Recording in Progress

This class is being recorded

Please turn off your video and/or video if you do not wish to be recorded

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

main.xml

```xml
<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

ShapeDrawActivity.kt

```
public override fun onCreate(savedInstanceState: Bundle?) {
    …
    val relativeLayout = findViewById<RelativeLayout>(R.id.frame)
    val bubbleView = ImageView(applicationContext)
    val tmp = getDrawable(R.drawable.b512) as BitmapDrawable?
    if (null != tmp) {
        tmp.setTint(Color.WHITE)
        bubbleView.setImageDrawable(tmp)
    }
    val width = resources.getDimension(R.dimen.image_width).toInt()
    val height = resources.getDimension(R.dimen.image_height).toInt()
    val params = RelativeLayout.LayoutParams(width, height)
    params.addRule(RelativeLayout.CENTER_IN_PARENT)
    bubbleView.layoutParams = params
    relativeLayout.addView(bubbleView)
}
```

ShapeDrawable

Used for drawing primitive shapes

Shape represented by a Shape class

PathShape - lines

RectShape - rectangles

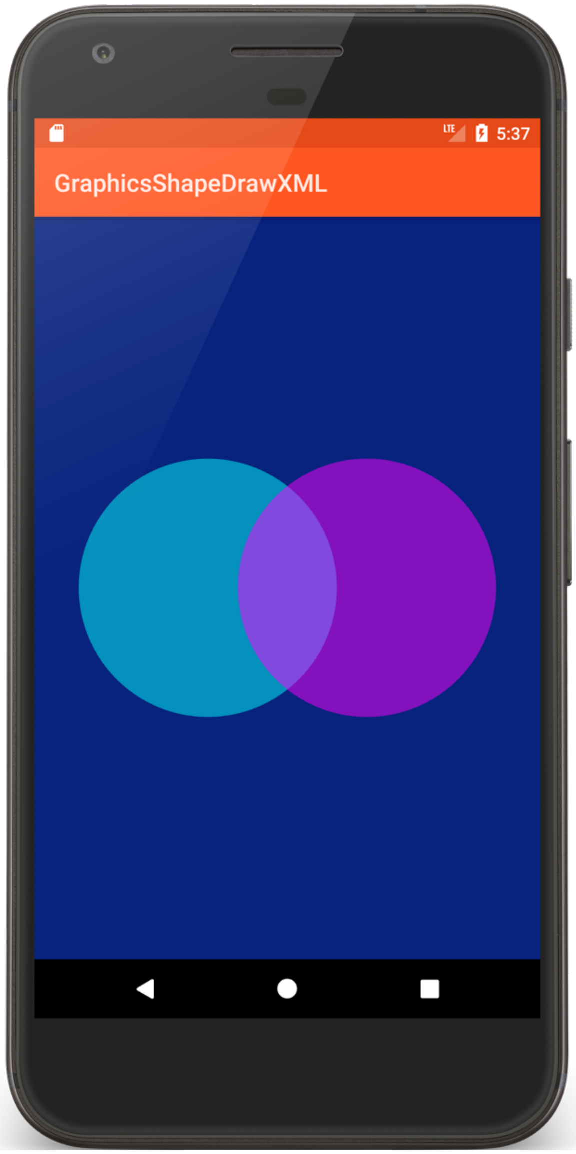
OvalShape - ovals & rings

GraphicsShapeDraw

Applications display two Shapes within a RelativeLayout

The two shapes are partially overlapping and semi-transparent

Graphics
ShapeDrawXML



main.xml

```xml
<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="@dimen/image_size"
        android:layout_height="@dimen/image_size"
        android:layout_alignParentStart="true"
        android:layout_centerVertical="true"
        android:contentDescription="@string/cyan_circle"
        android:padding="@dimen/activity_margin"
        android:src="@drawable/cyan_shape" />
    …
```

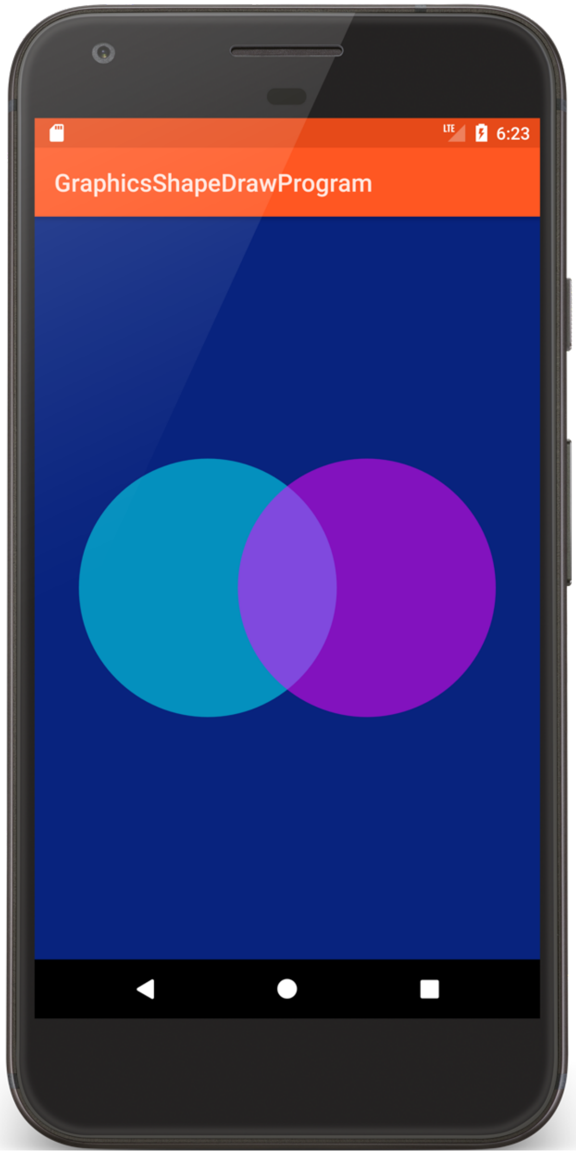
main.xml

```xml
    <ImageView
        android:layout_width="@dimen/image_size"
        android:layout_height="@dimen/image_size"
        android:layout_alignParentEnd="true"
        android:layout_centerVertical="true"
        android:contentDescription="@string/magenta_circle"
        android:padding="@dimen/activity_margin"
        android:src="@drawable/magenta_shape" />

</RelativeLayout>
```

```
//magenta_shape.xml

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

```
//cyan_shape.xml

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

Graphics
ShapeDraw



ShapeDrawActivity.kt

```
public override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.main)
    val width = resources.getDimension(R.dimen.image_width).toInt()
    val height = resources.getDimension(R.dimen.image_height).toInt()
    val padding = resources.getDimension(R.dimen.padding).toInt()

    // Get container View
    val rl = findViewById<RelativeLayout>(R.id.main_window)

    // Create Cyan Shape
    val cyanShape = ShapeDrawable(OvalShape())
    cyanShape.paint.color = Color.CYAN
    cyanShape.intrinsicHeight = height
    cyanShape.intrinsicWidth = width
    cyanShape.alpha = ALPHA
```

ShapeDrawActivity.kt

```
// Put Cyan Shape into an ImageView
val cyanView = ImageView(applicationContext)
cyanView.setImageDrawable(cyanShape)
cyanView.setPadding(padding, padding, padding, padding)

// Specify placement of ImageView within RelativeLayout
val cyanViewLayoutParams = RelativeLayout.LayoutParams(
    height, width
)
cyanViewLayoutParams.addRule(RelativeLayout.CENTER_VERTICAL)
cyanViewLayoutParams.addRule(RelativeLayout.ALIGN_PARENT_LEFT)
cyanView.layoutParams = cyanViewLayoutParams
rl.addView(cyanView)
```

ShapeDrawActivity.kt

```kotlin
// Create Magenta Shape
val magentaShape = ShapeDrawable(OvalShape())
magentaShape.paint.color = Color.MAGENTA
magentaShape.intrinsicHeight = height
magentaShape.intrinsicWidth = width
magentaShape.alpha = ALPHA

// Put Magenta Shape into an ImageView
val magentaView = ImageView(applicationContext)
magentaView.setImageDrawable(magentaShape)
magentaView.setPadding(padding, padding, padding, padding)
```

ShapeDrawActivity.kt

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

    magentaView.layoutParams = 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

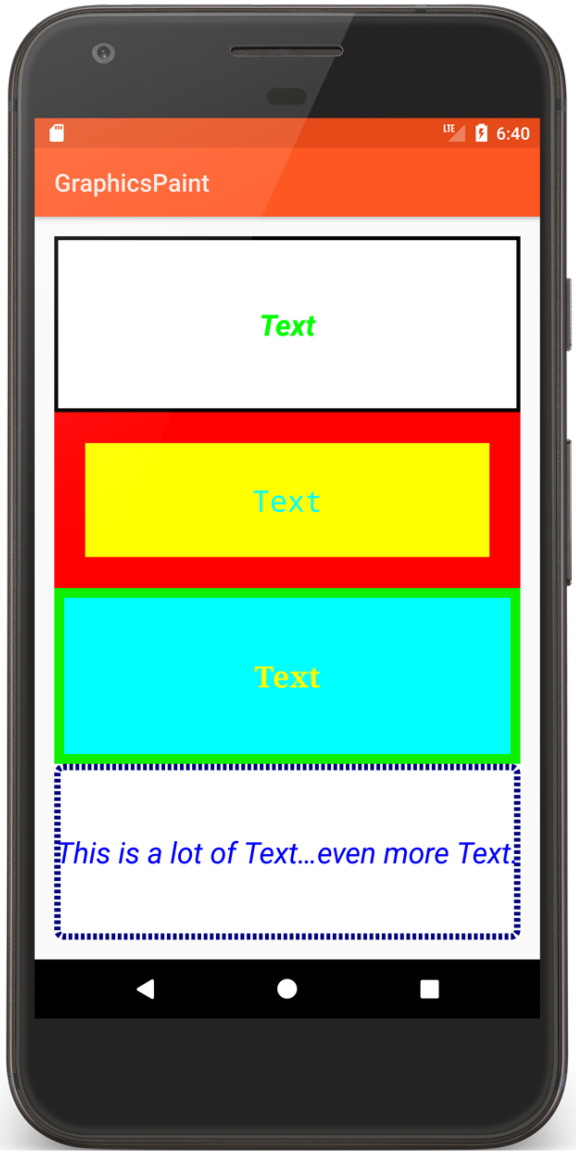
Specifies style parameters for drawing, e.g.,

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

GraphicsPaint

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

Graphics Paint



main.xml

```
<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" />
```

main.xml

```xml
<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" />
```

main.xml

```xml
<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" />
```

main.xml

```
<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

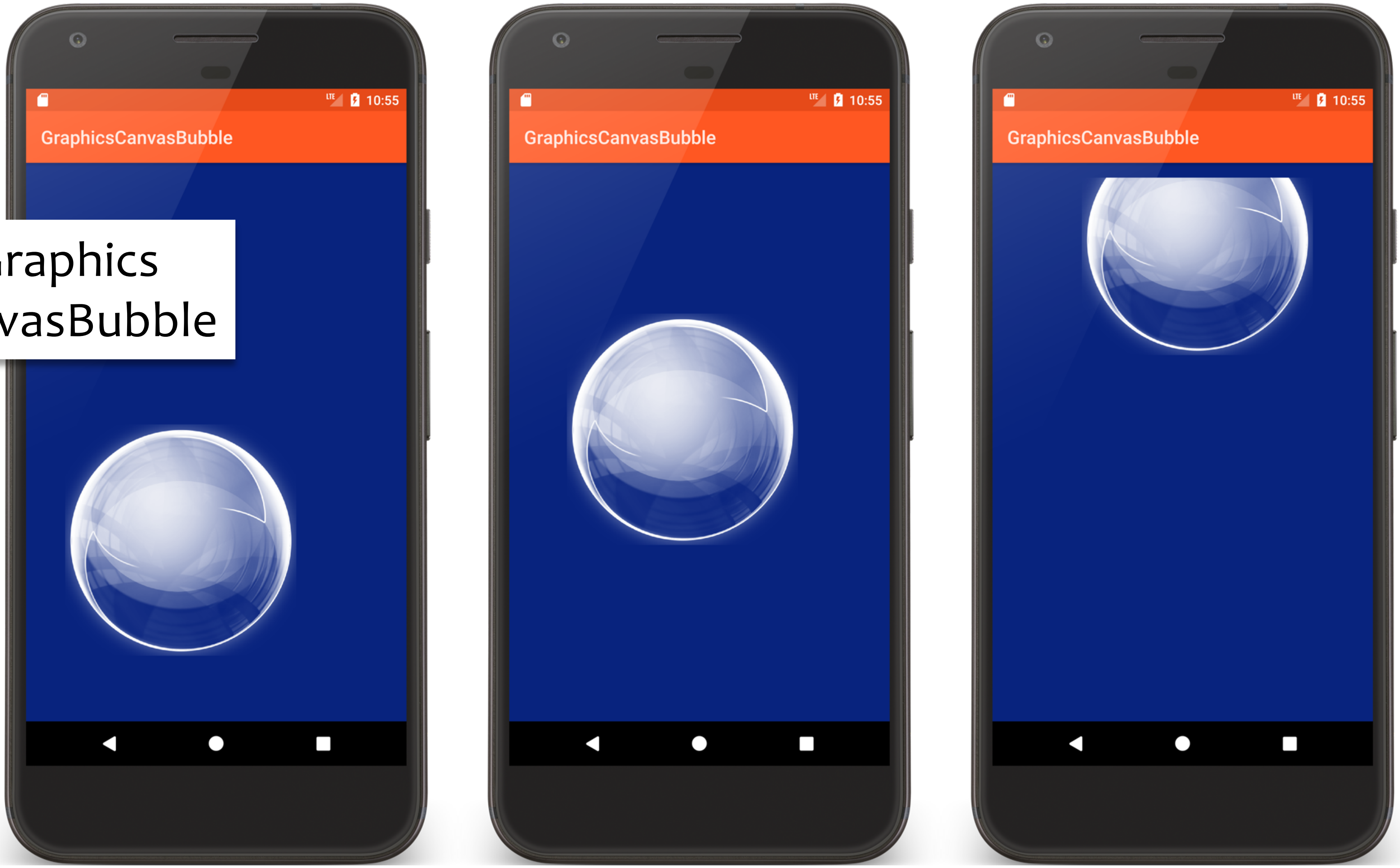
Application provides its own Canvas and has greater control over drawing

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



BubbleActivity.kt

```
public override fun onCreate(savedInstanceState: Bundle?) {
    …
    val frame = findViewById<RelativeLayout>(R.id.frame)
    val bitmap = BitmapFactory.decodeResource(resources,R.drawable.b512)
    val bubbleView = BubbleView(applicationContext,bitmap)
    frame.addView(bubbleView)
    Thread {
        while (bubbleView.moveWhileOnscreen()) {
            try {
                Thread.sleep(25)
            } catch (e: InterruptedException) {
                Log.i(TAG, "InterruptedException")
            }
            bubbleView.postInvalidate()
        }
    }.start()
}
```

BubbleActivity.kt

```
private inner class BubbleView
   internal constructor(context: Context, bitmap: Bitmap) : View(context) {
    ..
    init {
        // Scale bitmap
        this.mBitmap = Bitmap.createScaledBitmap(bitmap,
                 mBitmapWidthAndHeight, mBitmapWidthAndHeight, false)
        mBitmapWidthAndHeightAdj = mBitmapWidthAndHeight + margin
        // Set useful parameters
         mBitmapWidthAndHeightAdj = mBitmapHeightAndWidth / 2
         mBitmapHeightAndWidthPadded = mBitmapHeightAndWidth +
                                                           mBitmapPadding
         // Determine screen size
         mDisplayWidth = windowManager.currentWindowMetrics.bounds.width()
         mDisplayHeight = windowManager.currentWindowMetrics.bounds.height()
```

BubbleActivity.kt

```
    // Set random starting point
    val r = Random()
    val x = r.nextInt(mDisplayWidth - mBitmapWidthAndHeight).toFloat()
    val y = r.nextInt(mDisplayHeight - mBitmapWidthAndHeight).toFloat()
    mCurrentCoords = Coords(x, y)
    // Set random movement direction and speed
    var dy = max(r.nextFloat(), 0.1f) * stepSize
    dy *= (if (r.nextInt(2) == 1) 1 else -1).toFloat()
    var dx = max(r.nextFloat(), 0.1f) * stepSize
    dx *= (if (r.nextInt(2) == 1) 1 else -1).toFloat()
    mDxDy = Coords(dx, dy)
    // Add some painting directives
    mPainter.isAntiAlias = true
    mPainter.colorFilter =
        PorterDuffColorFilter(Color.WHITE, PorterDuff.Mode.SRC_ATOP)
}
```

BubbleActivity.kt

```
    override fun onDraw(canvas: Canvas) {
        val tmp = mCurrentCoords.coords
        canvas.drawBitmap(mBitmap, tmp.mX, tmp.mY, mPainter)
    }

    internal fun moveWhileOnscreen(): Boolean {
        mCurrentCoords = mCurrentCoords.move(mDxDy)
        return !(mCurrentCoords.mY < 0 - mBitmapWidthAndHeightAdj
             || mCurrentCoords.mY > mDisplayHeight + mBitmapWidthAndHeightAdj
             || mCurrentCoords.mX < 0 - mBitmapWidthAndHeightAdj
             || mCurrentCoords.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()

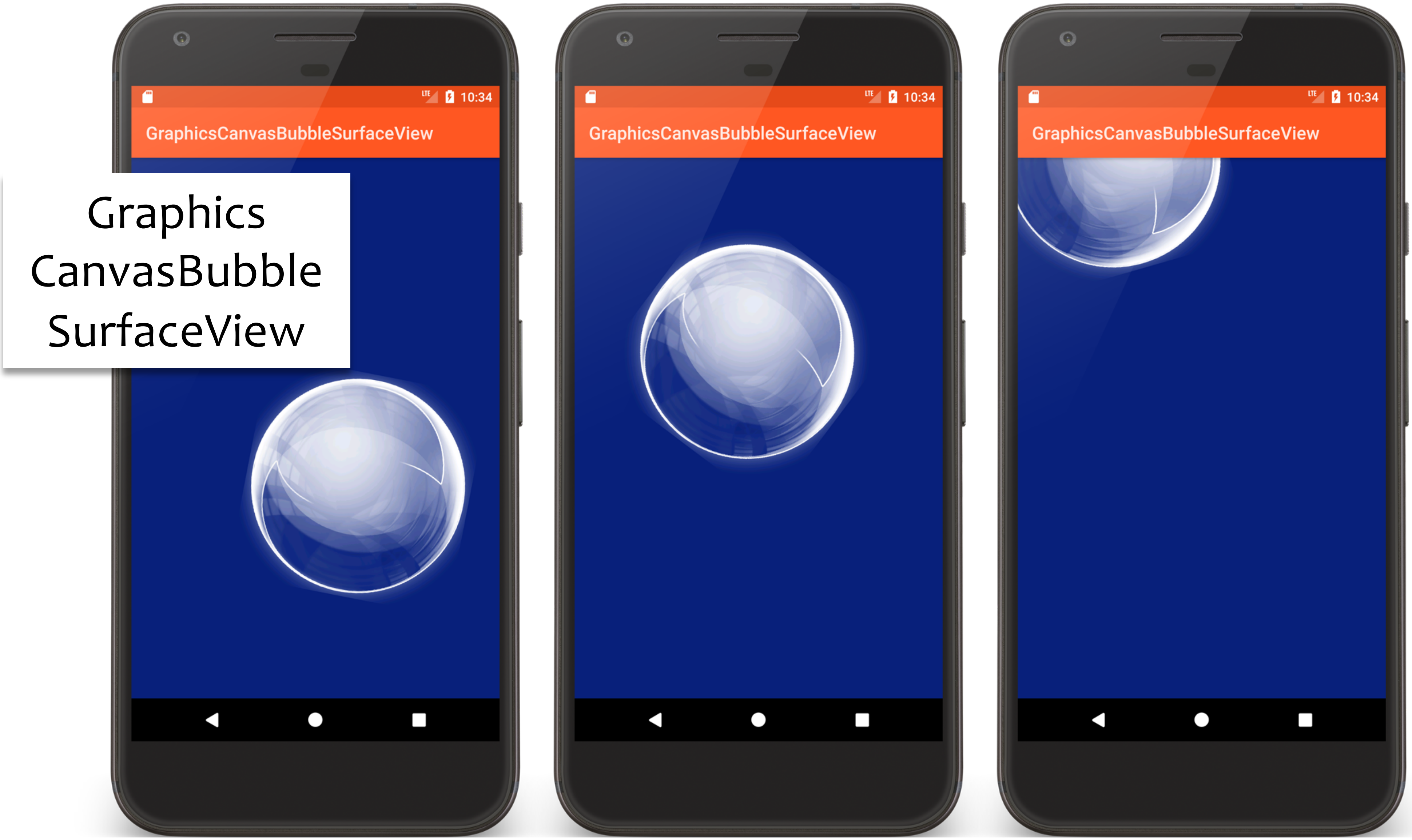
surfaceDestroyed()

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
GraphicsCanvasBubbleSurfaceView
10:34
GraphicsCanvasBubbleSurfaceView
10:34
GraphicsCanvasBubbleSurfaceView
10:34

BubbleActivity.kt

```kotlin
public override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.main)

    val relativeLayout = findViewById<RelativeLayout>(R.id.frame)
    val bubbleView = BubbleView(applicationContext,
        BitmapFactory.decodeResource(resources, R.drawable.b512))
    relativeLayout.addView(bubbleView)
}
```

BubbleActivity.kt

```
private inner class BubbleView internal constructor(context: Context,
bitmap: Bitmap) : SurfaceView(context), SurfaceHolder.Callback {
    init {
        // Resize bubble
        this.mBitmap = Bitmap.createScaledBitmap(bitmap,
                 mBitmapHeightAndWidth, mBitmapHeightAndWidth, false)
        // Set useful parameters
        mBitmapWidthAndHeightAdj = mBitmapHeightAndWidth / 2
        mBitmapHeightAndWidthPadded = mBitmapHeightAndWidth + mBitmapPadding
        // Determine screen size
         mDisplayWidth = windowManager.currentWindowMetrics.bounds.width()
         mDisplayHeight = windowManager.currentWindowMetrics.bounds.height()
```

BubbleActivity.kt

```kotlin
// Set random starting point for BubbleView
val r = Random()
val x = r.nextInt(mDisplayWidth / 4).toFloat() + mDisplayWidth / 4
val y = r.nextInt(mDisplayHeight / 4).toFloat() + mDisplayHeight / 4
mCurrent = Coords(x, y)

// Set random movement direction and speed, and set rotation
var dy = max(r.nextFloat(), 0.1f) * mStep
dy *= (if (r.nextInt(2) == 1) 1 else -1).toFloat()
var dx = max(r.nextFloat(), 0.1f) * mStep
dx *= (if (r.nextInt(2) == 1) 1 else -1).toFloat()
mDxDy = Coords(dx, dy)
mRotation = 1.0f
```

BubbleActivity.kt

```
        mPainter.isAntiAlias = true
        mPainter.colorFilter = PorterDuffColorFilter(Color.WHITE,
                                                     PorterDuff.Mode.SRC_ATOP)
        // Prepare surface for drawing
        mSurfaceHolder = holder
        mSurfaceHolder.addCallback(this)
    }

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

BubbleActivity.kt

```kotlin
private fun moveWhileOnscreen(): Boolean {
    mCurrent = mCurrent.move(mDxDy)
    return !(mCurrent.mY < 0 - mBitmapHeightAndWidthPadded
            || mCurrent.mY > mDisplayHeight + mBitmapHeightAndWidthPadded
            || mCurrent.mX < 0 - mBitmapHeightAndWidthPadded
            || mCurrent.mX > mDisplayWidth + mBitmapHeightAndWidthPadded)
}
```

BubbleActivity.kt

```
override fun surfaceCreated(holder: SurfaceHolder) {
        mDrawingThread = Thread(Runnable {
         while (!Thread.currentThread().isInterrupted && moveWhileOnscreen()) {
                val canvas = mSurfaceHolder.lockCanvas()
                if (null != canvas) {
                    drawBubble(canvas)
                    mSurfaceHolder.unlockCanvasAndPost(canvas)
                }
            }
        })
        mDrawingThread?.start()
    }
override fun surfaceDestroyed(holder: SurfaceHolder) {
        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

TransitionDrawableActivity.kt

```
public override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.main)

    val transition = resources.getDrawable(R.drawable.shape_transition,
                                           null) as TransitionDrawable
    transition.isCrossFadeEnabled = true
    (findViewById<ImageView>(R.id.image_view)).setImageDrawable(transition)
    transition.startTransition(5000)
}
```

shape_transition.xml

```
<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

view_animation.xml

```xml
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="true" >
    <item
        android:drawable="@drawable/empty_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>
```

GraphicsFrameAnimationActivity.kt

```
public override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.main)

    val imageView = findViewById<ImageView>(R.id.countdown_frame)
    imageView.setBackgroundResource(R.drawable.view_animation)
    mAnim = imageView.background as AnimationDrawable
}
```

GraphicsFrameAnimationActivity.kt

```
override fun onPause() {
    super.onPause()
    if (mAnim.isRunning) {
        mAnim.stop()
    }
}

override fun onWindowFocusChanged(hasFocus: Boolean) {
    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

view_animation.xml

```
<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" />

    …
```

view_animation.xml

```
    <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>
```

GraphicsTweenAnimationActivity.kt

```kotlin
public override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.main)

    mImageView = findViewById(R.id.icon)
    mAnim = AnimationUtils.loadAnimation(this, R.anim.view_animation)
}

override fun onWindowFocusChanged(hasFocus: Boolean) {
    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

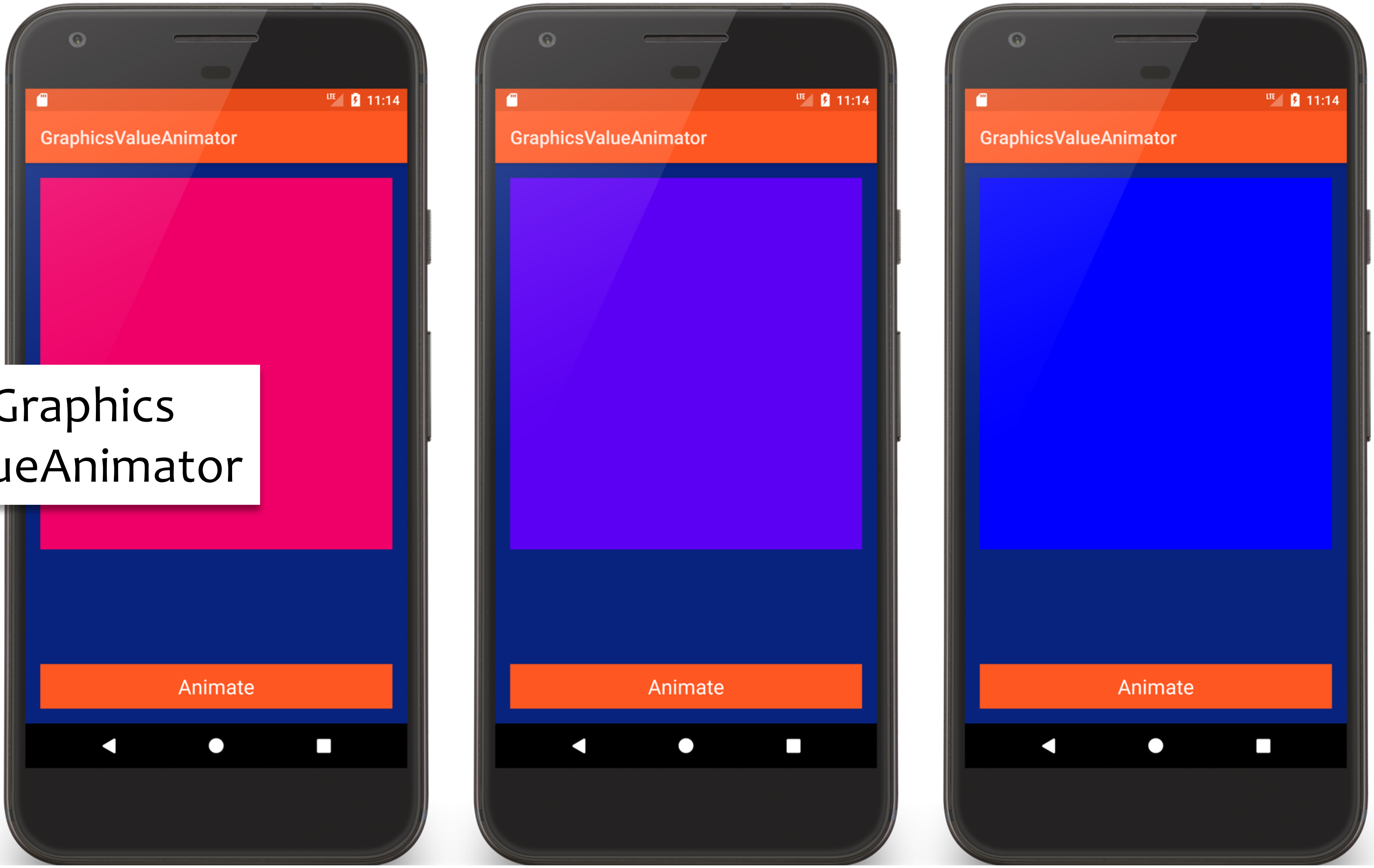
Property Animation Architecture

AnimatorSet – combines individual animations to create more complex animations

GraphicsValueAnimator

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

Graphics ValueAnimator



ValueAnimatorActivity.kt

```kotlin
class ValueAnimatorActivity : Activity() {

    companion object {
        private const val RED = Color.RED
        private const val BLUE = Color.BLUE
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.main)
    }

    fun onClick(v: View) {
        startAnimation()
    }
```

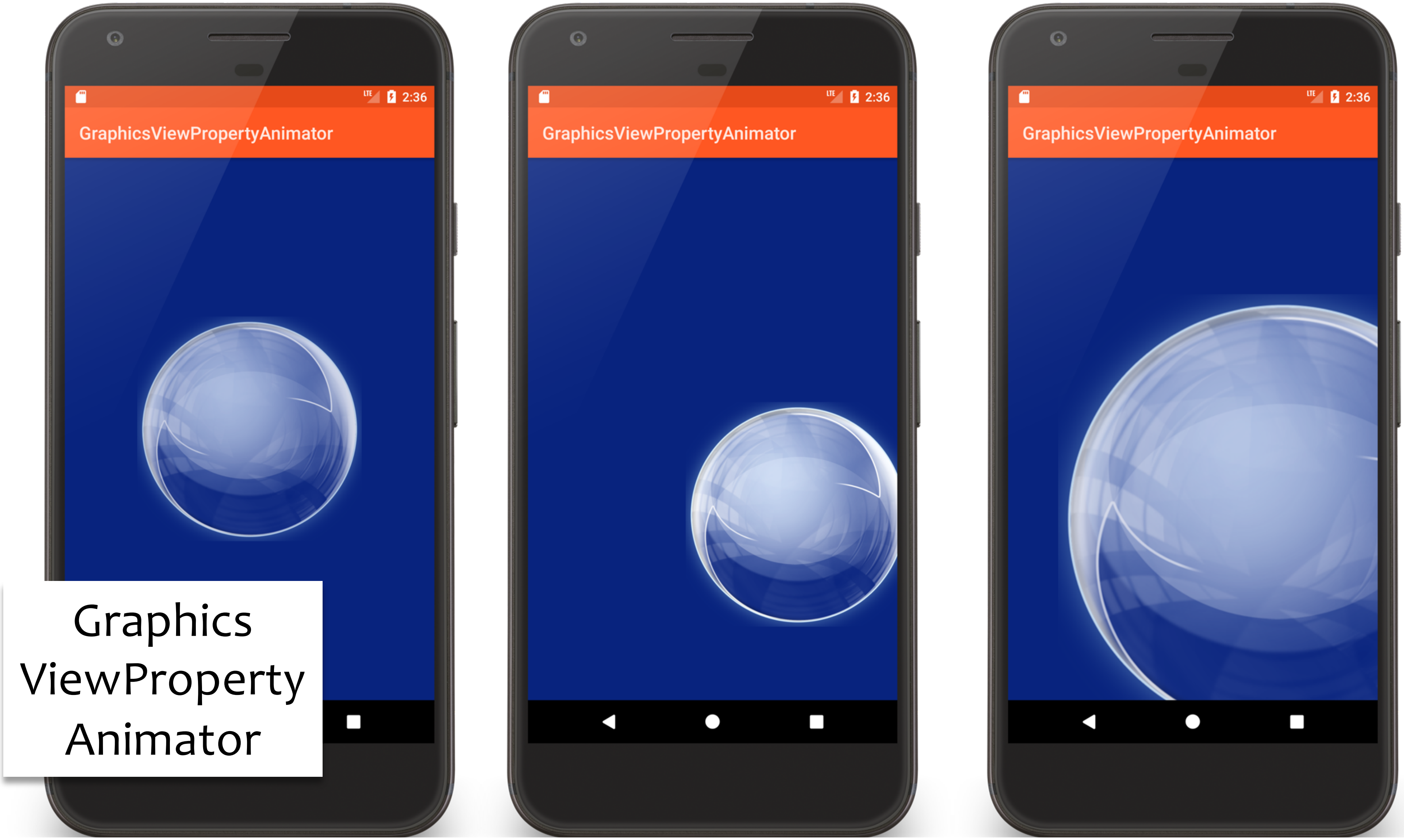
ValueAnimatorActivity.kt

```kotlin
private fun startAnimation() {

    val imageView = findViewById<ImageView>(R.id.image_view)

    val anim = ValueAnimator.ofObject(ArgbEvaluator(),RED, BLUE)

    anim.addUpdateListener { animation ->
        imageView.setBackgroundColor(
            animation.animatedValue as Int
        )
    }

    anim.duration = 10000
    anim.start()
}
```

GraphicsViewPropertyAnimator

Same as the GraphicsTweenAnimation

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


2:36
GraphicsViewPropertyAnimator
2:36
GraphicsViewPropertyAnimator
2:36
GraphicsViewPropertyAnimator
Graphics
ViewProperty
Animator

GraphicsViewPropertyAnimatorActivity.kt

```
override fun onWindowFocusChanged(hasFocus: Boolean) {
    super.onWindowFocusChanged(hasFocus)
    mImageView = findViewById(R.id.bubble)
    if (hasFocus) {
        fadeIn.run()
    }
}

private val fadeIn = Runnable {
    mImageView.animate().setDuration(3000)
        .setInterpolator(LinearInterpolator()).alpha(1.0f)
        .withEndAction(rotate)
}
```

GraphicsViewPropertyAnimatorActivity.kt

```kotlin
private val rotate = Runnable {
    mImageView.animate().setDuration(4000)
        .setInterpolator(AccelerateInterpolator())
        .rotationBy(720.0f).withEndAction(translate)
}

private val translate = Runnable {
    val translation = resources.getDimension(R.dimen.translation)
    mImageView.animate().setDuration(3000)
        .setInterpolator(OvershootInterpolator())
        .translationXBy(translation).translationYBy(translation)
        .withEndAction(scale)
}
```

GraphicsViewPropertyAnimatorActivity.kt

```
private val scale = Runnable {
    mImageView.animate().setDuration(3000)
        .setInterpolator(AnticipateInterpolator())
        .scaleXBy(1.0f).scaleYBy(1.0f).withEndAction(fadeOut)
}

private val fadeOut = Runnable {
    mImageView.animate().setDuration(2000)
        .setInterpolator(DecelerateInterpolator()).alpha(0.0f)
}
```

Next Time

MultiTouch & Gestures

Example Applications

GraphicsBubbleXML

GraphicsBubbleProgram

GraphicsShapeDrawXML

GraphicsShapeDraw

GraphicsPaint

GraphicsCanvasBubble

GraphicsCanvas
BubbleSurfaceView

GraphicsTransitionDrawable

GraphicsFrameAnimation

GraphicsTweenAnimation

GraphicsValueAnimator

GraphicsView
PropertyAnimator