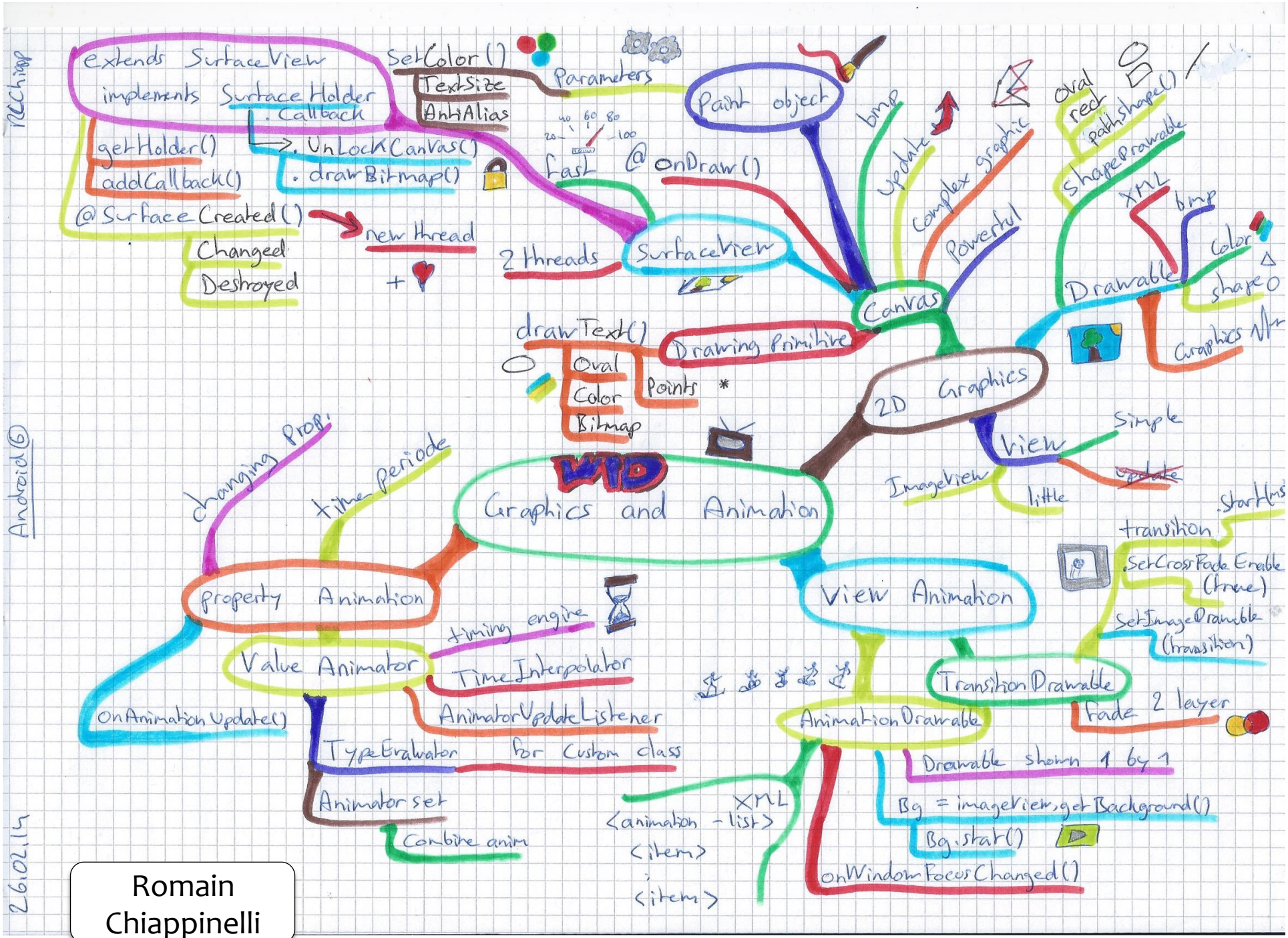


Romain Chiappinelli

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


16:01
GraphicsBubbleXML

GraphicsBubbleXML

```
<ImageView
    android:id="@+id/imageView1"
    android:layout_width="250dp"
    android:layout_height="250dp"
    android:layout_centerInParent="true"
    android:contentDescription="@string/bubble_desc"
    android:src="@drawable/b128" />
```

GraphicsBubbleProgram

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

        RelativeLayout relativeLayout = (RelativeLayout) findViewById(R.id.frame);

        ImageView bubbleView = new ImageView(getApplicationContext());
        bubbleView
                .setImageDrawable(getResources().getDrawable(R.drawable.b128));

        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

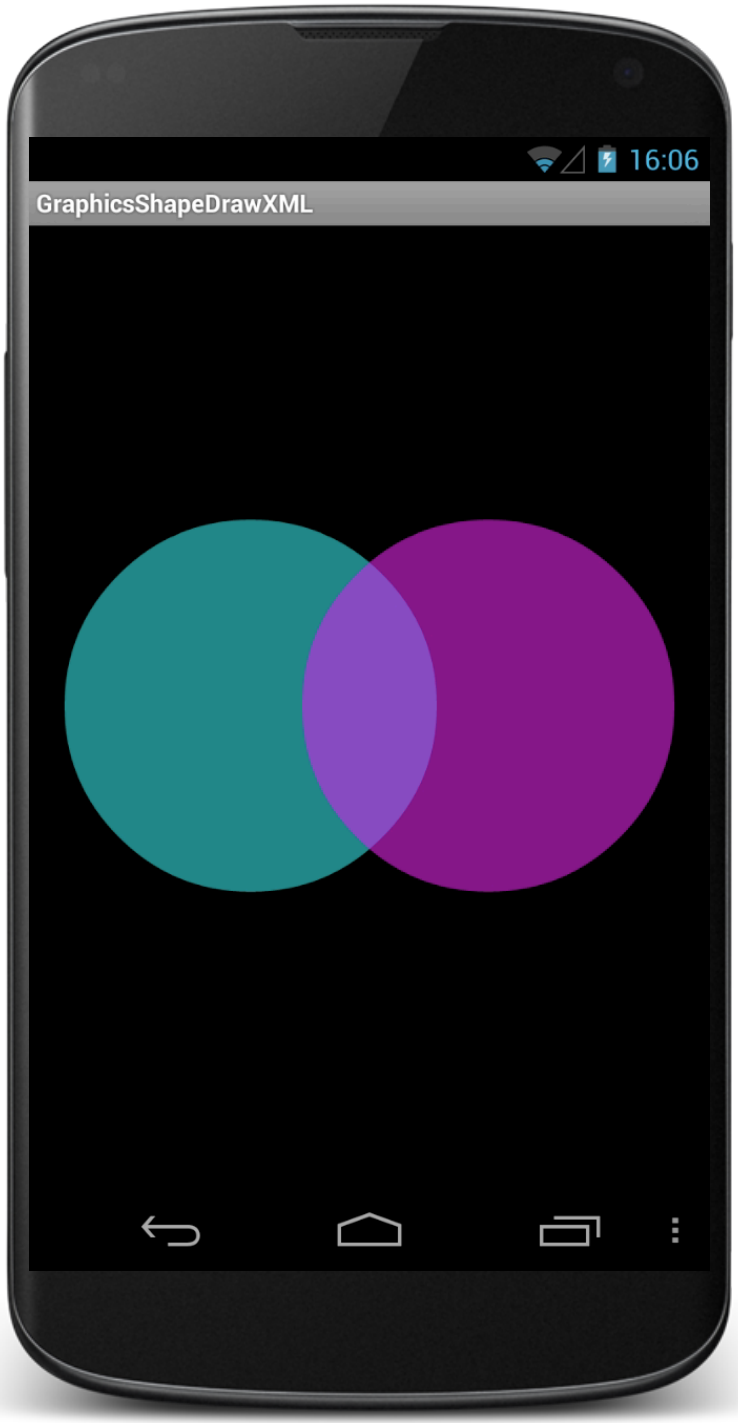
RectShape – rectangles

OvalShape – ovals & rings

GraphicsShapeDraw

Applications display two Shapes within a RelativeLayout

The two shapes are partially overlapping and semi-transparent


16:06
GraphicsShapeDrawXML

GraphicsShapeDraw

```
public class ShapeDrawActivity extends Activity {
    int alpha = 127;

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

        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);
```

GraphicsShapeDraw

```
// 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);
```

GraphicsShapeDraw

```
// 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()

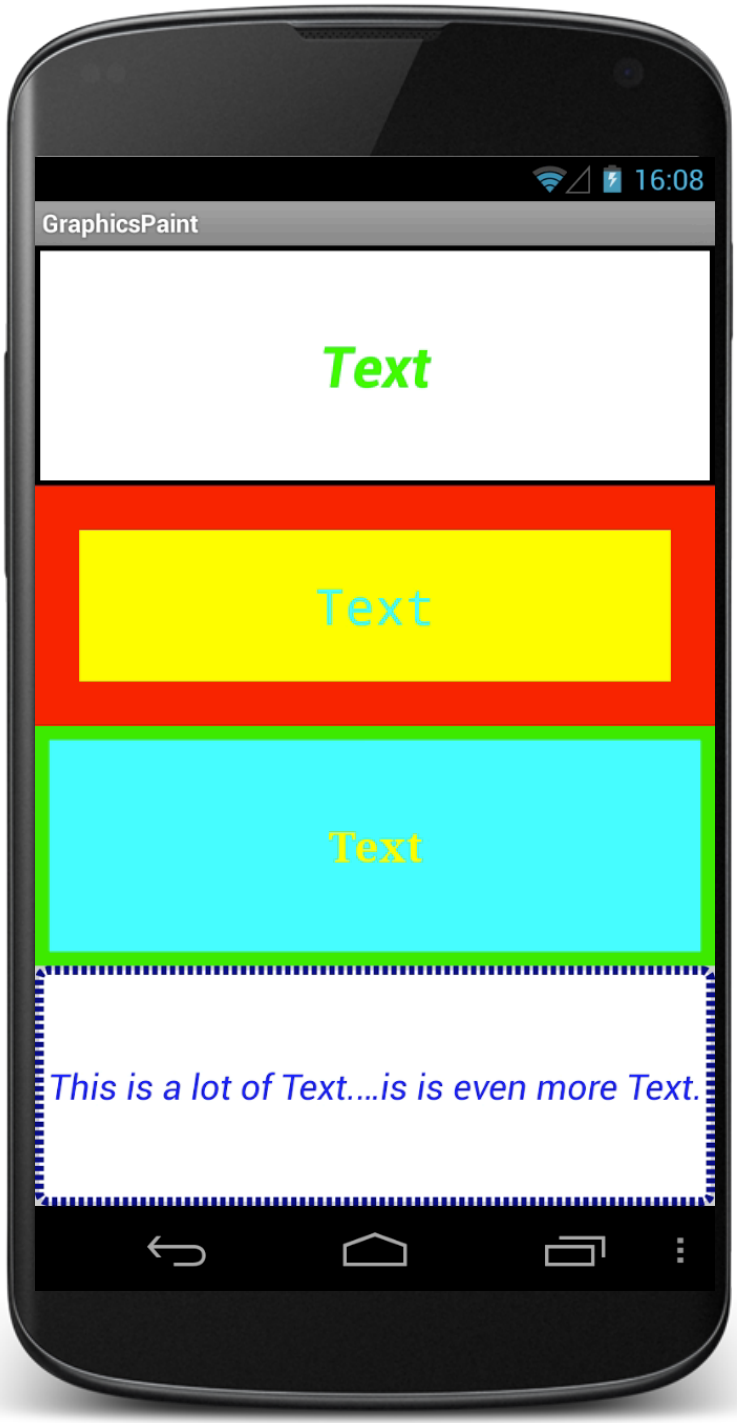
setTextSize()

setColor()

setAntiAlias()

GraphicsPaint

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


16:08
GraphicsPaint
Text
Text
Text
This is a lot of Text....is is even more Text.

GraphicsPaint

```
<TextView
    android:id="@+id/textView1"
    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="32sp"
    android:textStyle="bold|italic"
    android:typeface="normal" />
```

GraphicsPaint

```
<TextView
    android:id="@+id/imageView2"
    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="28sp"
    android:textStyle="normal"
    android:typeface="monospace" />
```

GraphicsPaint

```
<TextView
    android:id="@+id/imageView3"
    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" />
```

GraphicsPaint

```
<TextView
    android:id="@+id/imageView4"
    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="20sp"
    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 to the View when it calls the View's onDraw() method

Drawing to SurfaceViews

Create a Custom SurfaceView

Provide secondary thread for drawing

Application provides its own canvas and has greater control over drawing

GraphicsBubble

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


16:12
GraphicsCanvasBubble

GraphicsCanvasBubble

```
public class BubbleActivity extends Activity {
    protected static final String TAG = "BubbleActivity";

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

        final RelativeLayout frame = (RelativeLayout) findViewById(R.id.frame);
        final Bitmap bitmap = BitmapFactory.decodeResource(getResources(),
                R.drawable.b128);
        final BubbleView bubbleView = new BubbleView(getApplicationContext(),
                bitmap);

        frame.addView(bubbleView);

        new Thread(new Runnable() {
            @Override
            public void run() {
                while (bubbleView.move()) {
                    bubbleView.postInvalidate();
                    try {
                        Thread.sleep(1000);
                    } catch (InterruptedException e) {
                        Log.i(TAG, "InterruptedException");
                    }

                }
            }
        }).start();
    }
```

GraphicsCanvasBubble

```
private class BubbleView extends View {

    private static final int STEP = 100;
    final private Bitmap mBitmap;

    private Coords mCurrent;
    final private Coords mDxDy;

    final private DisplayMetrics mDisplayMetrics;
    final private int mDisplayWidth;
    final private int mDisplayHeight;
    final private int mBitmapWidthAndHeight, mBitmapWidthAndHeightAdj;
    final private Paint mPainter = new Paint();
```

GraphicsCanvasBubble

```
public BubbleView(Context context, Bitmap bitmap) {
    super(context);

    mBitmapWidthAndHeight = (int) getResources().getDimension(
            R.dimen.image_height);
    this.mBitmap = Bitmap.createScaledBitmap(bitmap,
            mBitmapWidthAndHeight, mBitmapWidthAndHeight, false);

    mBitmapWidthAndHeightAdj = mBitmapWidthAndHeight + 20;

    mDisplayMetrics = new DisplayMetrics();
    BubbleActivity.this.getWindowManager().getDefaultDisplay()
            .getMetrics(mDisplayMetrics);
    mDisplayWidth = mDisplayMetrics.widthPixels;
    mDisplayHeight = mDisplayMetrics.heightPixels;
```

GraphicsCanvasBubble

```
        Random r = new Random();
        float x = (float) r.nextInt(mDisplayWidth);
        float y = (float) r.nextInt(mDisplayHeight);
        mCurrent = new Coords(x, y);

        float dy = (float) r.nextInt(mDisplayHeight) / mDisplayHeight;
        dy *= r.nextInt(2) == 1 ? STEP : -1 * STEP;
        float dx = (float) r.nextInt(mDisplayWidth) / mDisplayWidth;
        dx *= r.nextInt(2) == 1 ? STEP : -1 * STEP;
        mDxDy = new Coords(dx, dy);

        mPainter.setAntiAlias(true);

    }
```

GraphicsCanvasBubble

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

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

    if (mCurrent.mY < 0 - mBitmapWidthAndHeightAdj
            || mCurrent.mY > mDisplayHeight + mBitmapWidthAndHeightAdj
            || mCurrent.mX < 0 - mBitmapWidthAndHeightAdj
            || mCurrent.mX > mDisplayWidth + mBitmapWidthAndHeightAdj) {
        return false;
    } else {
        return true;
    }
}
```

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()


17:47
GraphicsCanvasBubbleSurfaceView

GraphicsCanvasBubbleSurfaceView

```
public class BubbleActivity extends Activity {

    BubbleView mBubbleView;

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

        RelativeLayout relativeLayout = (RelativeLayout) findViewById(R.id.frame);
        final BubbleView bubbleView = new BubbleView(getApplicationContext(),
                BitmapFactory.decodeResource(getResources(), R.drawable.b128));

        relativeLayout.addView(bubbleView);
    }
```

GraphicsCanvasBubbleSurfaceView

```
private class BubbleView extends SurfaceView implements
        SurfaceHolder.Callback {

    private final Bitmap mBitmap;
    private final int mBitmapHeightAndWidth, mBitmapHeightAndWidthAdj;
    private final DisplayMetrics mDisplay;
    private final int mDisplayWidth, mDisplayHeight;
    private float mX, mY, mDx, mDy, mRotation;
    private final SurfaceHolder mSurfaceHolder;
    private final Paint mPainter = new Paint();
    private Thread mDrawingThread;

    private static final int MOVE_STEP = 1;
    private static final float ROT_STEP = 1.0f;
```

GraphicsCanvasBubbleSurfaceView

```
public BubbleView(Context context, Bitmap bitmap) {
    super(context);

    mBitmapHeightAndWidth = (int) getResources().getDimension(
            R.dimen.image_height_width);
    this.mBitmap = Bitmap.createScaledBitmap(bitmap,
            mBitmapHeightAndWidth, mBitmapHeightAndWidth, false);

    mBitmapHeightAndWidthAdj = mBitmapHeightAndWidth / 2;

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

GraphicsCanvasBubbleSurfaceView

```
        Random r = new Random();
        mX = (float) r.nextInt(mDisplayHeight);
        mY = (float) r.nextInt(mDisplayWidth);
        mDx = (float) r.nextInt(mDisplayHeight) / mDisplayHeight;
        mDx *= r.nextInt(2) == 1 ? MOVE_STEP : -1 * MOVE_STEP;
        mDy = (float) r.nextInt(mDisplayWidth) / mDisplayWidth;
        mDy *= r.nextInt(2) == 1 ? MOVE_STEP : -1 * MOVE_STEP;
        mRotation = 1.0f;

        mPainter.setAntiAlias(true);

        mSurfaceHolder = getHolder();
        mSurfaceHolder.addCallback(this);
    }
```

GraphicsCanvasBubbleSurfaceView

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

private boolean move() {
    mX += mDx;
    mY += mDy;
    if (mX < 0 - mBitmapHeightAndWidth
            || mX > mDisplayHeight + mBitmapHeightAndWidth
            || mY < 0 - mBitmapHeightAndWidth
            || mY > mDisplayWidth + mBitmapHeightAndWidth) {
        return false;
    } else {
        return true;
    }
}

@Override
public void surfaceChanged(SurfaceHolder holder, int format, int width,
        int height) {
}
```

GraphicsCanvasBubbleSurfaceView

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

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

View Animation

Changing the properties of a View over a period of time

Size

Position

Transparency

Orientation

View Animation Classes

TransitionDrawable

AnimationDrawable

Animation

TransitionDrawable

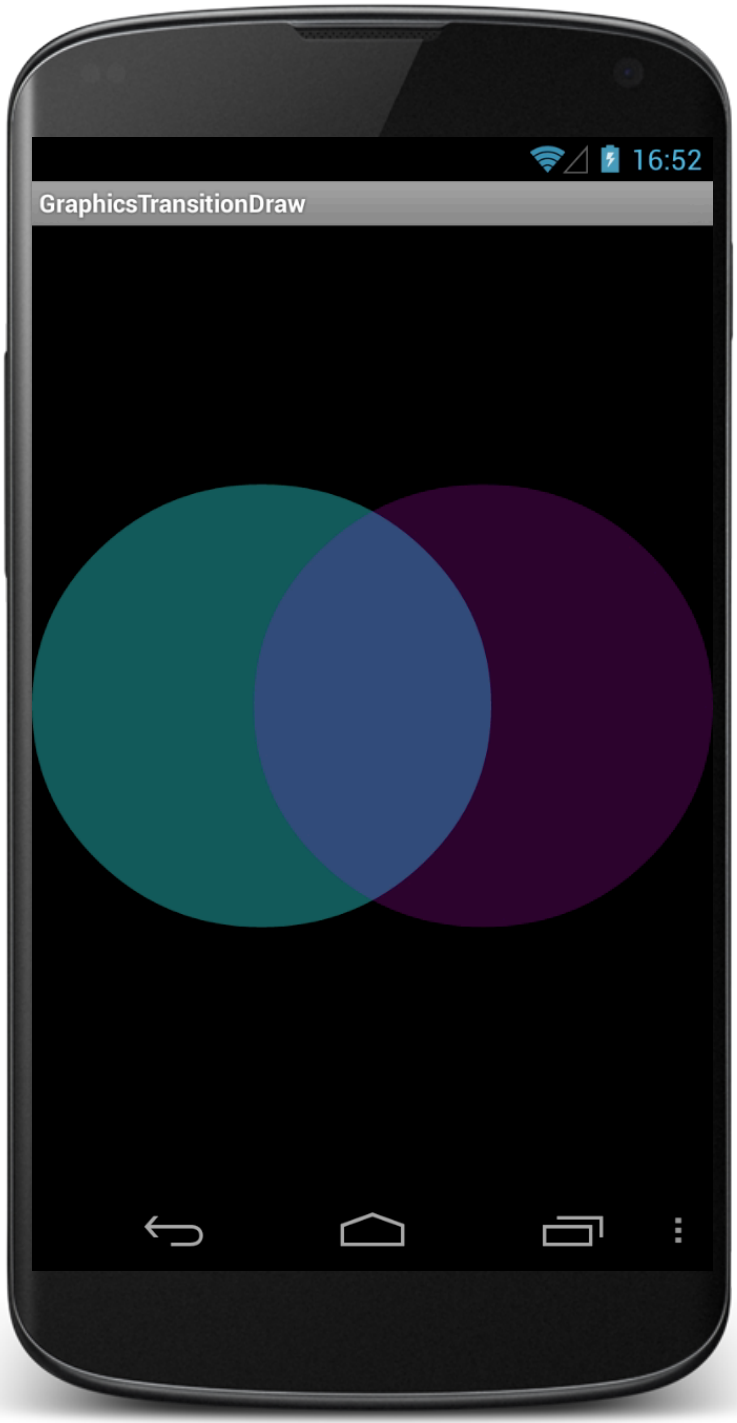
A 2-layer Drawable

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


16:52
GraphicsTransitionDraw

GraphicsTransitionDrawable

```
<ImageView
    android:id="@+id/image_view"
    android:layout_width="match_parent"
    android:layout_height="250dp"
    android:contentDescription="@string/fading_image_desc"
    android:layout_centerVertical="true"
```

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

    setContentView(R.layout.main);

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

    transition.setCrossFadeEnabled(true);

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

    transition.startTransition(5000);
}
```

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


16:54
GraphicsFrameAnimation

GraphicsFrameAnimation

```
<ImageView
    android:id="@+id/countdown_frame"
    android:layout_width="300dp"
    android:layout_height="250dip"
    android:layout_gravity="center"
    android:layout_marginBottom="20dp"
    android:layout_marginTop="20dp"
    android:contentDescription="@string/animation_desc"
    android:scaleType="centerCrop" />
```

GraphicsFrameAnimation

```
private AnimationDrawable mAnim;

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

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

    imageView.setBackgroundResource(R.drawable.view_animation);

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

GraphicsFrameAnimation

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


@Override
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


13:26
GraphicsTweenAnimation

GraphicsTweenAnimation

```
<ImageView
    android:id="@+id/icon"
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:src="@drawable/b128"
    android:contentDescription="@string/bubble_desc"
    android:visibility="invisible"/>
```

GraphicsTweenAnimation

```
public class GraphicsTweenAnimationActivity extends Activity {

    private ImageView mImageView;
    private Animation mAnim;

    @Override
    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);

    }
```

GraphicsTweenAnimation

```
    @Override
    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

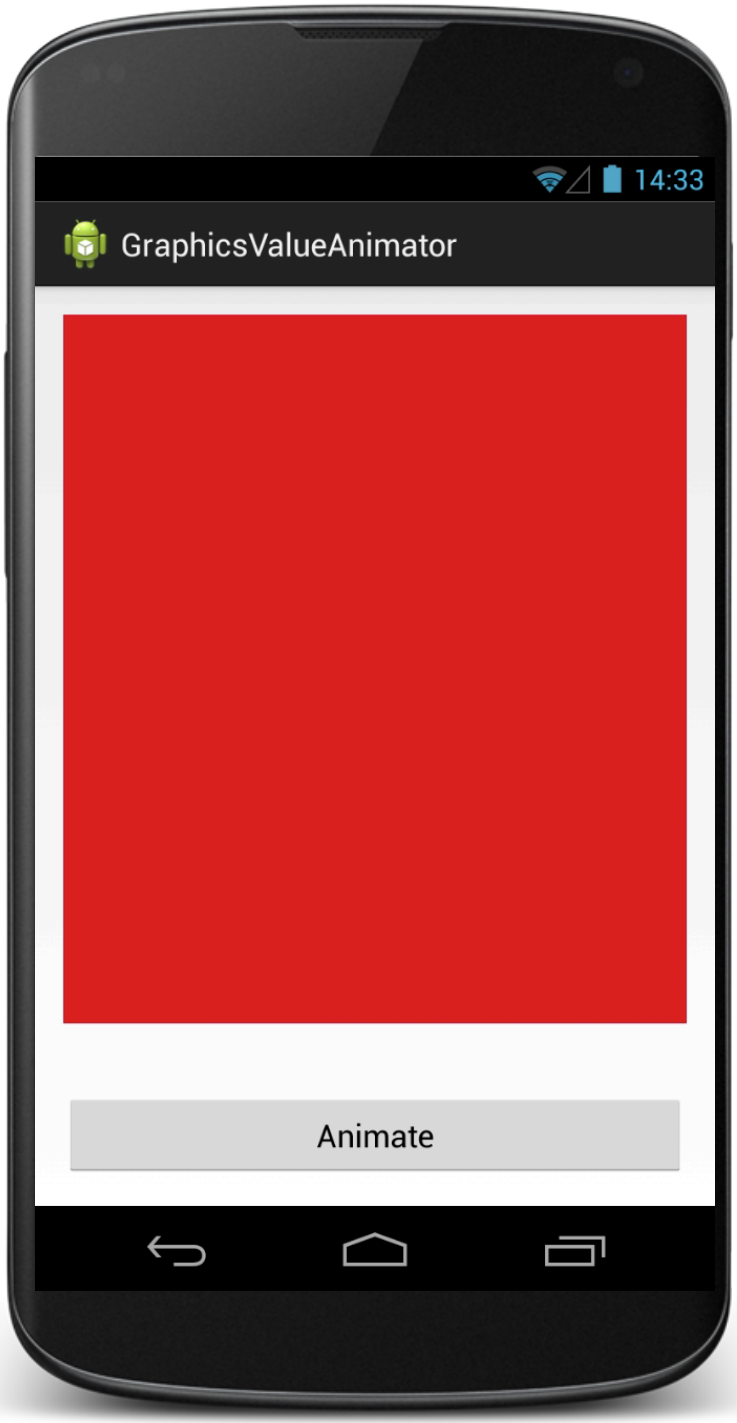
TypeEvaluator – Calculates a property's value at a given point in time

Property Animation Architecture

AnimatorSet – combines individual animations to create more complex animations

GraphicsValueAnimator

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


14:33
GraphicsValueAnimator
Animate

GraphicsValueAnimator

```
<ImageView
    android:id="@+id/image_view"
    android:layout_width="match_parent"
    android:layout_height="400dp"
    android:contentDescription="@string/app_name" />

<Button
    android:id="@+id/start_animation_button"
    android:layout_width="match_parent"
    android:layout_height="48dp"
    android:layout_alignParentBottom="true"
    android:text="@string/button_label"/>
```

GraphicsValueAnimator

```
public class ValueAnimatorActivity extends Activity {

    protected static final String TAG = "ValueAnimatorActivity";
    final private static int RED = Color.RED;
    final private static int BLUE = Color.BLUE;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button startButton = (Button) findViewById(R.id.start_animation_button);
        startButton.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                startAnimation();
            }
        });

    }
```

GraphicsValueAnimator

```
public void startAnimation() {

    final ImageView imageView = (ImageView) findViewById(R.id.image_view);

    ValueAnimator anim = ValueAnimator.ofObject(new ArgbEvaluator(), RED,
            BLUE);

    anim.addUpdateListener(new AnimatorUpdateListener() {

        @Override
        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


17:08
GraphicsViewPropertyAnimator

GraphicsViewPropertyAnimator

```
<ImageView
    android:id="@+id/icon"
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:src="@drawable/b128"
    android:alpha="0"
    android:contentDescription="@string/bubble_desc"/>
```

GraphicsViewPropertyAnimator

```
public class GraphicsViewPropertyAnimatorActivity extends Activity {

    private ImageView mImageView;

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

    }

    @Override
    public void onWindowFocusChanged(boolean hasFocus) {
        super.onWindowFocusChanged(hasFocus);

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

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

GraphicsViewPropertyAnimator

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

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

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);
    }
};
```

GraphicsViewPropertyAnimator

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

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

Next Time

MultiTouch & Gestures