapeDrawable

- Use for drawing primitive shapes
- Shape represented by Shape class, e.g.,
  - PathShape - lines
  - RectShape - rectangles
  - OvalShape - ovals & rings

ShapeDrawable via XML

```
<RelativeLayout …>
  <LinearLayout android:id="@+id/linearLayout1”” … >
    <ImageView android:id="@+id/imageView1” …/>
    <ImageView android:id="@+id/imageView2”
          android:src="@drawable/cyan_shape” …/>
  </LinearLayout>
  <ImageView android:id="@+id/imageView3”
          android:src="@drawable/magenta_shape” …/>
</RelativeLayout>
```

ShapeDrawable via XML (cont.)

```
// cyan_shape.xml
<shape android:shape="oval" ... >
   <size  android:width="160px" android:height="160px" />
   <solid  android:color="#7F00FFFF" />
</shape>

//  magenta_shape.xml
<shape android:shape="oval" ...>
   <size  android:width="160px" android:height="160px"/>
   <solid  android:color="#7FFF00FF" />
</shape>
```

ShapeDrawable via XML (cont.)

```
public class ShapeDrawActivity extends Activity {
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
  }
}
```

ShapeDrawable in Code

```
public void onCreate(Bundle savedInstanceState) {
 ...
 RelativeLayout rl = ...;
 myDrawableView magentaView =
                    new myDrawableView(this, Color.MAGENTA);
 magentaView.setLayoutParams(
                    new LinearLayout.LayoutParams(160, 160));
 myDrawableView cyanView =
                    new myDrawableView(this, Color.CYAN);
 cyanView.setLayoutParams(
                    new LinearLayout.LayoutParams(160, 160));
...
```

ShapeDrawable in Code (cont.)

```
...
 ImageView spacerView = new ImageView(this);
 spacerView.setMinimumWidth(50);
 spacerView.setAlpha(0);
 LinearLayout ll = new LinearLayout(this);
 ll.setLayoutParams(new LinearLayout.LayoutParams(
         LinearLayout.LayoutParams.WRAP_CONTENT,
         LinearLayout.LayoutParams.WRAP_CONTENT));
 ll.addView(spacerView);
 ll.addView(cyanView);
 rl.addView(ll);
 rl.addView(magentaView);
}
```

ShapeDrawable in Code

```
...
private class myDrawableView extends ImageView {
  private ShapeDrawable mDrawable;
  public myDrawableView(Context context, int color) {
     ...
    mDrawable = new ShapeDrawable(new OvalShape());
    mDrawable.getPaint().setColor(color);
    // must set bounds or ShapeDrawable will not be displayed
    mDrawable.setBounds(0, 0, size, size);
    mDrawable.setAlpha(alpha);
  }
  protected void onDraw(Canvas canvas) {
    mDrawable.draw(canvas);
  }
...
```

Canvas

- Helper class for drawing Bitmaps
  - Bitmap represents a matrix of Pixels
- Drawing operations additionally require
  - Something to draw (e.g. Rect, Path, Text, Bitmap)
  - A paint object (for setting drawing colors & styles)

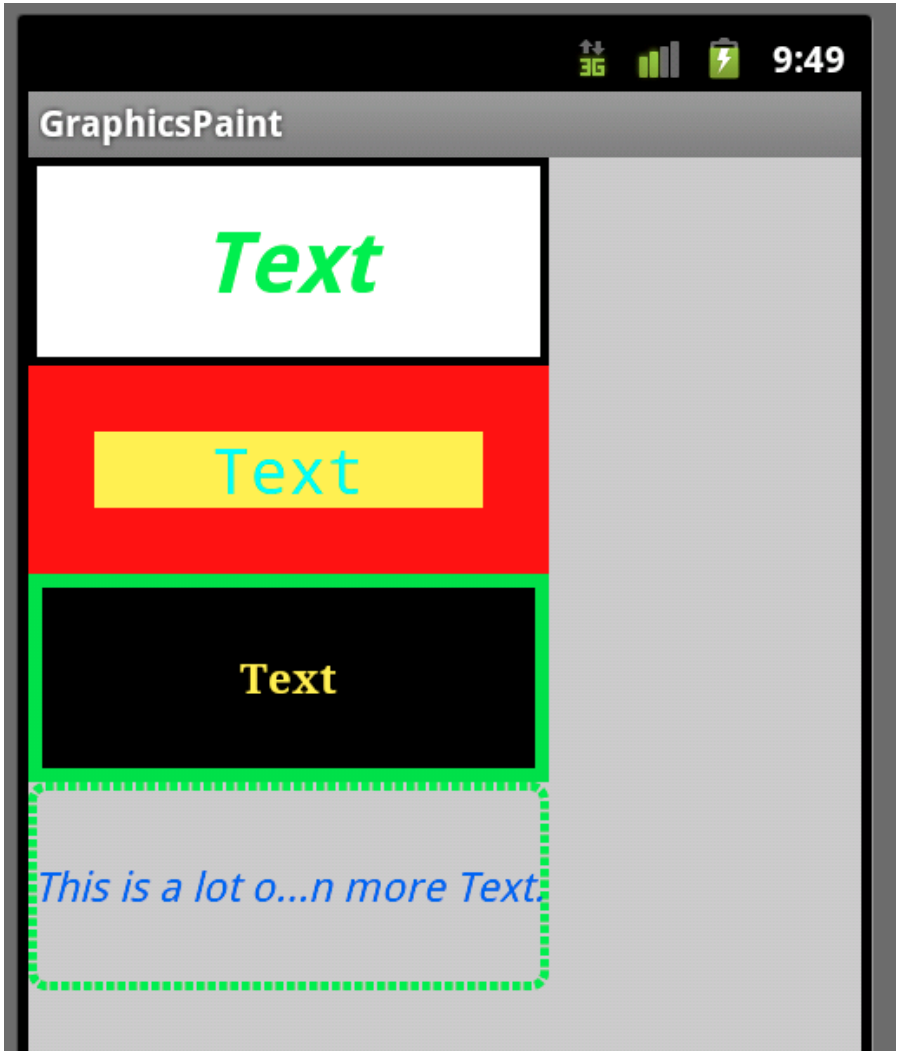
Drawing Primitives

- Canvas supports multiple drawing methods, e.g.,
  - drawText()
  - drawPoints()
  - drawColor()
  - drawOval()
  - drawBitmap()

Paint

- Specifies style parameters for drawing, e.g.,
  - setAntiAlias()
  - setStrokeWidth()
  - setTextSize()
  - setColor()

Paint (cont.)



- Some paint settings
- Text
  - Size, Color, Style, Typeface
- Background
  - Border color, border width, border dash, fill color

Using a Canvas

- Can draw to generic Views or SurfaceViews
- Drawing to Views
  - System provides canvas and controls when drawing occurs
- Drawing to SurfaceView
  - User provides canvas and has greater control over drawing

Canvas with View

```
private class BubbleView extends View {
  ...
 Paint painter = new Paint();
 public BubbleView(Context context, Bitmap bitmap) {
  // Initialize object
 }
 protected void onDraw(Canvas canvas) {
  canvas.drawBitmap(bitmap, y, x, painter);
 }
 protected boolean move() {
  // Move bubble, return false when bubble moves offscreen
 }
}
```

Canvas with View (cont.)

```
public class BubbleActivity extends Activity {
…
 public void onCreate(Bundle savedInstanceState) {
   …
   RelativeLayout frame =
                    (RelativeLayout) findViewById(R.id.frame);
   Bitmap bm = BitmapFactory.decodeResource(
                                        getResources(),R.drawable.b128);
   final BubbleView bubbleView = new BubbleView(this, bm);
   frame.addView(bubbleView);
   …
```

Canvas with View (cont.)

```
...
   // This approach will perform poorly
   new Thread(new Runnable() {
    public void run() {
     while (bubbleView.move()) {
         bubbleView.postInvalidate();
     }
    }
   }).start();
}
```

Canvas with SurfaceView

- If you want to draw high-performance drawing outside the UI thread
- Subclass SurfaceView & implement SurfaceHolder.Callback
- SurfaceView manages a low-level drawing area called a Surface
- SurfaceHolder.Callback declares lifecycle methods called when Surface changes

Canvas with SurfaceView (cont.)

- Using a SurfaceView
  - Setup
  - Use SurfaceHandler.getHolder() to acquire Surface
  - Register for callbacks with addCallback()
  - Create a thread for drawing operations & pass it the SurfaceHandler

Canvas with SurfaceView (cont.)

- To draw
- Acquire lock on Canvas
  - e.g., SurfaceHolder.lockCanvas()
- Draw
  - e.g., Canvas.drawBitmap()
- Unlock Canvas
  - e.g., SurfaceHolder.unlockCanvasAndPost()

Canvas with SurfaceView (cont.)

```
private class BubbleView extends SurfaceView implements
 ...
 final SurfaceHolder mSurfaceHolder;
 final Paint mPainter = new Paint();
 ...
 public BubbleView(Context context, Bitmap bitmap) {
  ...
  mSurfaceHolder = getHolder();
  mSurfaceHolder.addCallback(this);
  ...
 }
```

Canvas with SurfaceView (cont.)

```
public void surfaceCreated(SurfaceHolder holder) {
  new Thread(new Runnable() {
    public void run() {
      Canvas canvas = null;
      while (move()) {
        canvas = mSurfaceHolder.lockCanvas();
        onDraw(canvas);
        mSurfaceHolder.unlockCanvasAndPost(canvas);
      }
    }
  }).start();
}
```

Canvas with SurfaceView (cont.)

```
protected void onDraw(Canvas canvas) {
  canvas.drawColor(Color.DKGRAY);
  mRotation += 1.0f;
  canvas.rotate(mRotation,
                    mY +mBitmapWidthAdj,
                    mX + mBitmapHeightAdj);
  canvas.drawBitmap(mBitmap, mY, mX, mPainter);
}
```

Animation

- TransitionDrawable
- Tween Animation
- Frame-by-Frame Animation

TransitionDrawable

- A 2-layer Drawable
- Fades between 1st & 2nd layers

TransitionDrawable (cont.)

```
// shape_transition.xml
<transition … >
   <item android:drawable="@drawable/cyan_shape"
                                                android:right="50px"/>
   <item android:drawable="@drawable/magenta_shape"
                                                android:left="50px"/>
</transition>

// cyan_shape.xml
<shape android:shape="oval” … >
   <size  android:width=”160px”  android:height=”160px" />
   <solid  android:color="#7F00ffff" />
</shape>
```

TransitionDrawable (cont.)

```
public void onCreate(Bundle savedInstanceState) {
 ...
 setContentView(R.layout.main);
 RelativeLayout rl = ...
 ImageView iv = ...
 TransitionDrawable trans = (TransitionDrawable)
          getResources().getDrawable(R.drawable.shape_transition);
 iv.setImageDrawable(trans);
 rl.addView(iv);
 trans.setCrossFadeEnabled(true);
 trans.startTransition(5000);
}
```

Tween Animation

- Animates the contents of a View
- Animation - a series of timed changes to View properties, such as size, position, alpha, etc.
  - Manipulate times to give effect of sequential or simultaneous changes

Tween Animation (cont.)

```
// icon_animation.xml
<set … >
  <alpha android:fromAlpha="0.0" android:toAlpha="1.0"
         android:duration="3000" … />
 <rotate android:fromDegrees="0" android:toDegrees="360"
    android:pivotX="50%" android:pivotY="50%"
    android:duration="3000" android:startOffset="3000" … />
 <translate android:fromXDelta="0" android:toXDelta="50"
    android:fromYDelta="0" android:toYDelta="50"
    android:startOffset="6000" android:duration="2000" … />
</set>
```

Tween Animation (cont.)

```
public class GraphicsTweenAnimationActivity extends Activity {
 public void onCreate(Bundle savedInstanceState) {
 …
 setContentView(R.layout.main);
 ImageView iv = (ImageView) findViewById(R.id.icon);
 Animation anim = AnimationUtils.loadAnimation(
                                        this,R.anim.icon_animation);
 iv.startAnimation(anim);
 }
}
```

Frame-by-Frame Animation

- Animates a series of View
- Each View shown for a specific amount of time

Frame-by-Frame Animation (cont.)

```
// view_animation.xml
<animation-list android:oneshot="true" … >
<item android:drawable="@android:id/empty" android:duration="100"/>
<item android:drawable="@drawable/nine" android:duration="1000" />
…
<item android:drawable="@drawable/two" android:duration="1000" />
<item android:drawable="@drawable/one" android:duration="1000" />
<item android:drawable="@drawable/robot" android:duration="1000" />
</animation-list>
```

Frame-by-Frame Animation (cont.)

```
public class GraphicsFrameAnimationActivity extends Activity {
  public void onCreate(Bundle savedInstanceState) {
    …
    ImageView iv = (ImageView) findViewById(R.id.countdown_frame);
    iv.setBackgroundResource(R.drawable.view_animation);
    final AnimationDrawable ivAnim =
                                  (AnimationDrawable) iv.getBackground();
    final Button button = (Button) findViewById(R.id.button1);
    button.setOnClickListener(new OnClickListener() {
      public void onClick(View v) {
        if (ivAnim.isRunning()) { ivAnim.stop(); }
        ivAnim.start();
        }
    …
```

Source Code Examples

- GraphicsBubbleProgram
- GraphicsBubbleXML
- GraphicsShapeDraw
- GraphicsShapeDrawXML
- GraphicsPaint
- GraphicsCanvasBubble
- GraphicsCanvasBubbleSurfaceView
- GraphicsTransitionDrawable
- GraphicsFrameAnimation
- GraphicsTweenAnimation