CMSC436: Programming Handheld Systems

2D Graphics
&
Animation

Topics

- 2D Graphics
  - ImageView
  - Canvas
- View Animation
- Property Animation

Drawing 2D Graphics

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

Drawable

Something that can be drawn, such as a bitmap, color, shape, etc.

Examples:

BitmapDrawable

ShapeDrawable

ColorDrawable

Drawing to Views

Can set Drawable objects on Views

Can do this via XML or programmatically

GraphicsBubble

Applications display a single ImageView

ImageView holds an image of a bubble

Graphics
BubbleXML

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/frame"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:padding="@dimen/activity_margin">

   <ImageView
      android:id="@+id/imageView1"
      android:layout_width="@dimen/bubble_width"
      android:layout_height="@dimen/bubble_width"
      android:layout_centerInParent="true"
      android:contentDescription="@string/bubble_desc"
      android:src="@drawable/b512"
      android:tint="@android:color/white"/>

</RelativeLayout>
```

Graphics
BubbleProgram

```
public void onCreate(Bundle savedInstanceState) {
    …
    RelativeLayout relativeLayout = findViewById(R.id.frame);
    ImageView bubbleView = new ImageView(getApplicationContext());
    BitmapDrawable tmp = (BitmapDrawable) getDrawable(R.drawable.b512);
    tmp.setTint(Color.WHITE);
    bubbleView.setImageDrawable(tmp);

    int width = (int) getResources().getDimension(R.dimen.image_width);
    int height = (int) getResources().getDimension(R.dimen.image_height);

    RelativeLayout.LayoutParams params =
                        new RelativeLayout.LayoutParams(width, height);
    params.addRule(RelativeLayout.CENTER_IN_PARENT);
    bubbleView.setLayoutParams(params);
    relativeLayout.addView(bubbleView);
}
```

ShapeDrawable

Used for drawing primitive shapes

Shape represented by a Shape class

PathShape - lines

RectShape - rectangles

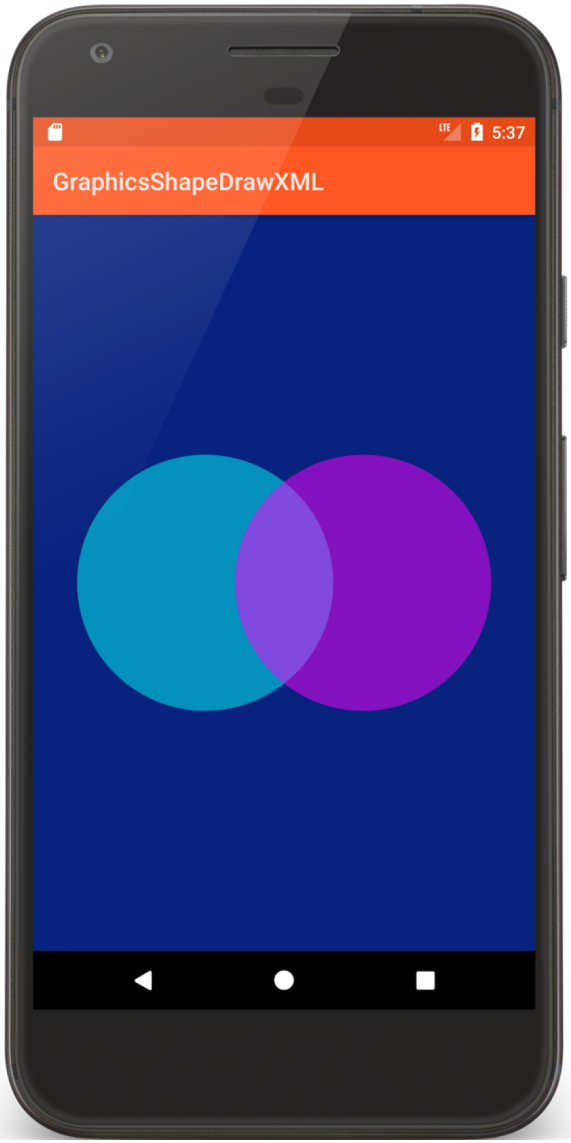
OvalShape - ovals & rings

GraphicsShapeDraw

Applications display two Shapes within a RelativeLayout

The two shapes are partially overlapping and semi-transparent

Graphics
ShapeDrawXML



```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:padding="@dimen/activity_margin">

   <ImageView
      android:layout_width="250dp"
      android:layout_height="250dp"
      android:layout_alignParentStart="true"
      android:layout_centerVertical="true"
      android:contentDescription="@string/cyan_circle"
      android:padding="20dp"
      android:src="@drawable/cyan_shape" />
...
```

```
    ...
    <ImageView
        android:layout_width="250dp"
        android:layout_height="250dp"
        android:layout_alignParentEnd="true"
        android:layout_centerVertical="true"
        android:contentDescription="@string/magenta_circle"
        android:padding="20dp"
        android:src="@drawable/magenta_shape" />

</RelativeLayout>
```

```
//cyan_shape.xml
<shape xmlns:android="http://schemas.android.com/apk/res/android"
   android:shape="oval" >
   <solid android:color="#7F00ffff" />
</shape>


//magenta_shape.xml
<shape xmlns:android="http://schemas.android.com/apk/res/android"
   android:shape="oval" >
   <solid android:color="#7Fff00ff" />
</shape>
```

Graphics
ShapeDraw

```
public void onCreate(Bundle savedInstanceState) {
…
  int width = (int) getResources().getDimension(R.dimen.image_width);
  int height = (int) getResources().getDimension(R.dimen.image_height);
  int padding = (int) getResources().getDimension(R.dimen.padding);

  // Get container View
  RelativeLayout rl = (RelativeLayout) findViewById(R.id.main_window);

  // Create Cyan Shape
  ShapeDrawable cyanShape = new ShapeDrawable(new OvalShape());
  cyanShape.getPaint().setColor(Color.CYAN);
  cyanShape.setIntrinsicHeight(height);
  cyanShape.setIntrinsicWidth(width);
  cyanShape.setAlpha(ALPHA);
   …
```

```
…
// Put Cyan Shape into an ImageView
ImageView cyanView = new ImageView(getApplicationContext());
cyanView.setImageDrawable(cyanShape);
cyanView.setPadding(padding, padding, padding, padding);

// Specify placement of ImageView within RelativeLayout
RelativeLayout.LayoutParams cyanViewLayoutParams =
                                new RelativeLayout.LayoutParams(height, width);
cyanViewLayoutParams.addRule(RelativeLayout.CENTER_VERTICAL);
cyanViewLayoutParams.addRule(RelativeLayout.ALIGN_PARENT_LEFT);
cyanView.setLayoutParams(cyanViewLayoutParams);
rl.addView(cyanView);
…
```

```
…
    // Create Magenta Shape
    ShapeDrawable magentaShape = new ShapeDrawable(new OvalShape());
    magentaShape.getPaint().setColor(Color.MAGENTA);
    magentaShape.setIntrinsicHeight(height);
    magentaShape.setIntrinsicWidth(width);
    magentaShape.setAlpha(ALPHA);

    // Put Magenta Shape into an ImageView
    ImageView magentaView = new ImageView(getApplicationContext());
    magentaView.setImageDrawable(magentaShape);
    magentaView.setPadding(padding, padding, padding, padding);
   …
```

```
…
// Specify placement of ImageView within RelativeLayout
RelativeLayout.LayoutParams magentaViewLayoutParams =
                                new RelativeLayout.LayoutParams(height, width);
magentaViewLayoutParams.addRule(RelativeLayout.CENTER_VERTICAL);
magentaViewLayoutParams.addRule(RelativeLayout.ALIGN_PARENT_RIGHT);

magentaView.setLayoutParams(magentaViewLayoutParams);

rl.addView(magentaView);
…
```

Drawing with a Canvas

A Bitmap (a matrix of Pixels)

A Canvas for drawing to the underlying Bitmap

A Drawing Primitive (e.g. Rect, Path, Text, Bitmap)

A paint object (for setting drawing colors & styles)

Drawing Primitives

Canvas supports multiple drawing methods

- drawText()
- drawPoints()
- drawColor()
- drawOval()
- drawBitmap()

Paint

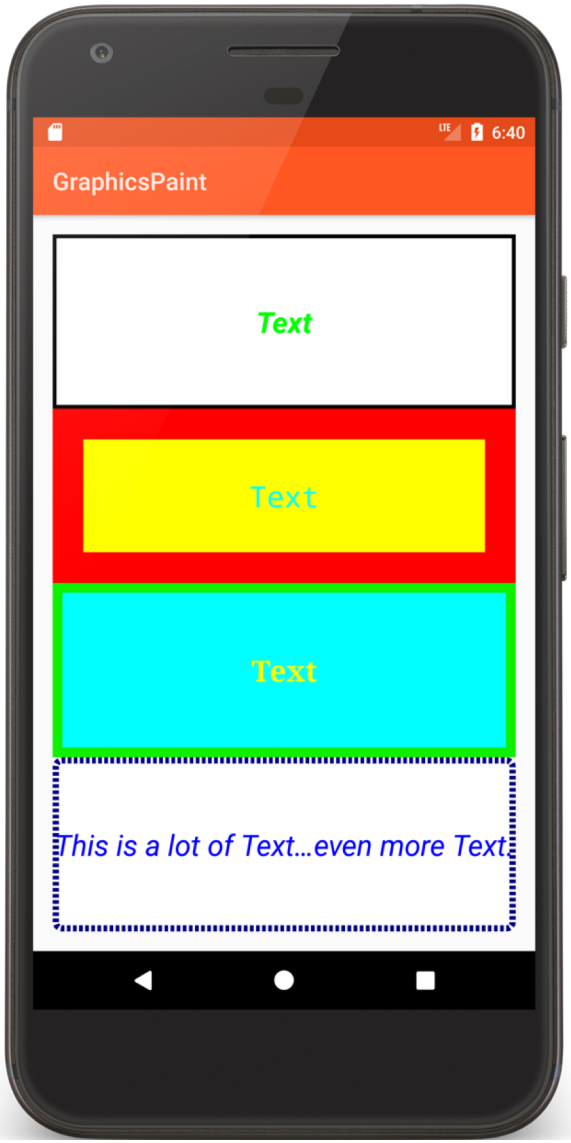
Specifies style parameters for drawing, e.g.,

- setStrokeWidth()
- setTextSize()
- setColor()
- setAntiAlias()

GraphicsPaint

Application draws several boxes holding text, using different paint settings each time

Graphics
Paint



```
<TextView
   android:layout_width="match_parent"
   android:layout_height="0dp"
   android:layout_weight="1"
   android:background="@drawable/sq1"
   android:gravity="center"
   android:text="@string/text_literal"
   android:textColor="#ff00ff00"
   android:textSize="24sp"
   android:textStyle="bold|italic"
   android:typeface="normal" />
```

```
<TextView
   android:layout_width="match_parent"
   android:layout_height="0dp"
   android:layout_weight="1"
   android:background="@drawable/sq2"
   android:gravity="center"
   android:text="@string/text_literal"
   android:textColor="#FF00FFFF"
   android:textSize="24sp"
   android:textStyle="normal"
   android:typeface="monospace" />
```

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:background="@drawable/sq3"
    android:gravity="center"
    android:text="@string/text_literal"
    android:textColor="#FFFFFF00"
    android:textSize="24sp"
    android:textStyle="bold"
    android:typeface="serif" />
```

```
<TextView
   android:layout_width="match_parent"
   android:layout_height="0dp"
   android:layout_weight="1"
   android:background="@drawable/sq4"
   android:ellipsize="middle"
   android:gravity="center"
   android:singleLine="true"
   android:text="@string/long_text"
   android:textColor="#FF0000FF"
   android:textSize="24sp"
   android:textStyle="italic"
   android:typeface="sans" />
```

Drawing with a Canvas

Can draw to generic Views, or to SurfaceViews

Drawing to Views

Use when updates are infrequent

Create a custom View class

System provides the Canvas for the View when it calls the View’s onDraw() method

Drawing to SurfaceViews

Use when updates are frequent

Create a custom SurfaceView

Provide secondary thread for drawing

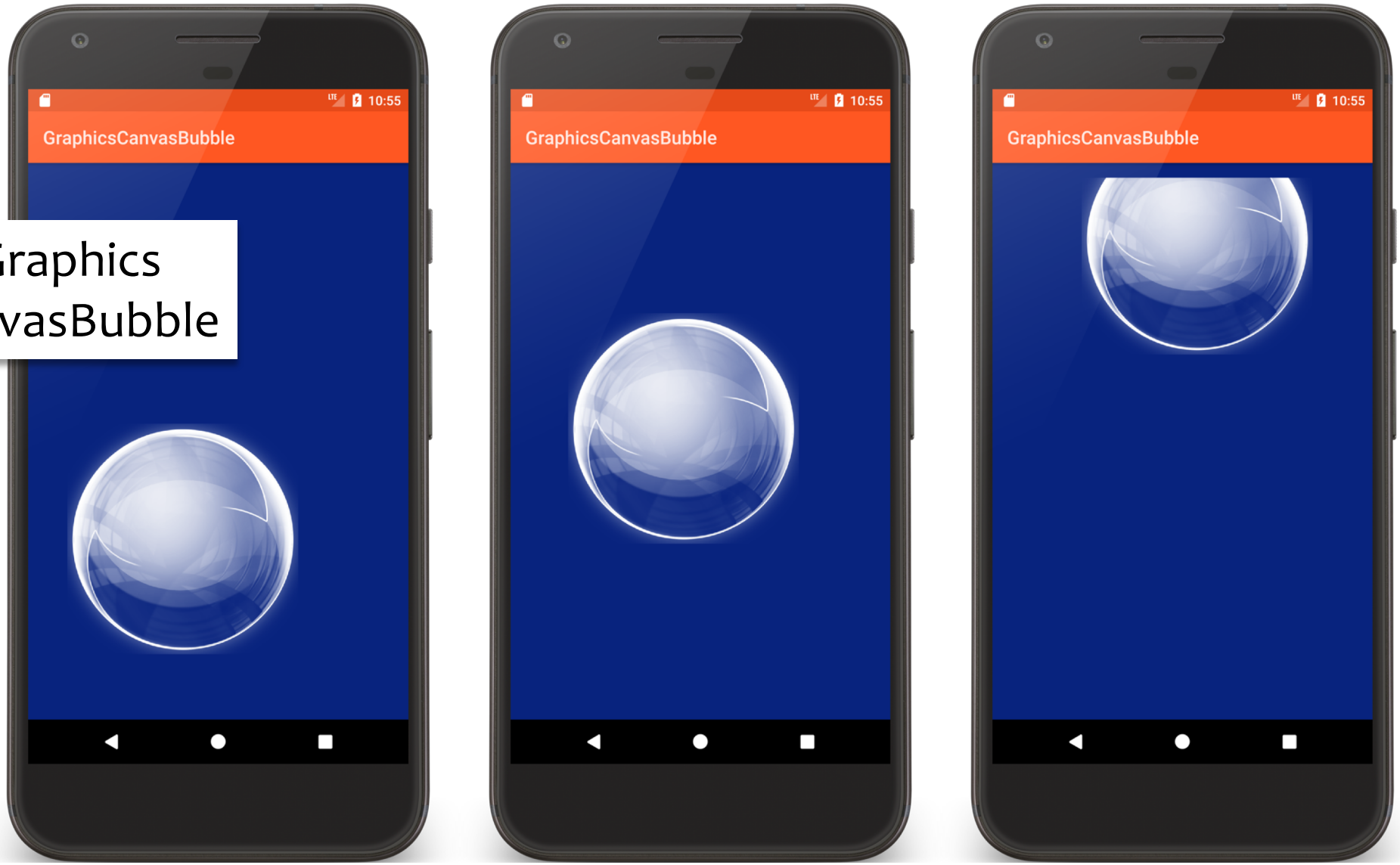
Application provides its own Canvas and has greater control over drawing

GraphicsCanvasBubble

This application draws to custom View

It has an internal Thread that periodically wakes up and causes the View to move and to be redrawn

Graphics
CanvasBubble



```
public void onCreate(Bundle savedInstanceState) {
  …
  final RelativeLayout frame = findViewById(R.id.frame);
  final Bitmap bitmap = BitmapFactory.decodeResource(getResources(), R.drawable.b512);
  final BubbleView bubbleView = new BubbleView(getApplicationContext(), bitmap);
  frame.addView(bubbleView);
    new Thread(new Runnable() {
      public void run() {
        while (bubbleView.moveWhileOnscreen()) {
          // sleep for 1 second
          bubbleView.postInvalidate();
        } } } ).start();
  }
```

```
private class BubbleView extends View {
  …
  public BubbleView(Context context, Bitmap bitmap) {
    super(context);
    // Scale bitmap
    mBitmapWidthAndHeight = (int) getResources()
                                                .getDimension(R.dimen.image_height);
    this.mBitmap = Bitmap.createScaledBitmap(bitmap,
                          mBitmapWidthAndHeight, mBitmapWidthAndHeight, false);
      mBitmapWidthAndHeightAdj = mBitmapWidthAndHeight + 20;
    // Get display size info
     mDisplayMetrics = new DisplayMetrics();
    BubbleActivity.this.getWindowManager()
                                            .getDefaultDisplay().getMetrics(mDisplayMetrics);
    mDisplayWidth = mDisplayMetrics.widthPixels;
    mDisplayHeight = mDisplayMetrics.heightPixels;
  …
```

```
...
    // Set random starting point
    Random r = new Random();
    float x = (float) r.nextInt(mDisplayWidth - mBitmapWidthAndHeight);
    float y = (float) r.nextInt(mDisplayHeight - mBitmapWidthAndHeight);
    mCurrent = new Coords(x, y);
    // Set random movement direction and speed
     float dy = Math.max(r.nextFloat(),0.1f) * STEP;
    dy *= r.nextInt(2) == 1 ? 1 : -1;
    float dx = Math.max(r.nextFloat(),0.1f) * STEP;
    dx *= r.nextInt(2) == 1 ? 1 : -1;
    mDxDy = new Coords(dx, dy);
    // Add some painting directives
      mPainter.setAntiAlias(true);
      mPainter.setColorFilter(
            new PorterDuffColorFilter(Color.WHITE, PorterDuff.Mode.SRC_ATOP));

  }
```

```
protected void onDraw(Canvas canvas) {
        Coords tmp = mCurrent.getCoords();
        canvas.drawBitmap(mBitmap, tmp.mX, tmp.mY, mPainter);
    }

 boolean moveWhileOnscreen() {
        mCurrent = mCurrent.move(mDxDy);

          return !(mCurrent.mY < 0 - mBitmapWidthAndHeightAdj
                || mCurrent.mY > mDisplayHeight + mBitmapWidthAndHeightAdj
                || mCurrent.mX < 0 - mBitmapWidthAndHeightAdj
                || mCurrent.mX > mDisplayWidth + mBitmapWidthAndHeightAdj);
    }
  }
}
```

Canvas with SurfaceView

Used for more high-performance drawing outside the UI thread

SurfaceView

SurfaceView manages a low-level drawing area called a Surface

The Surface represent a drawing area within the View hierarchy

Defining a Custom SurfaceView

Subclass SurfaceView & implement SurfaceHolder.Callback

SurfaceHolder.Callback declares lifecycle methods that are called when the Surface changes

Using a SurfaceView

Set up SurfaceView

Draw to SurfaceView

Setup

Use SurfaceView’s getHolder() to acquire Surface

Setup

Register for callbacks with SurfaceHolder’s addCallback()

surfaceCreate()

surfaceChanged()

surfaceDestroyed()

Setup

Create the Thread on which drawing operations will execute

Drawing

Acquire lock on Canvas

SurfaceHolder.lockCanvas()

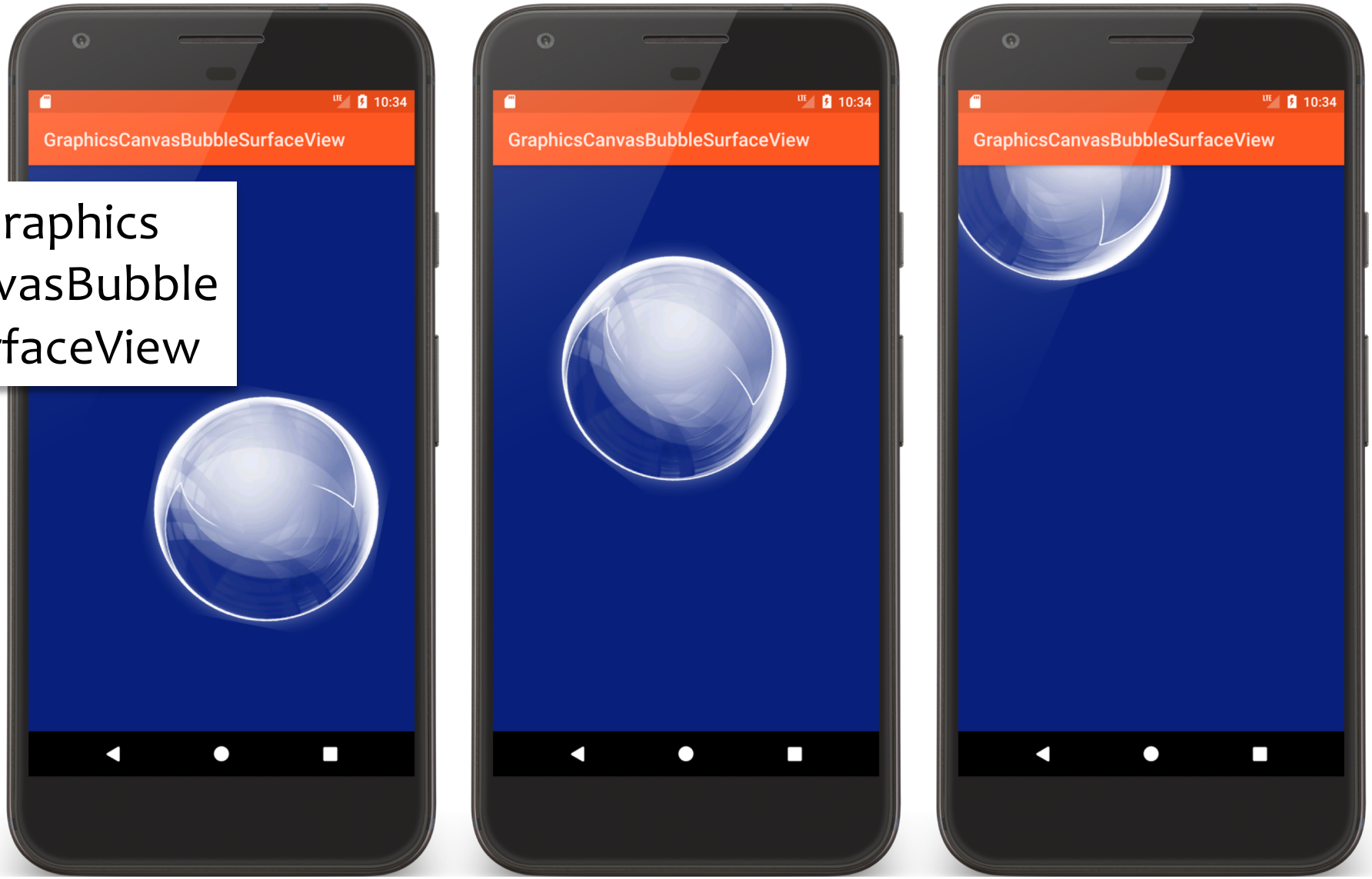
Draw

Canvas.drawBitmap()

Unlock Canvas

SurfaceHolder.unlockCanvasAndPost()

Graphics
CanvasBubble
SurfaceView



```
public class BubbleActivity extends Activity {

   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);

      final RelativeLayout relativeLayout = findViewById(R.id.frame);
      final BubbleView bubbleView = new BubbleView(getApplicationContext(),
            BitmapFactory.decodeResource(getResources(), R.drawable.b512));

      relativeLayout.addView(bubbleView);
   }
```

```
private class BubbleView extends SurfaceView implements SurfaceHolder.Callback {
  …
   public BubbleView(Context context, Bitmap bitmap) {
       super(context);

    // Resize bubble
     mBackgroundColor = context.getResources()
                                          .getColor(R.color.background_color, null);
     mBitmapHeightAndWidth =
                (int) getResources().getDimension(R.dimen.image_height_width);
     this.mBitmap = Bitmap.createScaledBitmap(bitmap,
                        mBitmapHeightAndWidth, mBitmapHeightAndWidth, false);
    // Set useful parameters
   mBitmapWidthAndHeightAdj = mBitmapHeightAndWidth / 2;
   mBitmapHeightAndWidthPadded = mBitmapHeightAndWidth + mBitmapPadding;
```

```
// Determine screen size
mDisplay = new DisplayMetrics();
BubbleActivity.this.getWindowManager().getDefaultDisplay().getMetrics(mDisplay);
 mDisplayWidth = mDisplay.widthPixels;
 mDisplayHeight = mDisplay.heightPixels;

 // Set random starting point for BubbleView
 Random r = new Random();
 float x = (float) r.nextInt( mDisplayWidth / 4) + mDisplayWidth / 4;
 float y = (float) r.nextInt(mDisplayHeight / 4) + mDisplayHeight / 4;
 mCurrent = new Coords(x, y);

 // Set random movement direction and speed, and set rotation
    …
```

```
 mPainter.setAntiAlias(true);
 mPainter.setColorFilter(
                  new PorterDuffColorFilter(Color.WHITE,PorterDuff.Mode.SRC_ATOP));
  // Prepare surface for drawing
    mSurfaceHolder = getHolder();
    mSurfaceHolder.addCallback(this);
}

private void drawBubble(Canvas canvas) {
    canvas.drawColor(mBackgroundColor);
    mRotation += ROT_STEP;
    canvas.rotate(mRotation, mCurrent.mX + mBitmapWidthAndHeightAdj,
                        mCurrent.mY + mBitmapWidthAndHeightAdj);
    canvas.drawBitmap(mBitmap, mCurrent.mX, mCurrent.mY, mPainter);
}
```

```
public void surfaceCreated(SurfaceHolder holder) {
    mDrawingThread = new Thread(new Runnable() {
        public void run() {
            while (!Thread.currentThread().isInterrupted() && moveWhileOnscreen()) {
                Canvas canvas = mSurfaceHolder.lockCanvas();
                if (null != canvas) {
                  drawBubble(canvas);
                  mSurfaceHolder.unlockCanvasAndPost(canvas);
                } } } });
        mDrawingThread.start();
    }

public void surfaceDestroyed(SurfaceHolder holder) {
        if (null != mDrawingThread)
            mDrawingThread.interrupt();
    }
```

View Animation

Changing View properties over a period of time

- Size
- Position
- Transparency
- Orientation

View Animation Classes

TransitionDrawable

AnimationDrawable

Animation

TransitionDrawable

A 2-layer Drawable

Can fade between 1st & 2nd layers

GraphicsTransitionDrawable

This application uses the same shapes as the GraphicsShapeDraw applications

Shows Cyan shape then fades to Magenta shape

Graphics
TransitionDrawable

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

    setContentView(R.layout.main);

    TransitionDrawable transition = (TransitionDrawable) getResources()
        .getDrawable(R.drawable.shape_transition,null);

    transition.setCrossFadeEnabled(true);

    ((ImageView) findViewById(R.id.image_view)).setImageDrawable(transition);

    transition.startTransition(5000);
  }
```

```
<transition xmlns:android="http://schemas.android.com/apk/res/android">

   <item
      android:drawable="@drawable/cyan_shape"
      android:right="@dimen/image_padding" />
   <item
      android:drawable="@drawable/magenta_shape"
      android:left="@dimen/image_padding" />

</transition>
```

AnimationDrawable

Animates a series of Drawables

Each Drawable is shown for a specific amount of time

GraphicsFrameAnimation

Uses an Animation Drawable to present a frame by frame animation

Graphics
FrameAnimation

```
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
   android:oneshot="true" >
   <item
      android:drawable="@drawable/black_background"
      android:duration="1000"/>
   <item
      android:drawable="@drawable/nine"
      android:duration="1000"/>
 …
 <item
      android:drawable="@drawable/one"
      android:duration="1000"/>
   <item
      android:drawable="@drawable/painter"
      android:duration="1000"/>
</animation-list>
```

```
public class GraphicsFrameAnimationActivity extends Activity {
  private AnimationDrawable mAnim;

  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    ImageView imageView = findViewById(R.id.countdown_frame);

    imageView.setBackgroundResource(R.drawable.view_animation);

    mAnim = (AnimationDrawable) imageView.getBackground();
  }
```

```
protected void onPause() {
    super.onPause();
    if (mAnim.isRunning()) {
      mAnim.stop();
    } }

public void onWindowFocusChanged(boolean hasFocus) {
    super.onWindowFocusChanged(hasFocus);
    if (hasFocus) {
      mAnim.start();
    }  }
}
```

Animation

A series of transformations applied to the content of a View

Can manipulate animation timing to give effect of sequential or simultaneous changes

GraphicsTweenAnimation

Application displays a single ImageView and animates several of its properties

Graphics
TweenAnimation

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
   android:shareInterpolator="false">
   <alpha
      android:duration="3000"
      android:fromAlpha="0.0"
      android:interpolator="@android:anim/linear_interpolator"
      android:toAlpha="1.0" />
   ...
```

```
  …
  <scale
     android:duration="3000"
     android:fromXScale="1"
     android:fromYScale="1"
     android:interpolator="@android:anim/anticipate_interpolator"
     android:pivotX="50%"
     android:pivotY="50%"
     android:startOffset="10000"
     android:toXScale="2"
     android:toYScale="2" />
  …
</set>
```

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    mImageView = (ImageView) findViewById(R.id.icon);

    mAnim = AnimationUtils.loadAnimation(this, R.anim.view_animation);

  }

  public void onWindowFocusChanged(boolean hasFocus) {
    super.onWindowFocusChanged(hasFocus);
    if (hasFocus) {
      mImageView.startAnimation(mAnim);
    }
  }
```

Property Animation

Animation - Changing properties of an Object over a period of time

Property Animation Architecture

ValueAnimator – Timing engine

TimeInterpolator – defines how values change as a function of time

AnimatorUpdateListener – called back at every animation frame change

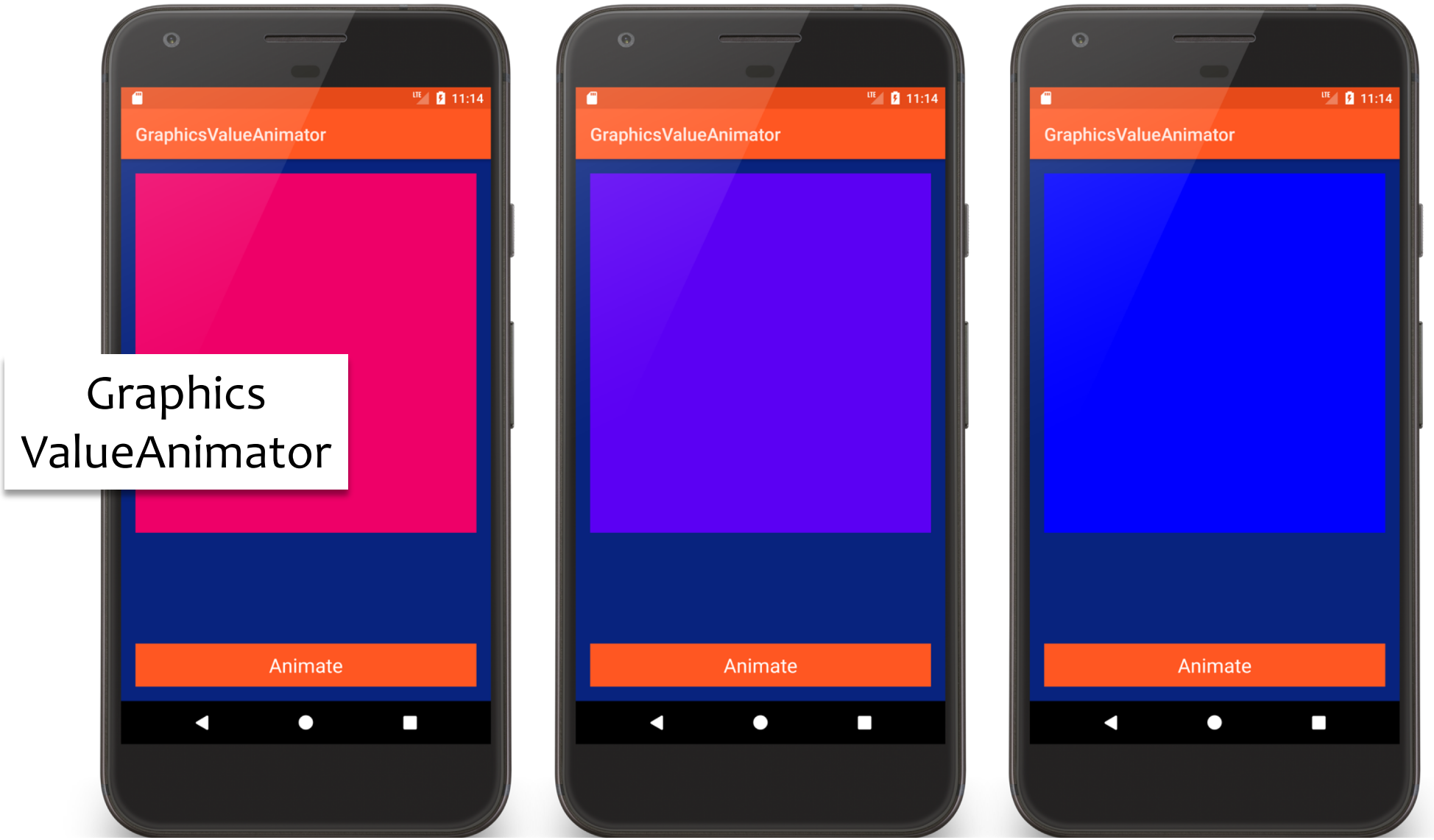
TypeEvaluator – Calculates a property’s value at a given point in time

Property Animation Architecture

AnimatorSet – combines individual animations to create more complex animations

GraphicsValueAnimator

Uses a ValueAnimator to animate changing an ImageView’s background color


Graphics
ValueAnimator
11:14
GraphicsValueAnimator
Animate
11:14
GraphicsValueAnimator
Animate
11:14
GraphicsValueAnimator
Animate

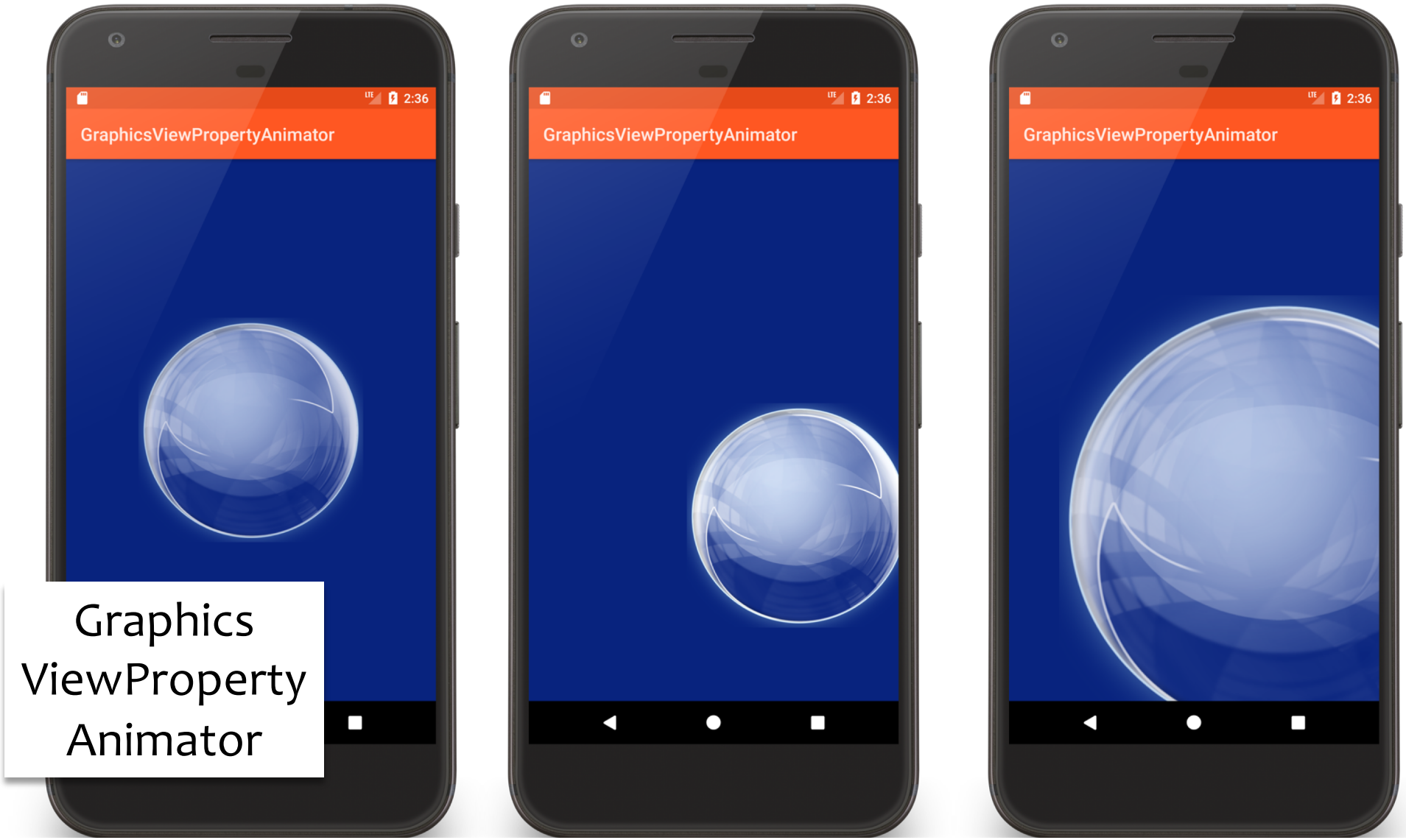
```
public class ValueAnimatorActivity extends Activity {
  …
  final private static int RED = Color.RED;
  final private static int BLUE = Color.BLUE;

  …
  public void onClick(@SuppressWarnings("unused") View v) {
    startAnimation();
  }
```

```
private void startAnimation() {
    final ImageView imageView = findViewById(R.id.image_view);
    ValueAnimator anim = ValueAnimator.ofObject(new ArgbEvaluator(), RED, BLUE);

    anim.addUpdateListener(new AnimatorUpdateListener() {
      public void onAnimationUpdate(ValueAnimator animation) {
        imageView.setBackgroundColor((Integer) animation
            .getAnimatedValue());
      } } );

    anim.setDuration(10000);
    anim.start();
  }
}
```

GraphicsViewPropertyAnimator

Same as the GraphicsTweenAnimation

Uses the ViewPropertyAnimator class, which is a simplified animator for Views



Graphics ViewProperty Animator

```
public void onWindowFocusChanged(boolean hasFocus) {
    super.onWindowFocusChanged(hasFocus);

    mImageView = findViewById(R.id.bubble);

    if (hasFocus) {
      fadeIn.run();
    }
  }

  private final Runnable fadeIn = new Runnable() {
    public void run() {
      mImageView.animate().setDuration(3000)
          .setInterpolator(new LinearInterpolator()).alpha(1.0f)
          .withEndAction(rotate);
    }
  };
```

```
private final Runnable rotate = new Runnable() {
  public void run() {
    mImageView.animate().setDuration(4000)
        .setInterpolator(new AccelerateInterpolator())
        .rotationBy(720.0f).withEndAction(translate);
  } };

private final Runnable translate = new Runnable() {
  public void run() {
    float translation = getResources().getDimension(R.dimen.translation);
    mImageView.animate().setDuration(3000)
        .setInterpolator(new OvershootInterpolator())
        .translationXBy(translation).translationYBy(translation)
        .withEndAction(scale);
  } };
```

```
private final Runnable scale = new Runnable() {
  public void run() {
    mImageView.animate().setDuration(3000)
        .setInterpolator(new AnticipateInterpolator())
        .scaleXBy(1.0f).scaleYBy(1.0f).withEndAction(fadeOut);
  }
};

private final Runnable fadeOut = new Runnable() {
  public void run() {
    mImageView.animate().setDuration(2000)
        .setInterpolator(new DecelerateInterpolator()).alpha(0.0f);
  }
};
```

Next Time

MultiTouch & Gestures

Example Applications

GraphicsBubbleXML

GraphicsBubbleProgram

GraphicsShapeDrawXML

GraphicsShapeDraw

GraphicsPaint

GraphicsCanvasBubble

GraphicsCanvas
BubbleSurfaceView

GraphicsTransitionDrawable

GraphicsFrameAnimation

GraphicsTweenAnimation

GraphicsValueAnimator

GraphicsView
PropertyAnimator