Graphics

Android Graphics

- Custom 2D graphics library
- OpenGL ES 1.0 for high performance 3D graphcs

- The design of an application and the APIs used depend on the graphical demands:
  - static graphical application
  - dynamic interactive 2D and 3D rendering for games

2D Graphics

- Drawing 2D graphics is done in one of two ways:
- Draw the graphics/animations into a View and let Android's View hierarchy take care of the drawing process
- Draw the graphics/animation directly to the Canvas by calling the appropriate class's draw() method passing a Canvas

Drawables

- A “Drawable” is a general abstraction for “something that can be drawn”
- The Drawable class provides a generic API for dealing with visual resources
- Subclasses for Drawable include (but are not limited to)
  - BitmapDrawable
  - ShapeDrawable
  - PictureDrawable
  - LayerDrawable

Adding Graphics

- Referencing an image (PNG (preferred), JPG (acceptable), GIF (discouraged)) is the easiest way to add graphics
- IMPORTANT
  - Images placed in res/drawable may be optimized with lossless compression by the aapt tool
  - Images placed in the res/raw folder are not optimized

Sample ImageView Code #1

```
protected void onCreate(Bundle savedInstanceState)
 {
     super.onCreate(savedInstanceState);

     // Add ImageView to the LinearLayout
     mLinearLayout = new LinearLayout (this);

     // Instantiate an ImageView
     ImageView i = new ImageView(this);
     i.setImageResource(R.drawable.ball_blue);

     // Add the ImageView to the LinearLayout
     mLinearLayout.addView(i);

     setContentView(mLinearLayout);
 }
```

Graphics Basics - Colors

- Android colors are represented with four numbers representing: alpha, red, green, blue
- Each value is 256 possible values (8-bits)
- alpha – a measure of transparency
  - 0 implies complete transparency
  - 255 implies completely opaque
  - middle values are translucent or semitransparent colors

Graphics Basics – Defining Colors

```
// solid blue
int color = Color.BLUE

// translucent purple
int color = Color.argb (128, 255, 0, 255)

<?xml version="1.0 encoding="utf-8"?>
<resources>
  <color name="ballcolor">#80ff00ff</color>
<resources>


color = getResources().getColor (R.color.ballcolor)
```

Graphics Basics - Paint

- The Paint class is extremely important
- Holds style, color, … for drawing
  - bitmaps
  - text
  - geometric shapes

```
cPaint.setColor (Color.BLUE);
```

Graphics Basics - Canvas

- Canvas – surface drawn on
- To draw something you need:
  - a Bitmap to hold the pixels
  - a Canvas to host the draw calls
  - a drawing primitive (e.g. Rect)
  - a paint
- An Activity hosts a View which hosts a Canvas
- Draw on the Canvas by overriding View.onDraw ()

```
setContentView (new GraphicsView (this));

private class GraphicsView extends View{
  public GraphicsView (Context context){
    super (context);
  }
  @Override
  public void onDraw (Canvas canvas){
    // create a path along circle with center (x,y) & radius
    Path circle = new Path();
    circle.addCircle(150, 150, 100, Direction.CW);

    // set the color and font size
    Paint paint = new Paint();
    paint.setColor(Color.BLUE);
    paint.setTextSize(50);
    paint.setAntiAlias(true);

    // draw the text along the circle
    canvas.drawTextOnPath(QUOTE, circle, 0, 30, paint);
    super.onDraw (canvas);
```

Results

Simple Text Animation

- 11.Code\SimpleAnimation

Animation Problem

Download 11.Code\BallAnimate

1. Get the blue ball bitmap to display on the screen
2. Get the ball to move across the entire screen horizontally
3. Get the ball to move randomly across the screen bouncing off any wall