CS371m - Mobile Computing

2D Graphics

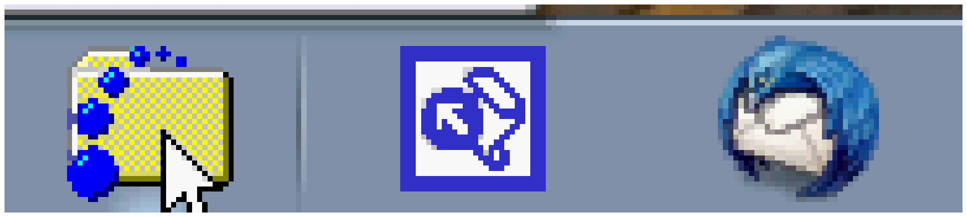
A Crash Course in Using (Android) 2D Graphics Libraries

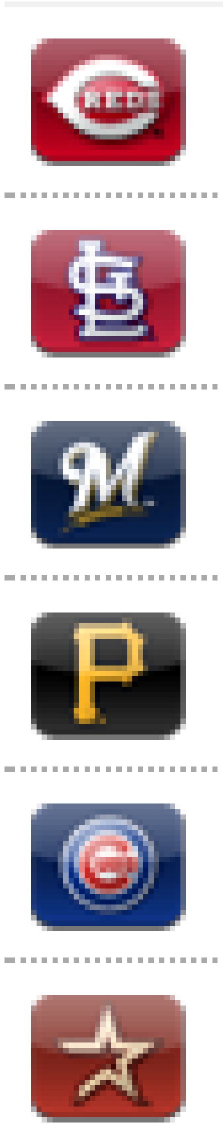
Using Graphics

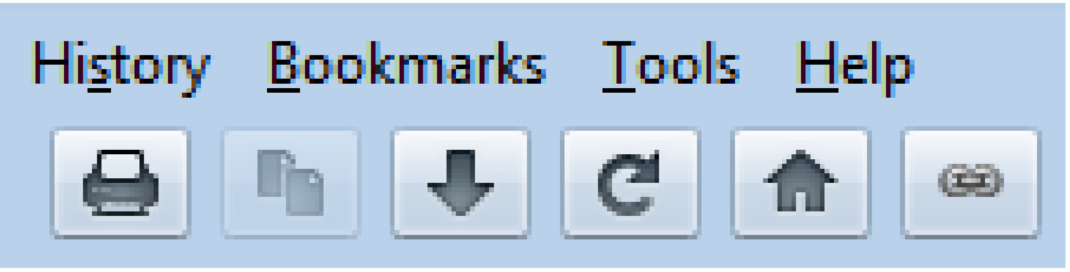
- Not an end in itself
- Create a richer, easier to use User Interface
- Display information in a easier to understand way
- Entertainment
- In Android picking the **theme** affects the appearance of your GUI widgets

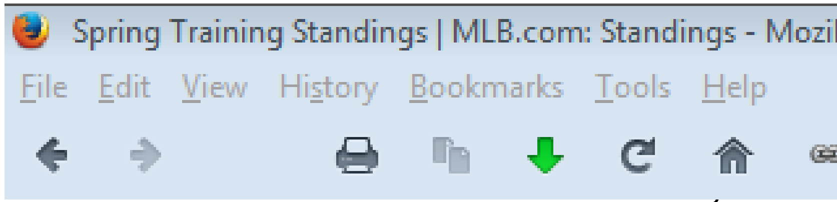
Richer UI

- The (OLD) Apple Effect?



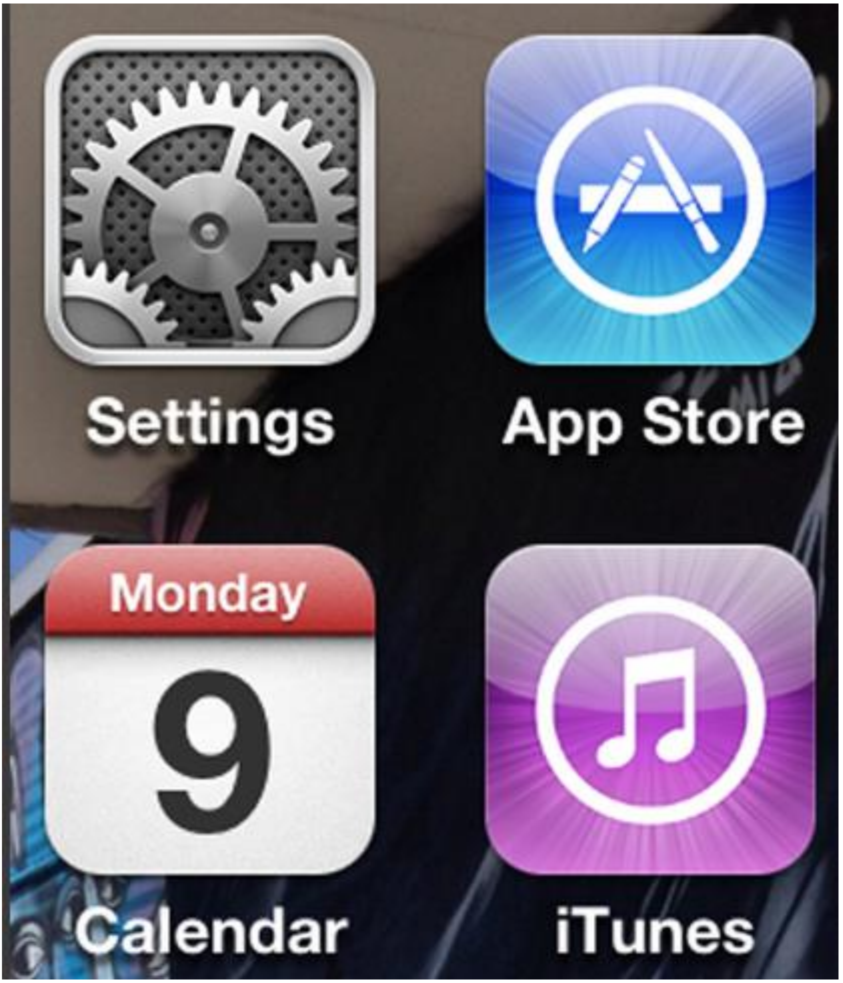


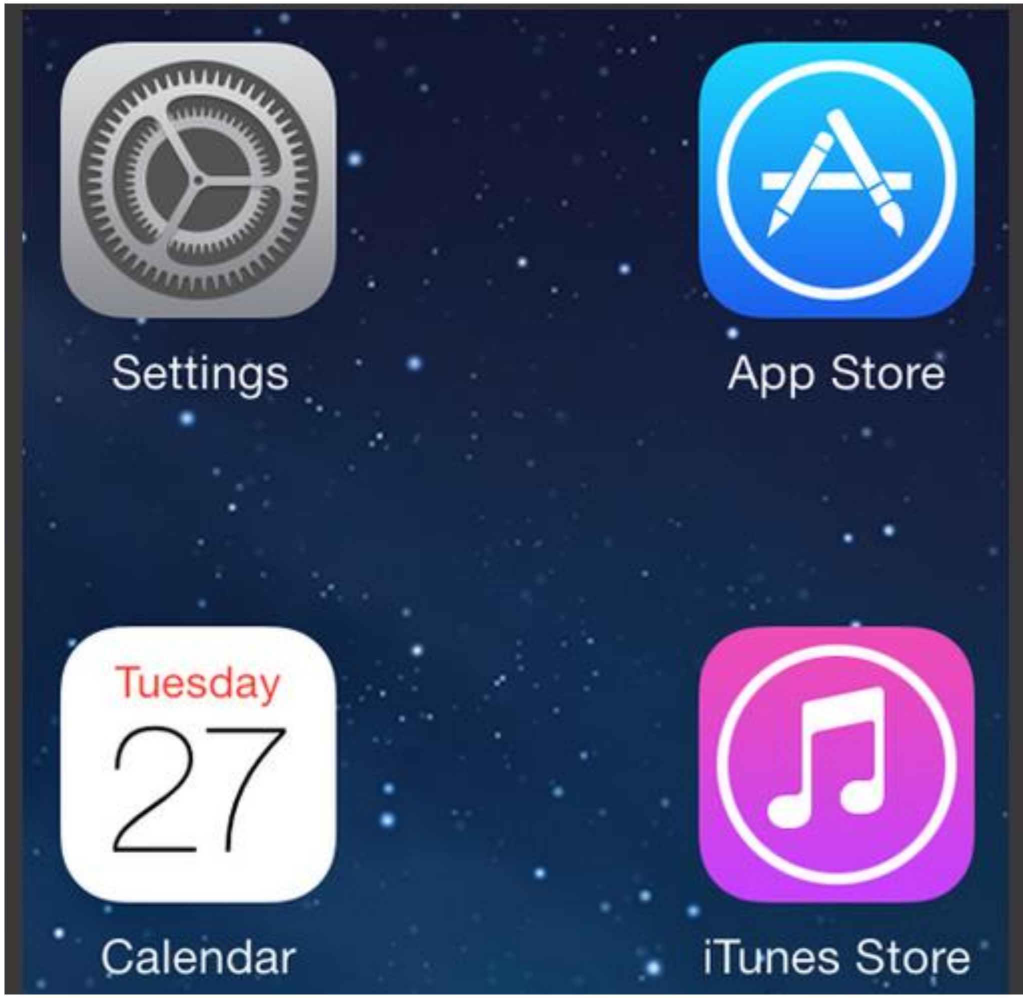




UI Design Changes

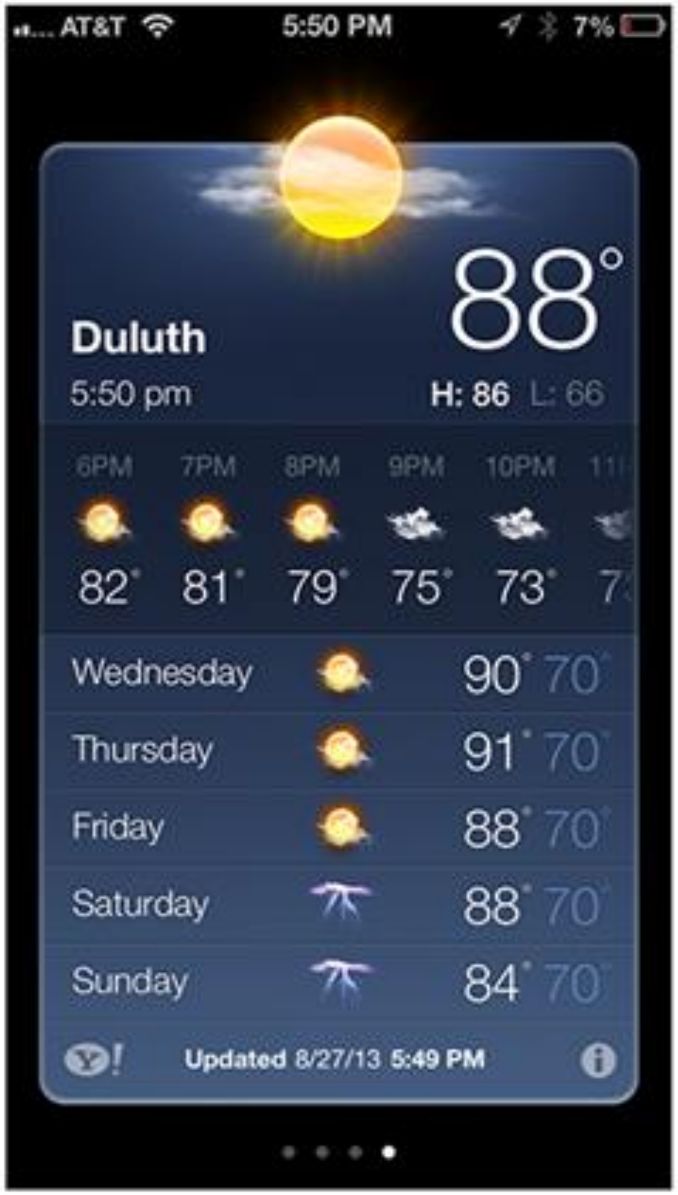
- The (New) Apple Effect



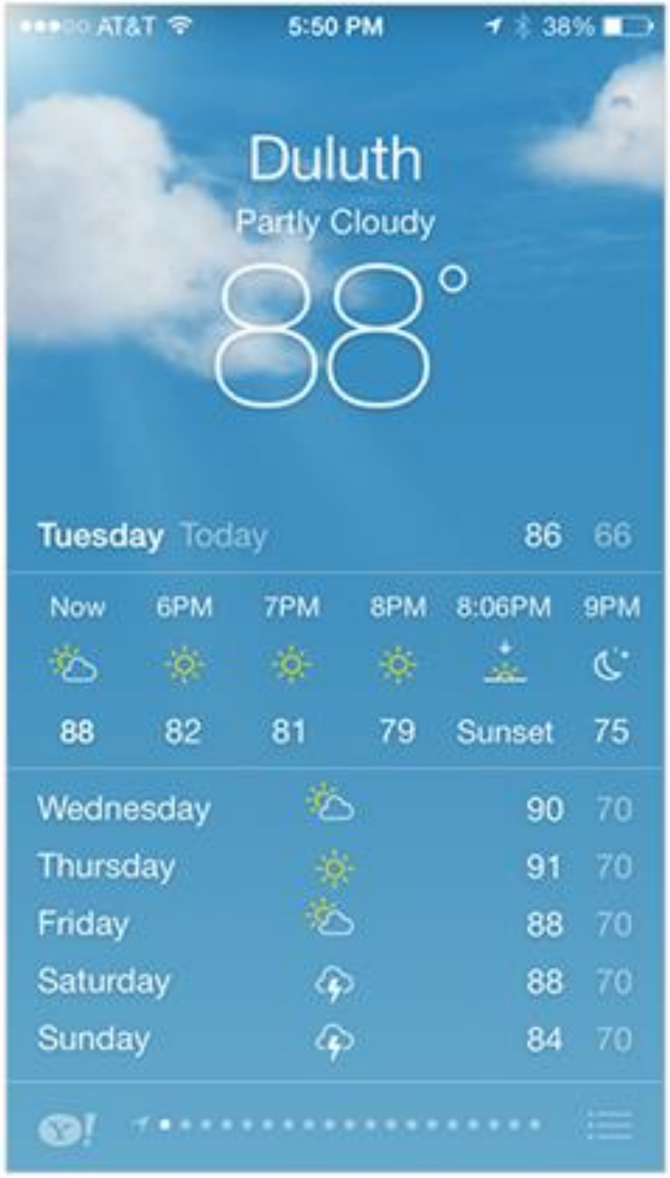


iOS 6 and iOS 7

iOS 6



iOS 7



Clicker

- Which do you like more:

A. The graphics with 3d like effects

B. The flat graphics

C. I don’t have a preference

Clicker

- How do you like your IDE’s set up?

A. Black text on white background

B. White text on black background

C. Something else

D. I don’t have a preference

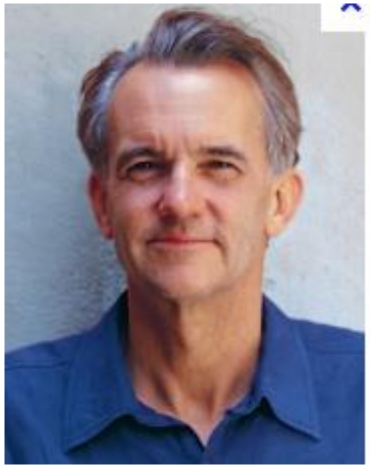
Android Theme

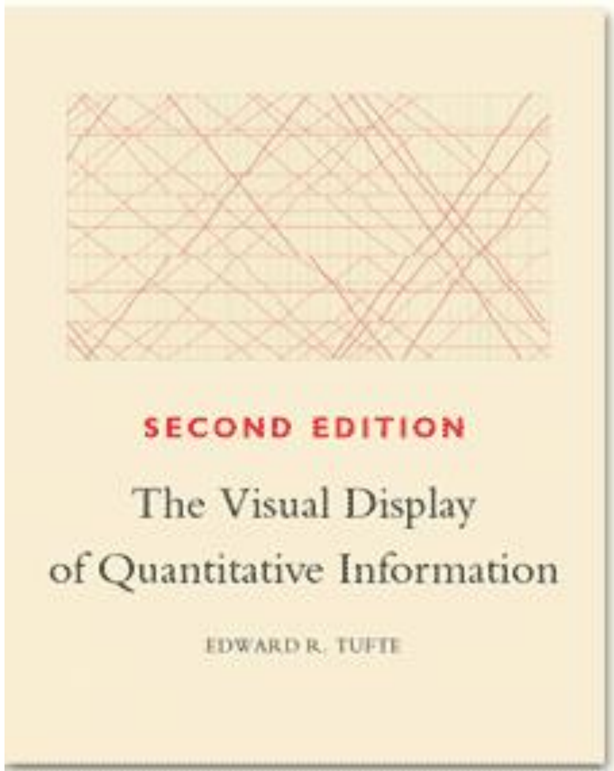
- Multiple Themes in Android's History
- Current Standard is the ***Material Theme***
- Set in Manifest
- Able to alter aspects of theme
- Creates a consistent look for app
- Emphasis on Feedback via animation

SIDETRACK ON THE VISUAL DISPLAY OF INFORMATION

Visual Display of Information

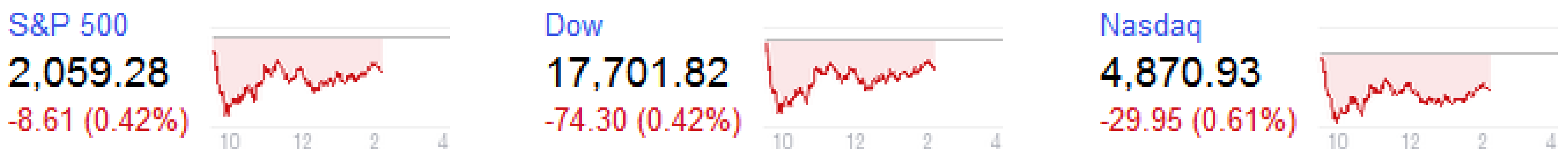
- Edward Tufte
  - Yale professor (poly sci, stats, and cs)
  - spark lines
  - small multiple
  - critical of PowerPoint



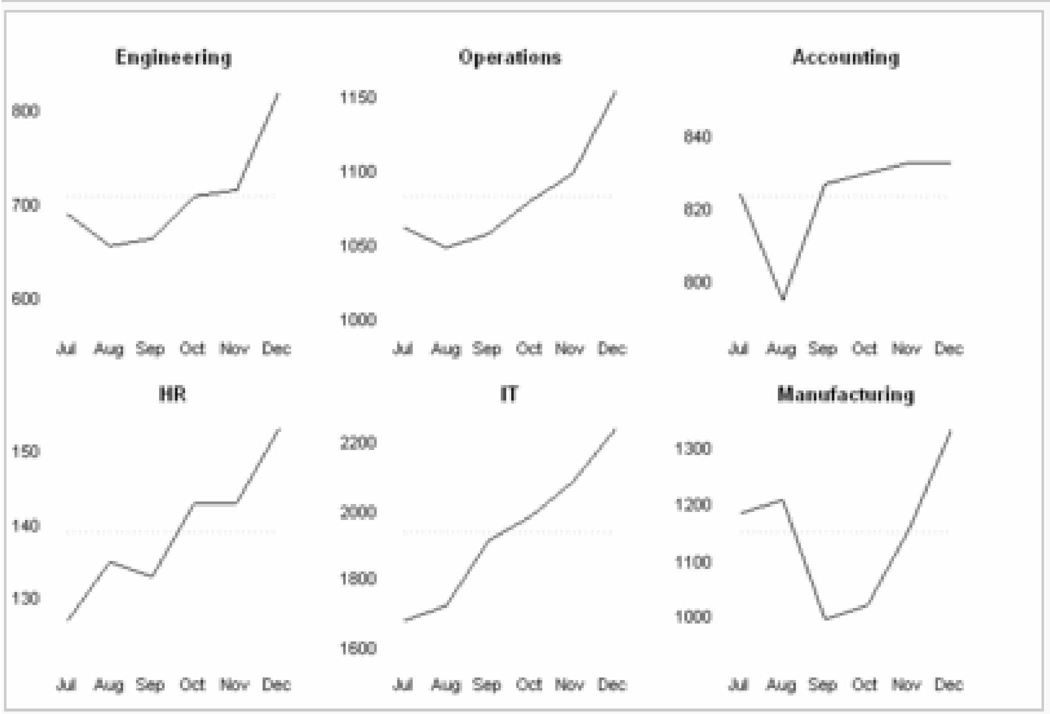


Visual Display

- Spark Lines



- Small Multiple

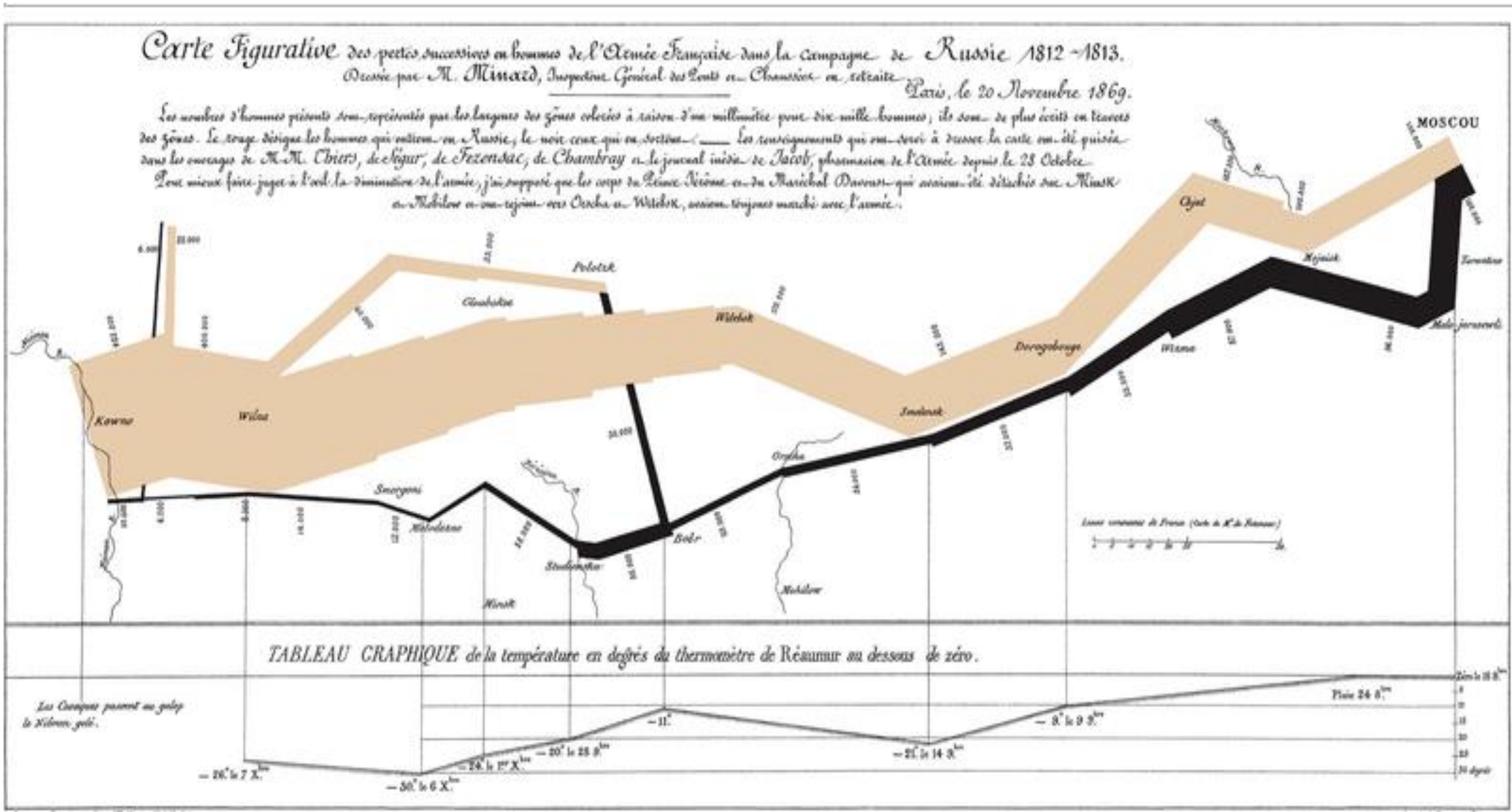


Departmental Salary Expenses

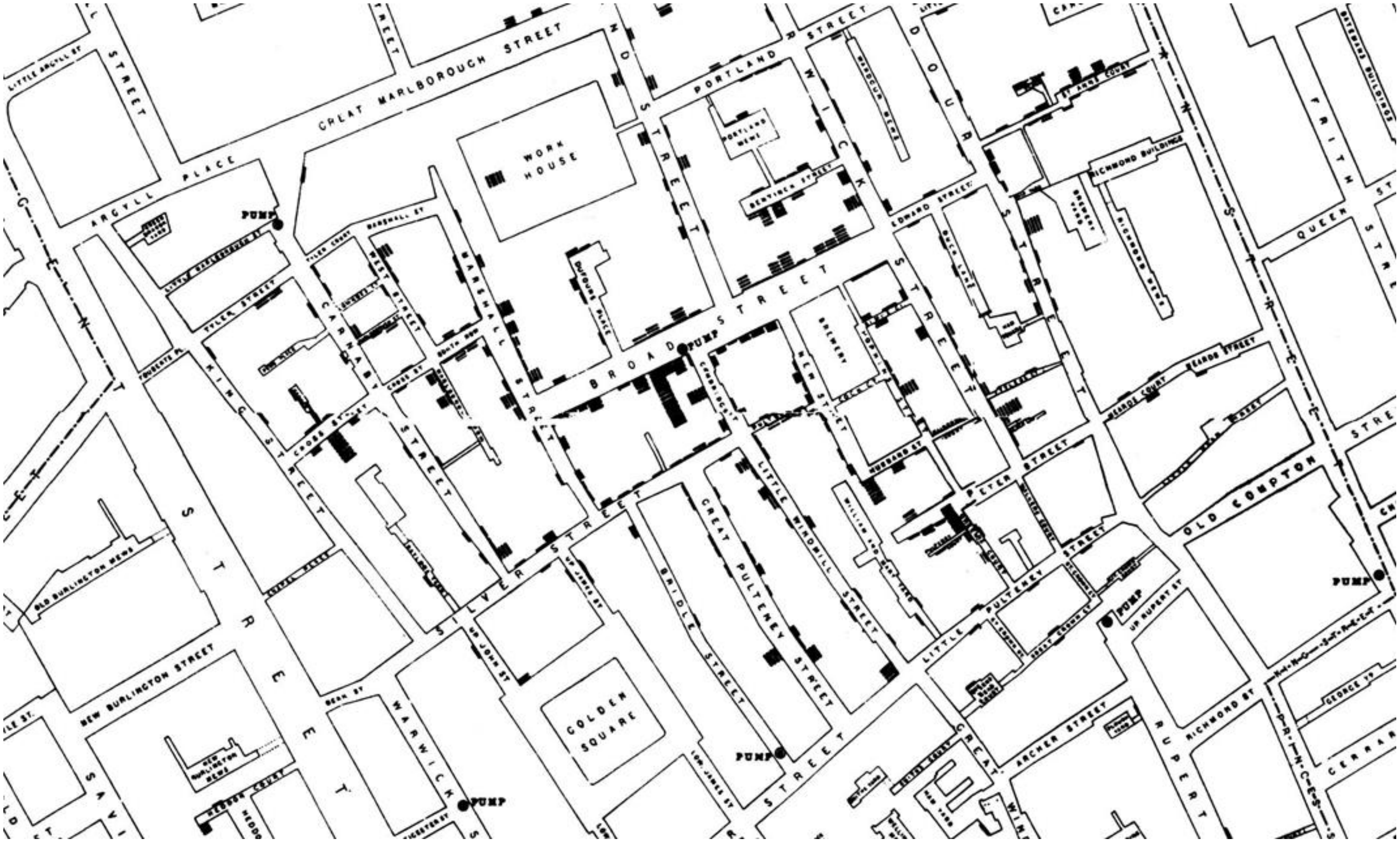
EXAMPLES OF THE VISUAL DISPLAY OF INFORMATION

Joseph Minard
Napoleon's Invasion of Russia

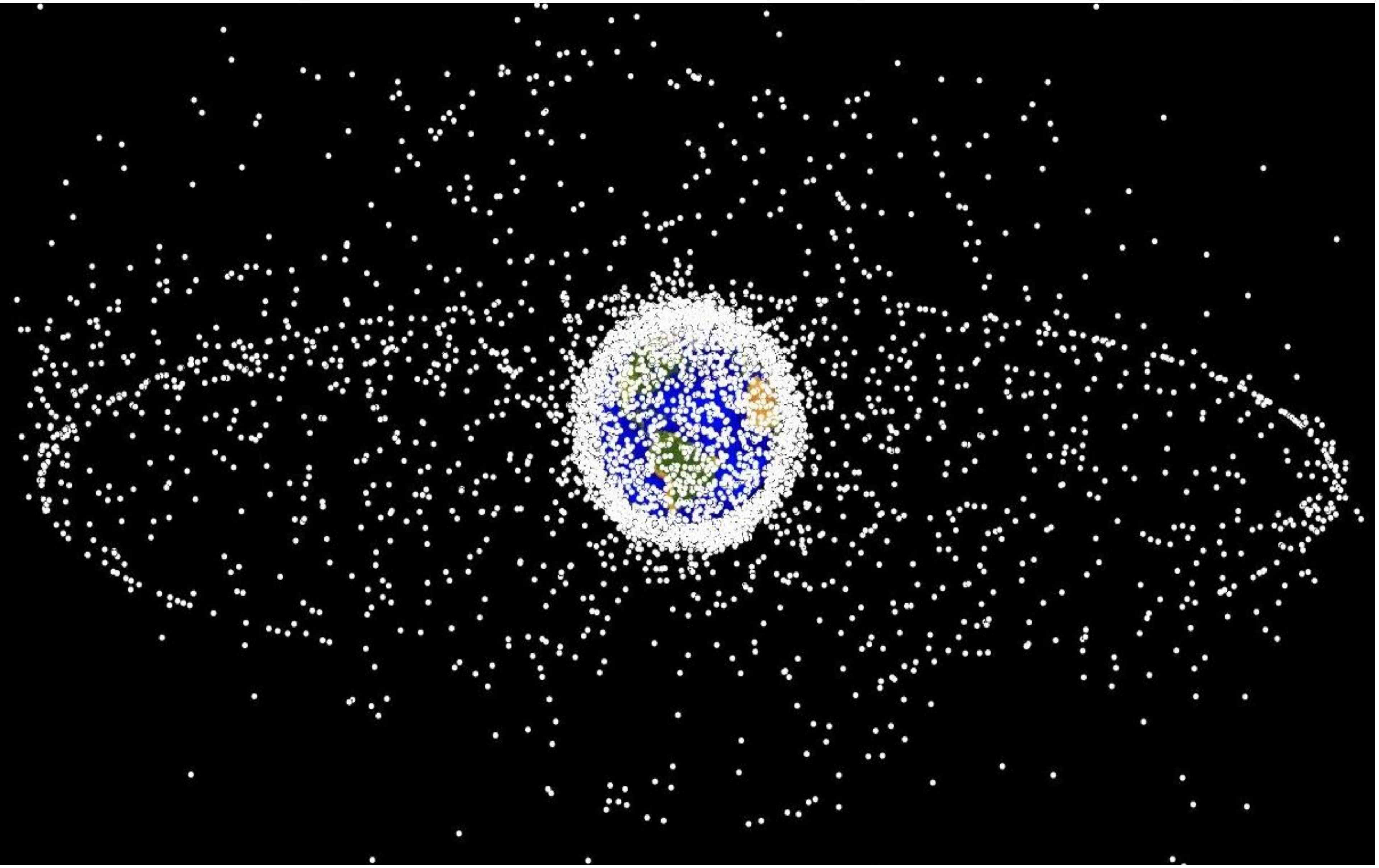
- Location, size of army, data, temperature



John Snow - Deaths from Cholera, 1854 Broad Street Outbreak (The Ghost Map)

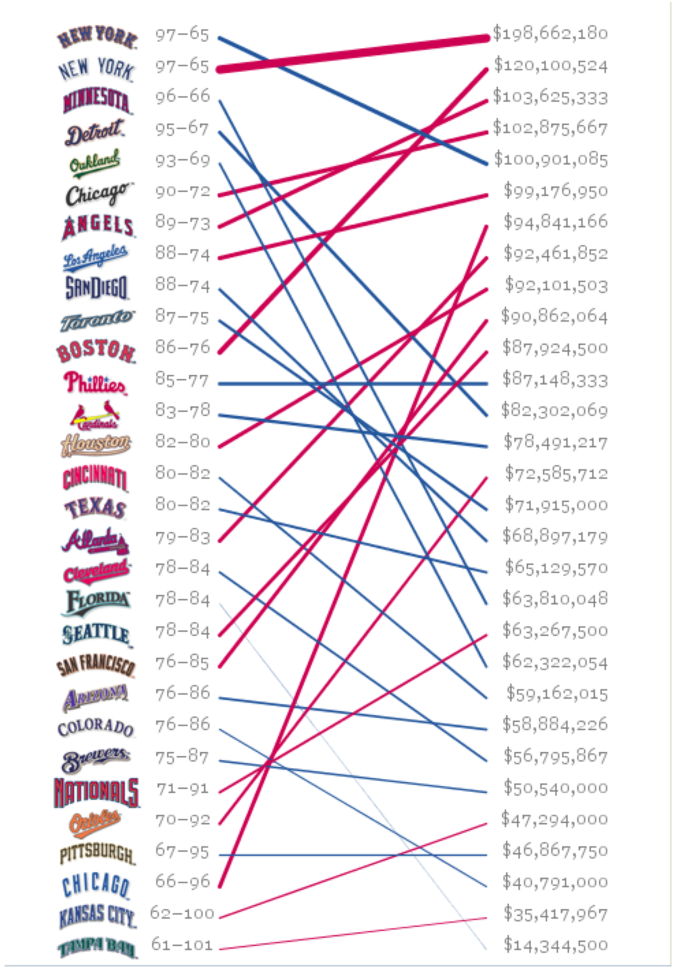


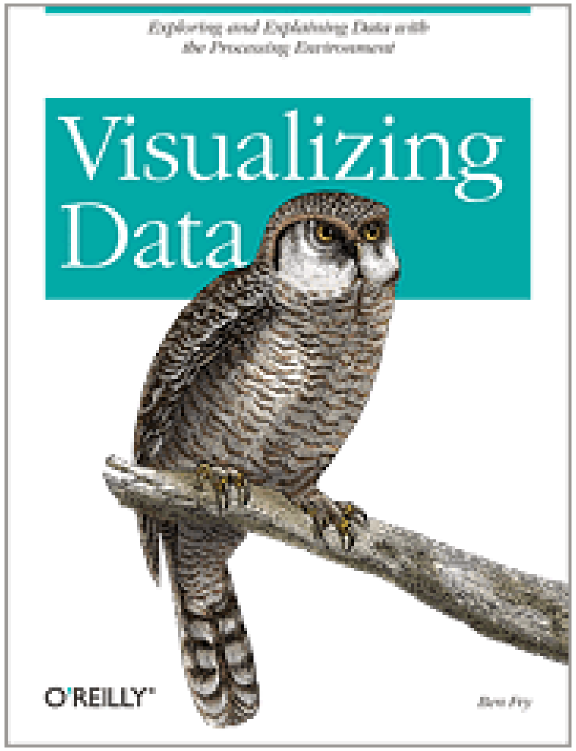
Space Debris Plot



Visualizing Data

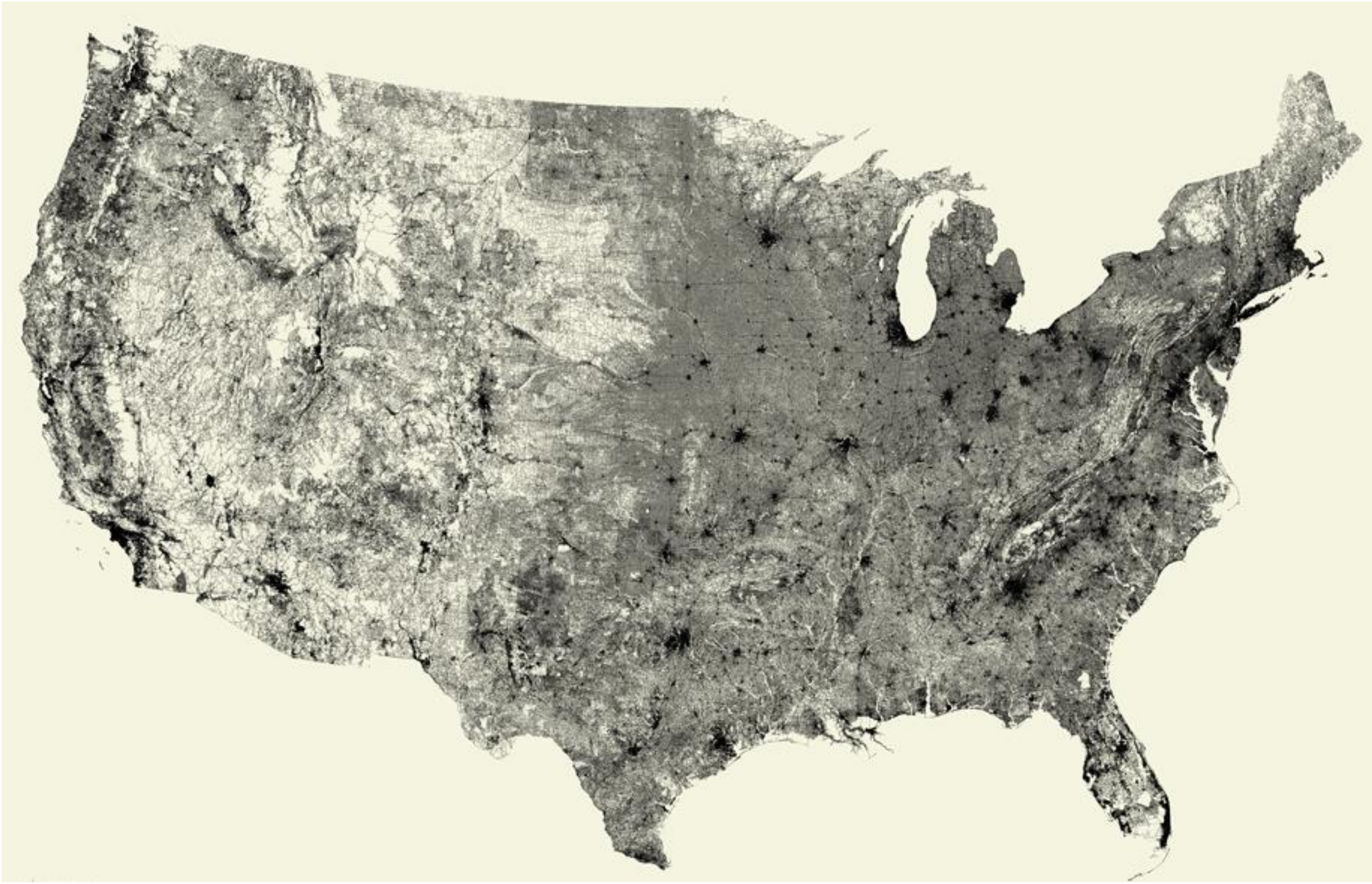
- Ben Fry, one of the creators of Processing



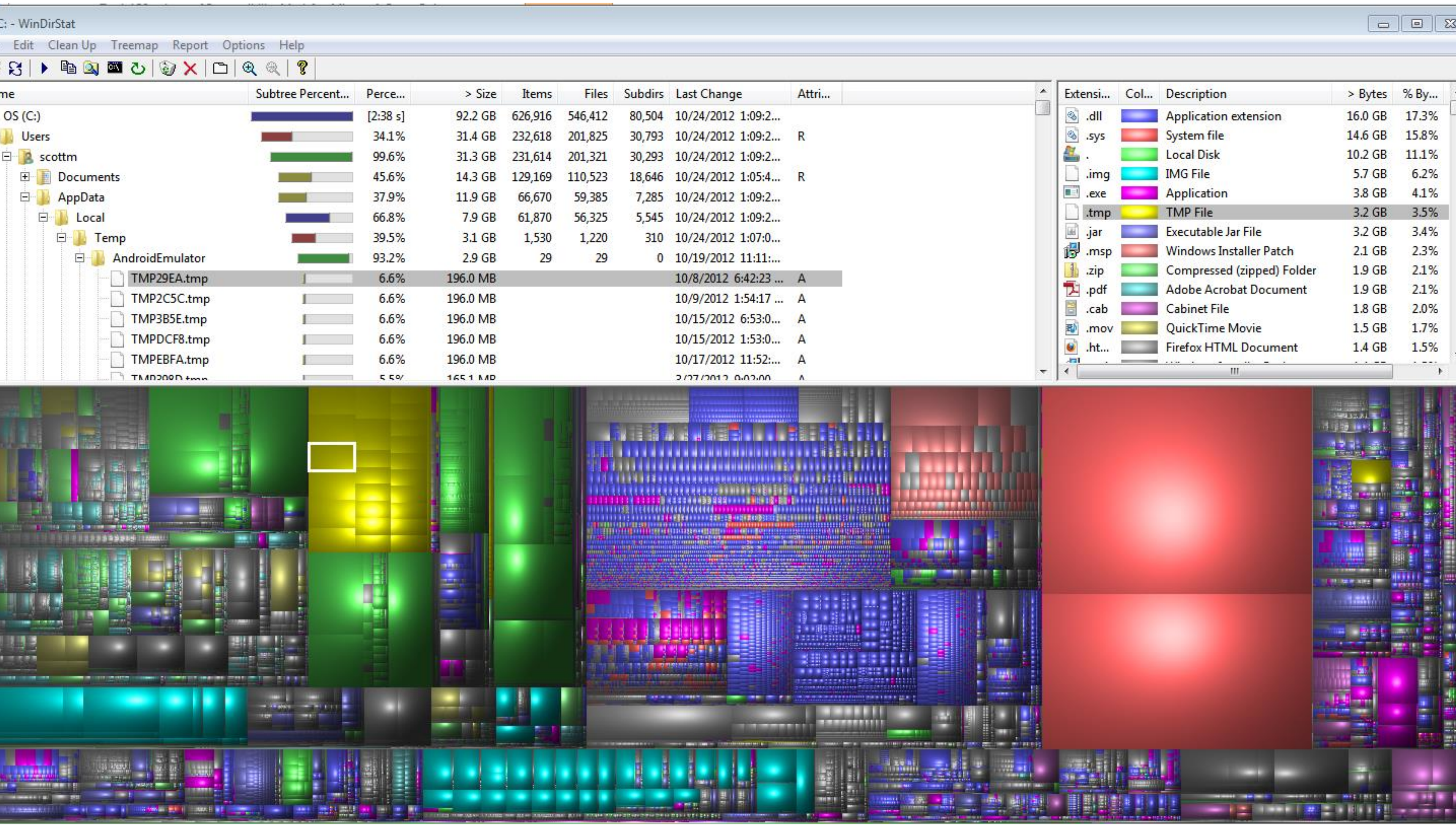


Exploring and Explaining Data with the Processing Environment
Visualizing Data
O'REILLY
Ben Fry

All Roads



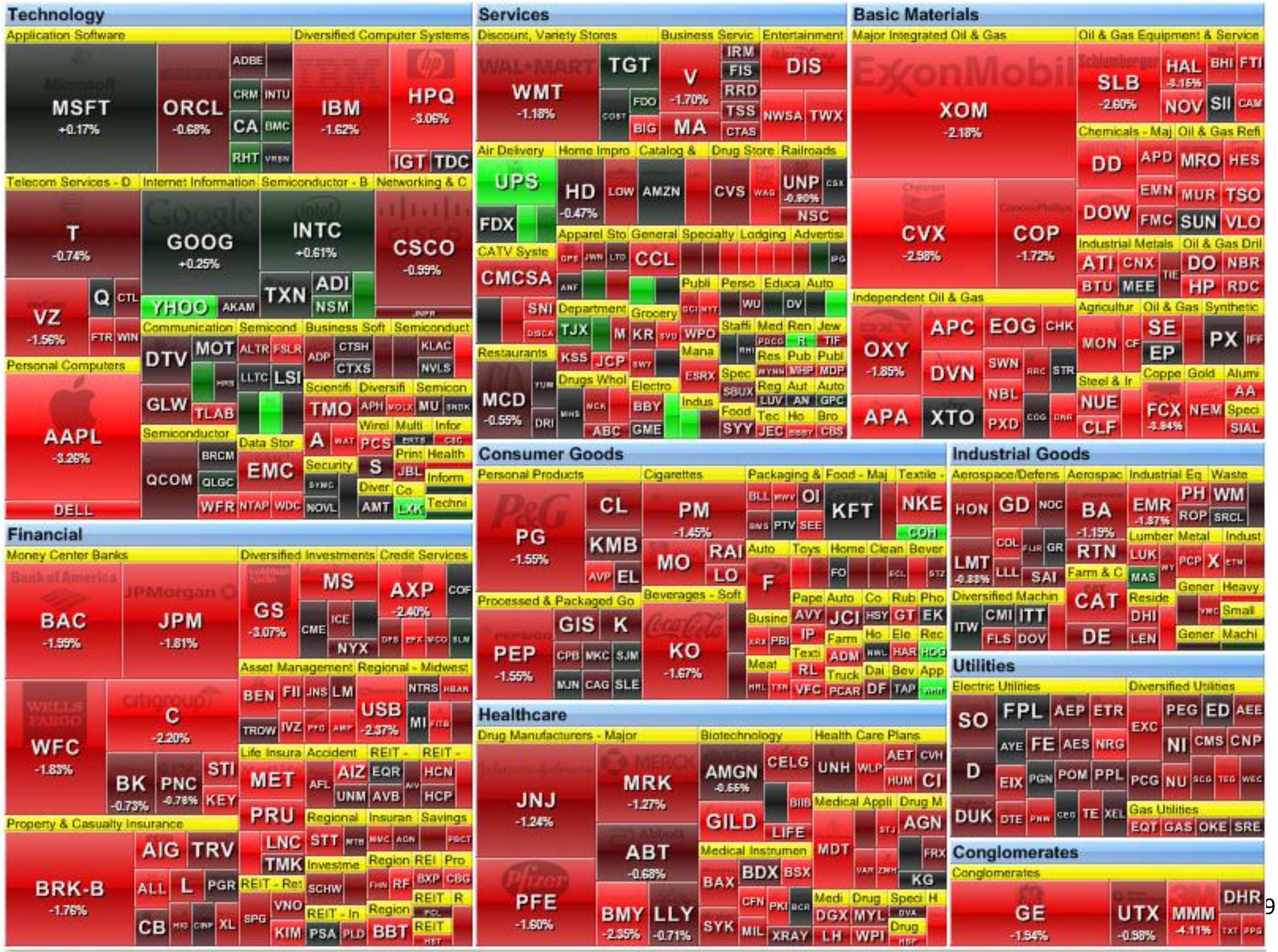
Files on Disk - WinDirStats



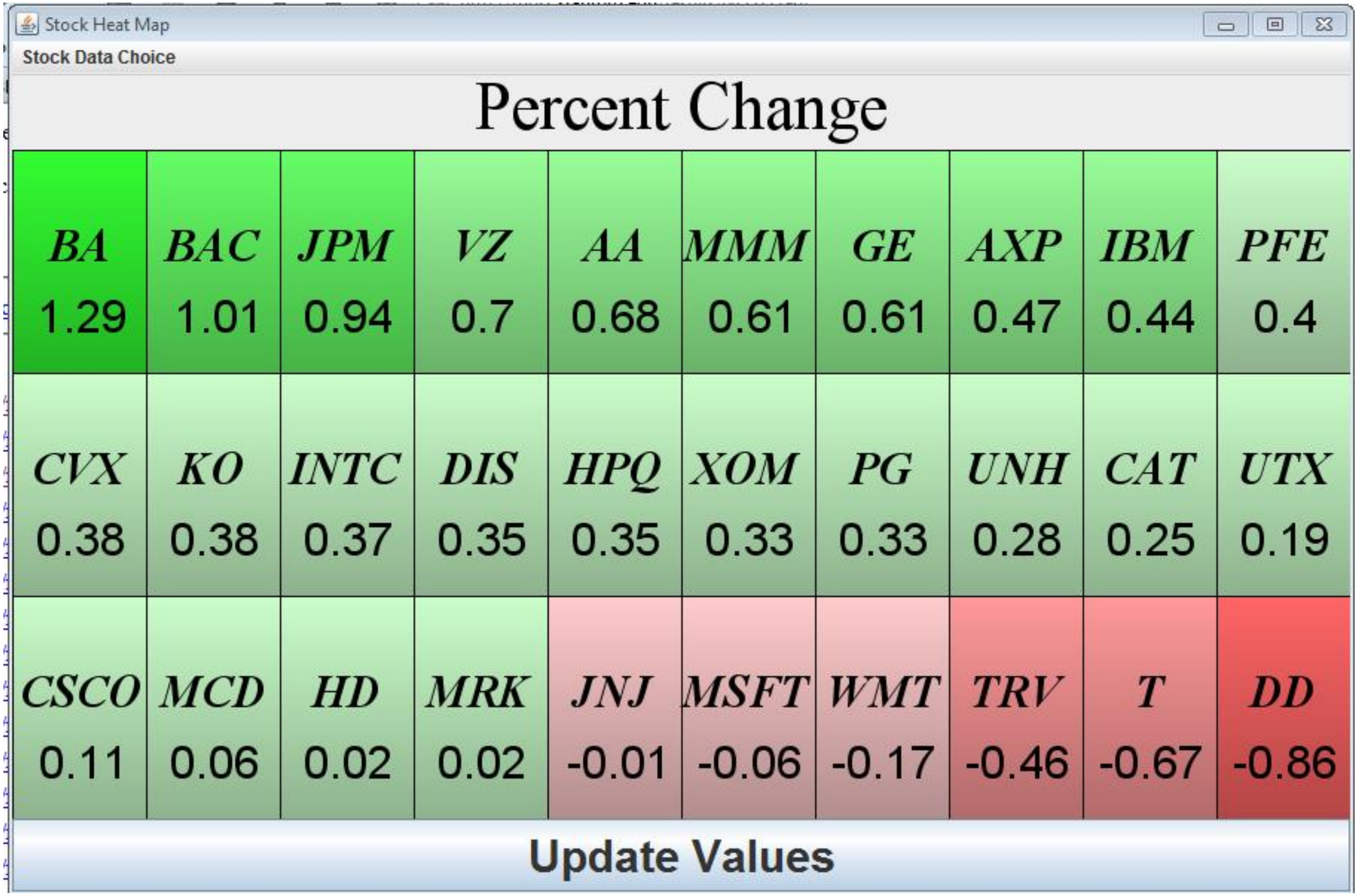
me	Subtree Percent...	Perce...	> Size	Items	Files	Subdirs	Last Change	Attri...
OS (C:)	▬	[2:38 s]	92.2 GB	626,916	546,412	80,504	10/24/2012 1:09:2...	
Users	▬	34.1%	31.4 GB	232,618	201,825	30,793	10/24/2012 1:09:2...	R
scottm	▬	99.6%	31.3 GB	231,614	201,321	30,293	10/24/2012 1:09:2...	
Documents	▬	45.6%	14.3 GB	129,169	110,523	18,646	10/24/2012 1:05:4...	R
AppData	▬	37.9%	11.9 GB	66,670	59,385	7,285	10/24/2012 1:09:2...	
Local	▬	66.8%	7.9 GB	61,870	56,325	5,545	10/24/2012 1:09:2...	
Temp	▬	39.5%	3.1 GB	1,530	1,220	310	10/24/2012 1:07:0...	
AndroidEmulator	▬	93.2%	2.9 GB	29	29	0	10/19/2012 11:11:...	
TMP29EA.tmp	▬	6.6%	196.0 MB				10/8/2012 6:42:23 ...	A
TMP2C5C.tmp	▬	6.6%	196.0 MB				10/9/2012 1:54:17 ...	A
TMP3B5E.tmp	▬	6.6%	196.0 MB				10/15/2012 6:53:0...	A
TMPDCF8.tmp	▬	6.6%	196.0 MB				10/15/2012 1:53:0...	A
TMPEBFA.tmp	▬	6.6%	196.0 MB				10/17/2012 11:52:...	A
TMP308D.tmp	▬	5.5%	165.1 MB				3/27/2012 9:02:00	A

Extensi...	Col...	Description	> Bytes	% By...
.dll	🟦	Application extension	16.0 GB	17.3%
.sys	🟥	System file	14.6 GB	15.8%
.	🟩	Local Disk	10.2 GB	11.1%
.img	🟦	IMG File	5.7 GB	6.2%
.exe	🟪	Application	3.8 GB	4.1%
.tmp	🟨	TMP File	3.2 GB	3.5%
.jar	🟦	Executable Jar File	3.2 GB	3.4%
.msp	🟥	Windows Installer Patch	2.1 GB	2.3%
.zip	🟩	Compressed (zipped) Folder	1.9 GB	2.1%
.pdf	🟦	Adobe Acrobat Document	1.9 GB	2.1%
.cab	🟪	Cabinet File	1.8 GB	2.0%
.mov	🟨	QuickTime Movie	1.5 GB	1.7%
.ht...	⬜	Firefox HTML Document	1.4 GB	1.5%

S&P 500 Heat Map



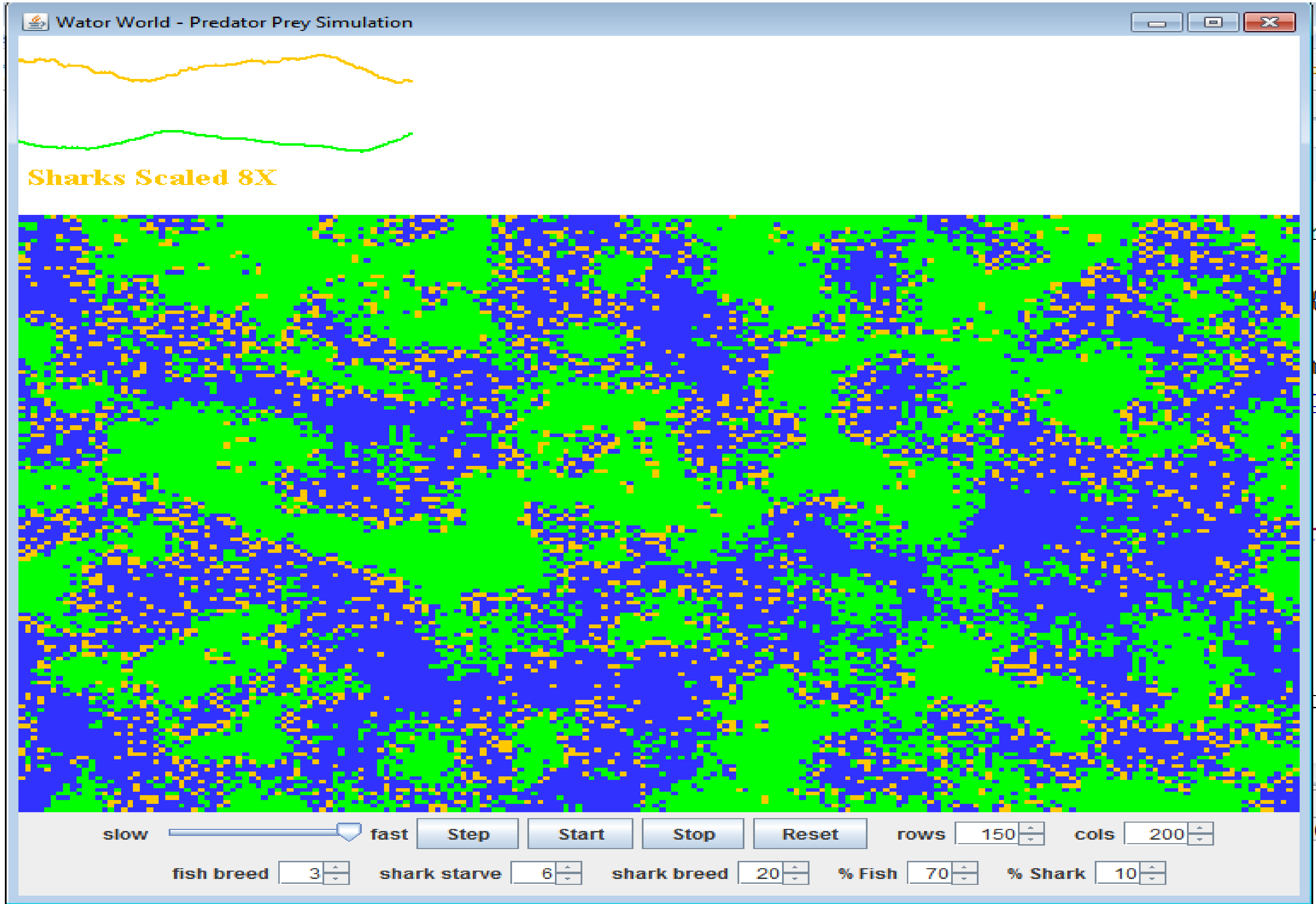
CS324E Heat Map



Percent Change

BA 1.29	BAC 1.01	JPM 0.94	VZ 0.7	AA 0.68	MMM 0.61	GE 0.61	AXP 0.47	IBM 0.44	PFE 0.4
CVX 0.38	KO 0.38	INTC 0.37	DIS 0.35	HPQ 0.35	XOM 0.33	PG 0.33	UNH 0.28	CAT 0.25	UTX 0.19
CSCO 0.11	MCD 0.06	HD 0.02	MRK 0.02	JNJ -0.01	MSFT -0.06	WMT -0.17	TRV -0.46	T -0.67	DD -0.86

CS324E Wator World



Wator World - Predator Prey Simulation

rows 150

cols 200

fish breed 3

shark starve 6

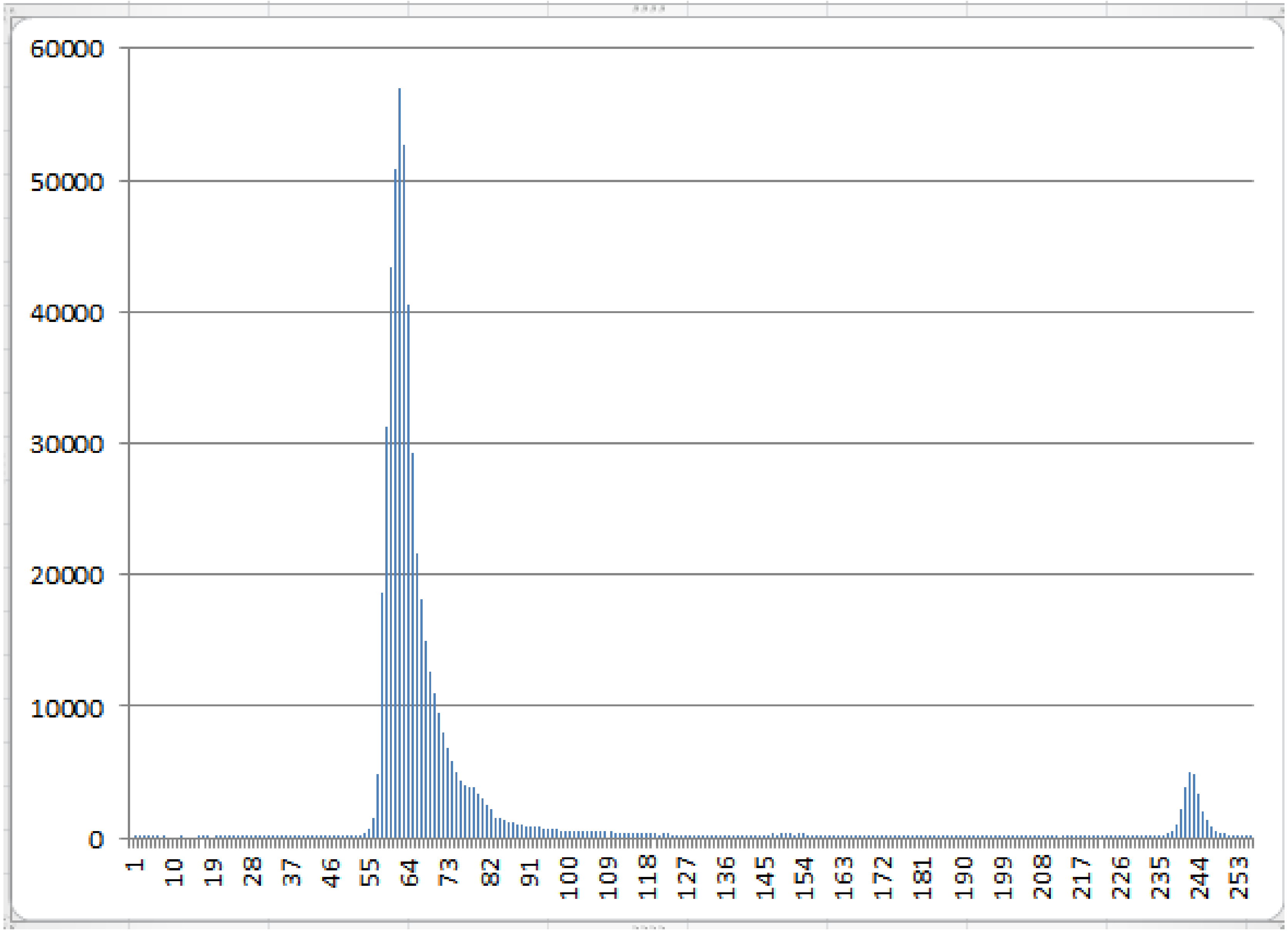
shark breed 20

% Fish 70

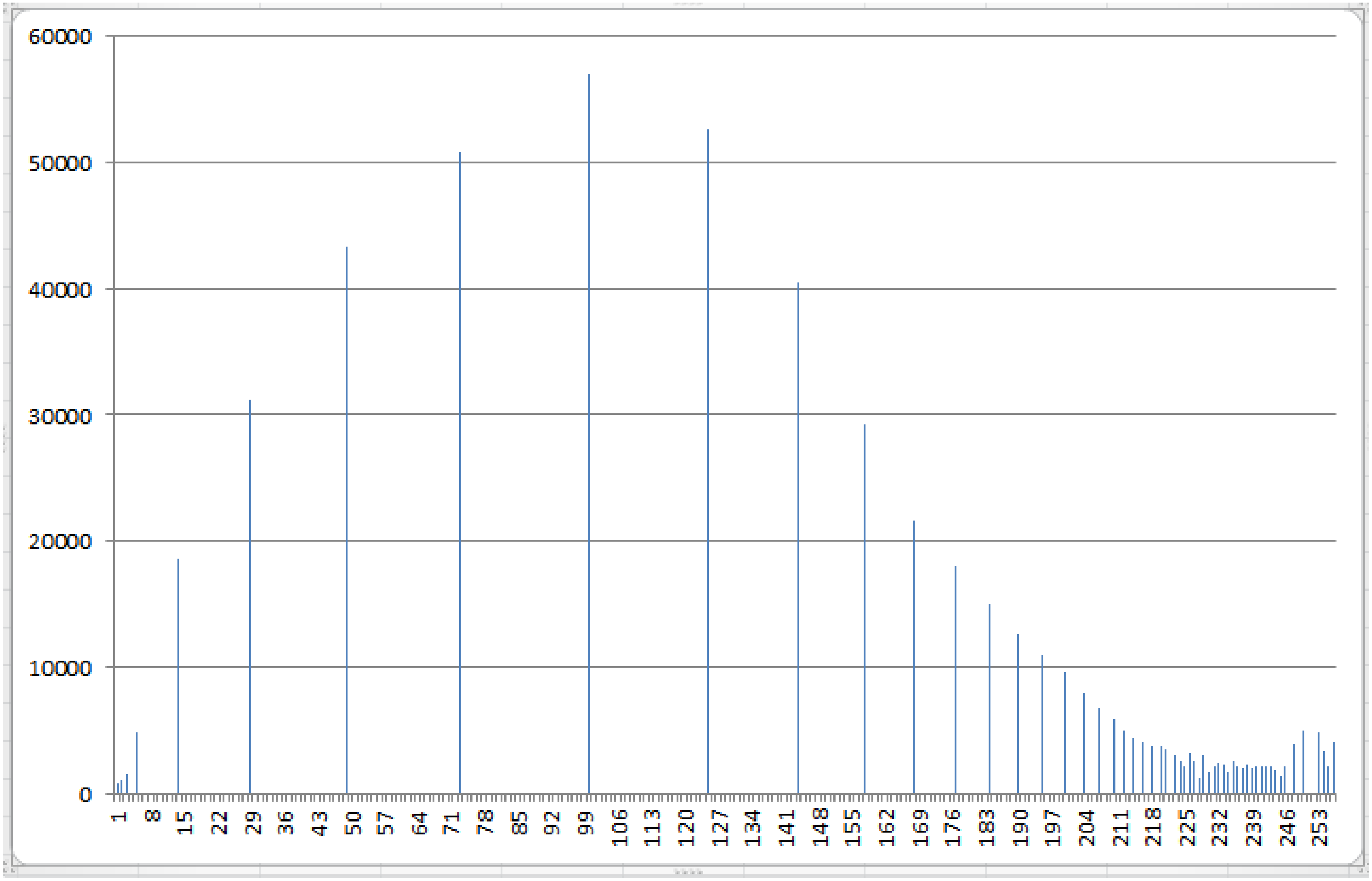
% Shark 10

Image Processing

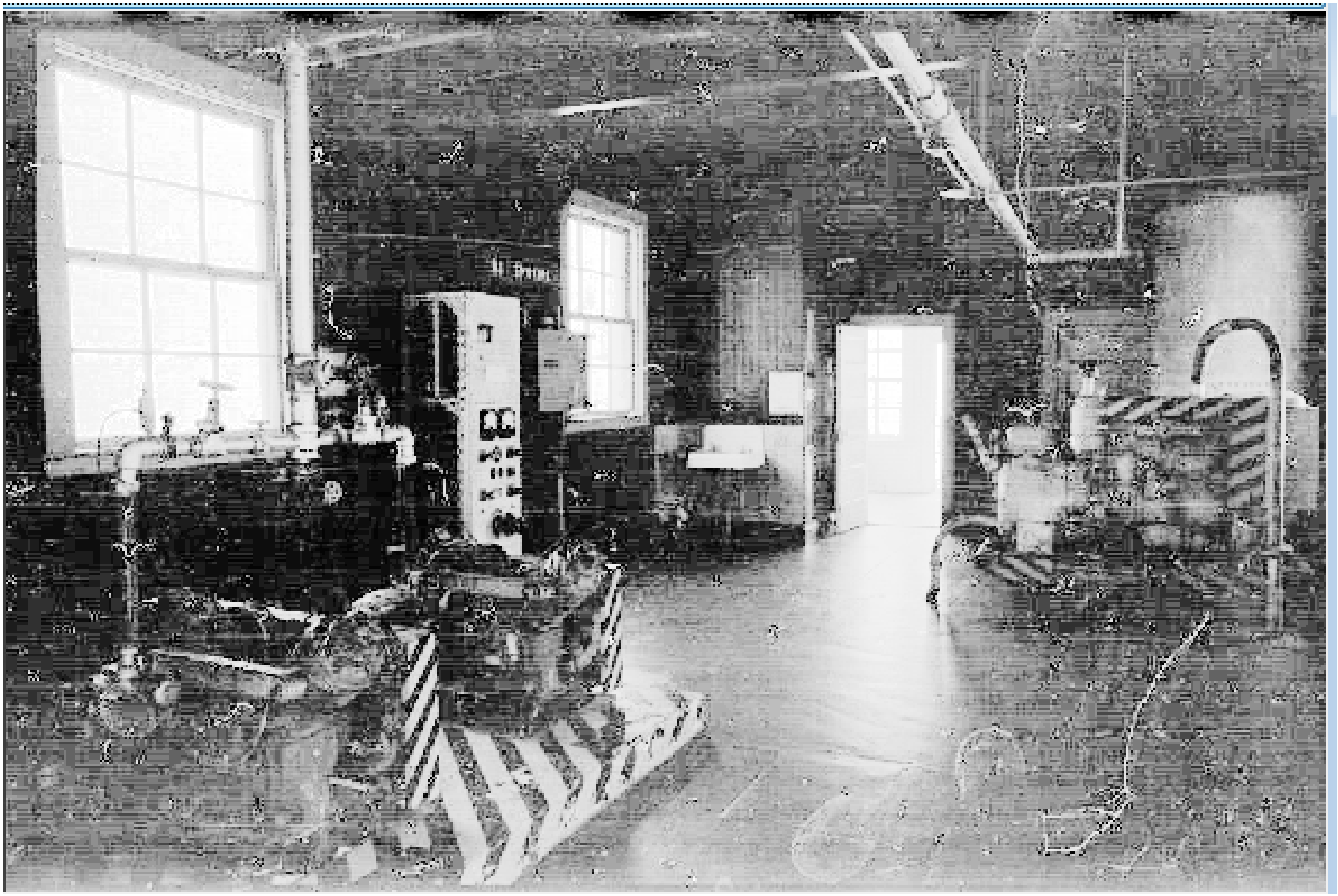
Color Histogram

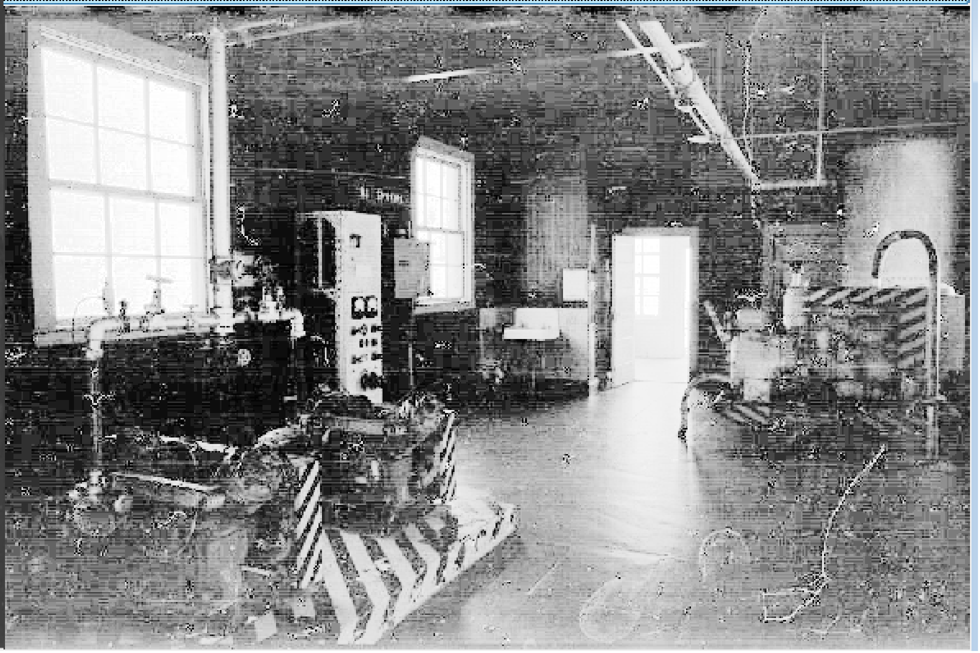


Histogram Equalization



Revised Image





USING ANDROID NATIVE 2D GRAPHICS

Android Graphics

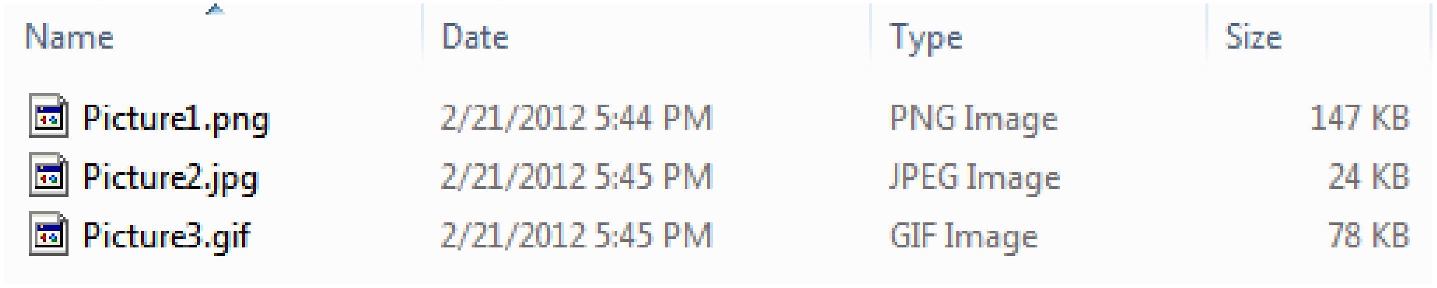
- NOT the Java awt or swing packages
- custom set of classes
- Canvas: class that holds code for various "draw" methods
  - Similar to Java Graphics, Graphics2D objects
- Paint: Controls the drawing. A whole host of properties.
- Bitmap: the things drawn on
- Drawable: the thing to draw. (rectangles, images, lines, etc.)
- Typeface: for fonts

Using a Canvas

- Simple way -> Create a custom View and override the onDraw method
- The Canvas is a parameter to onDraw
- Create a class that extends View
  - **<u>override the 2 parameter constructor</u>**
  - override the onDraw method
  - perform custom drawing in the onDraw method
  - add the View to the proper layout

Graphics Resources

- Use Drawables in Views
- Create a folder res/drawable
- Add images
  - png (preferred)
  - jpg (acceptable)
  - gif (discouraged)



Name	Date	Type	Size
Picture1.png	2/21/2012 5:44 PM	PNG Image	147 KB
Picture2.jpg	2/21/2012 5:45 PM	JPEG Image	24 KB
Picture3.gif	2/21/2012 5:45 PM	GIF Image	78 KB

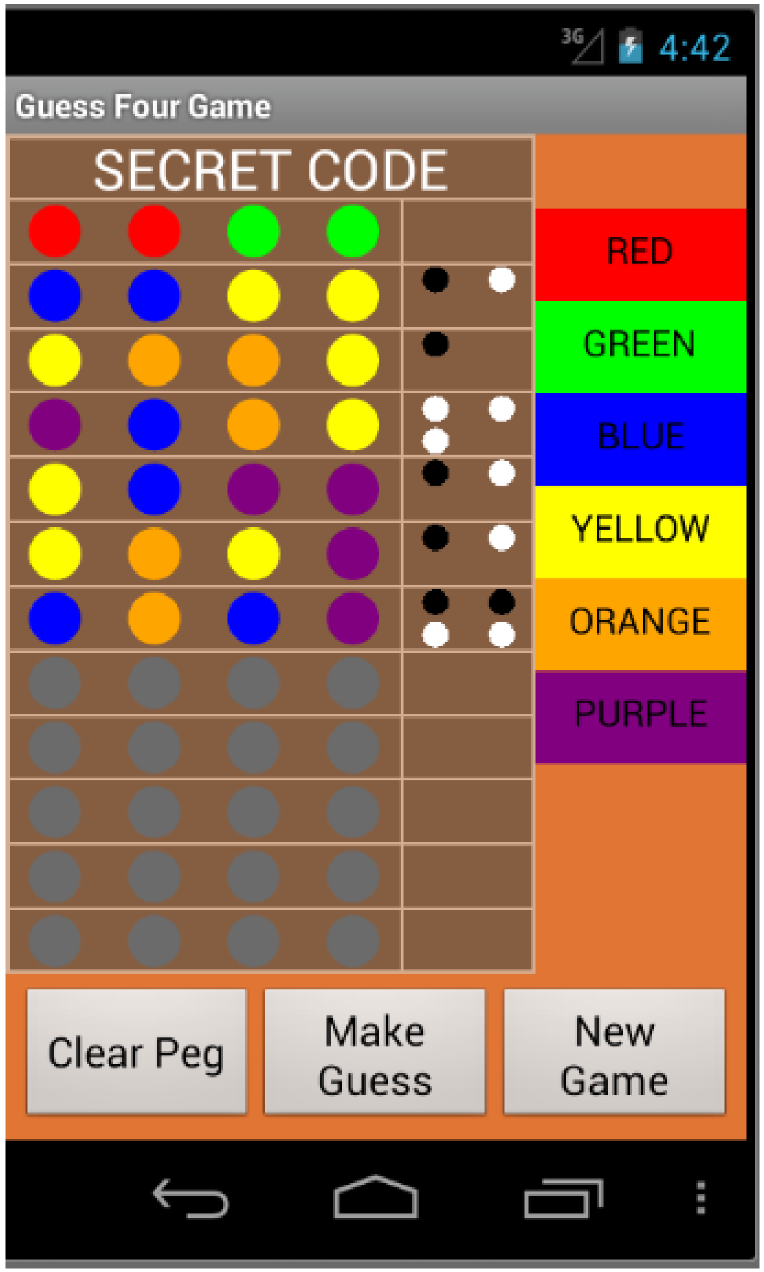
- Images can be added as background for Views

Requesting Redraws of a View

- call invalidate() on a View to redraw it
  - invalidate redraws the whole View
  - possible to redraw only a portion of the View, the part that changed
  - several overloaded versions of invalidate that accept a region of the View
  - only that portion redrawn
- Override the onDraw method for the View to redraw
  - **key we override onDraw but don't call it! call invalidate!**
- for really complex drawing and graphics move drawing off of the UI thread to a SurfaceView (more complex)

GuessFour

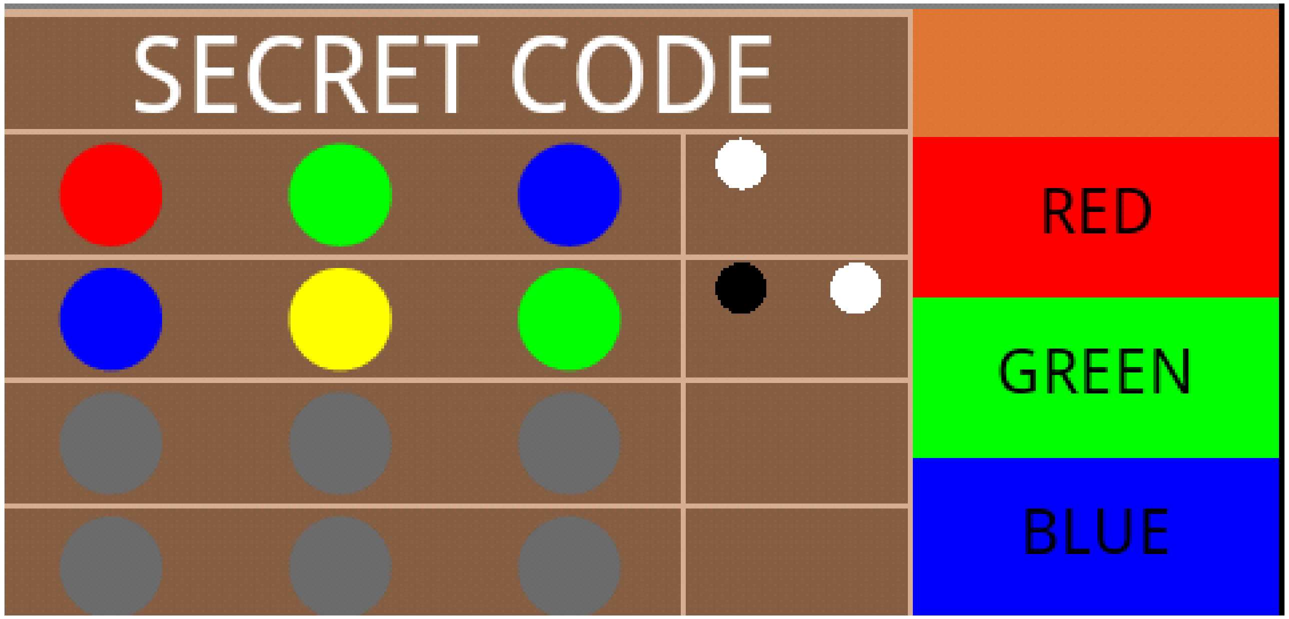
- Board drawn in onDraw method of a View
- Board will resize for different devices
- lines, ovals, rectangles, and texts



Paint

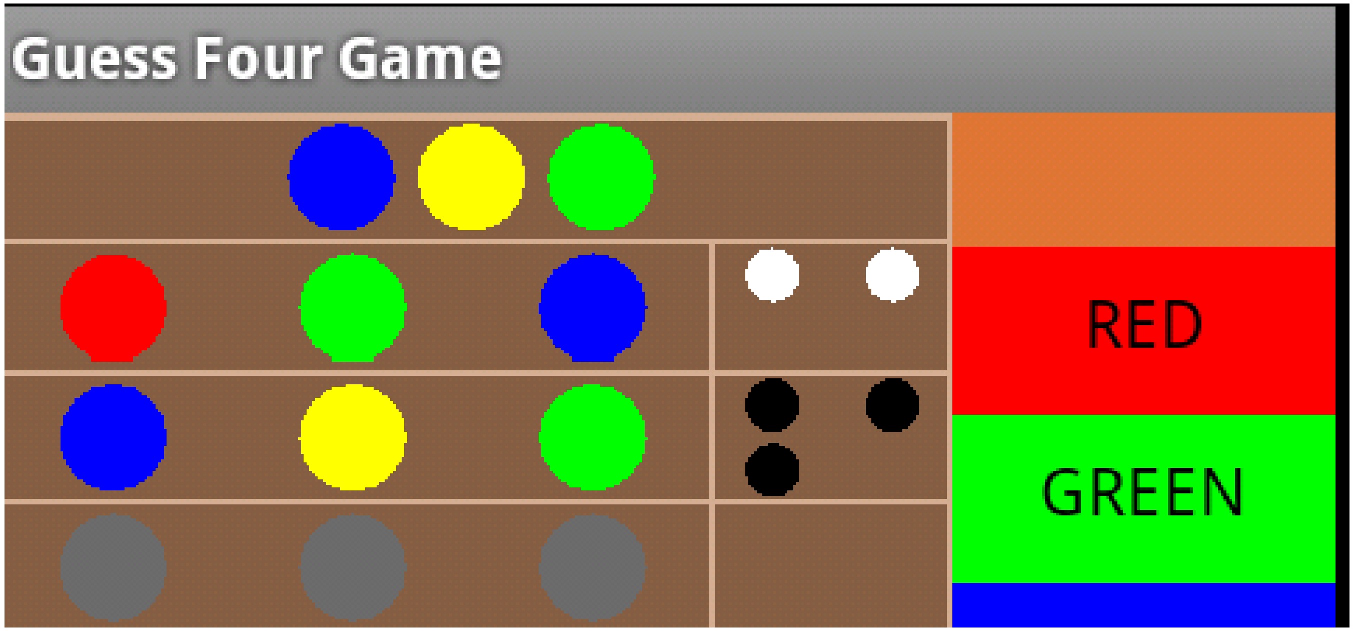
- typically create Paint with anti aliasing enable
- `Paint p =`
  ```
      new Paint(Paint.ANTI_ALIAS_FLAG);
  ```

Anti Aliasing on

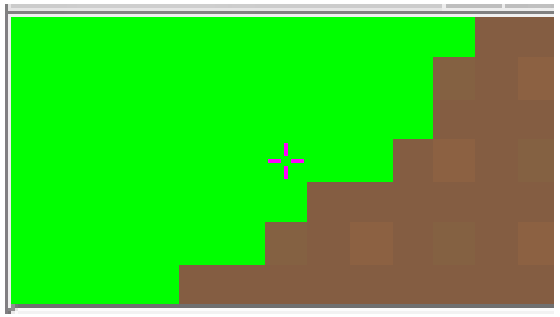


Anti Aliasing - The Jaggies

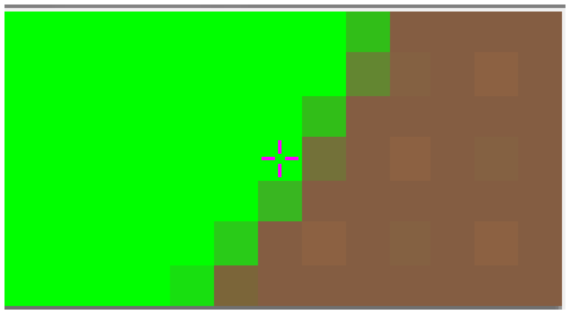
Anti Aliasing off



Anti Aliasing off



Anti Aliasing on



Using the Canvas Class

- methods to draw:
- rectangles
- lines
- arcs
- paths
- images
- circles
- ovals
- points
- text
- and many more

- Ability to set the "clip", portion of canvas where drawing takes place
  – commands to draw something that is outside clip are ignored
- Ability to translate, rotate, and scale the canvas

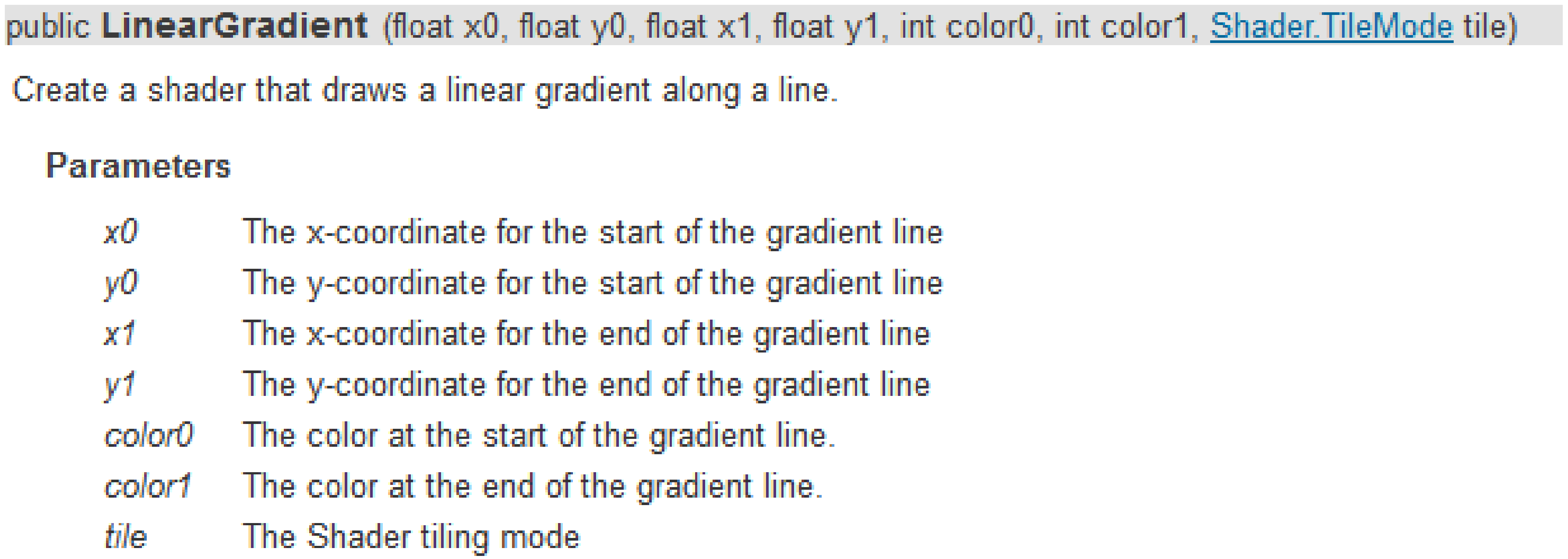
Paint Object

- many, many attributes and properties including:
  - current color to draw with
    - Color specified as aRGB
  - whether to fill or outline shapes
  - size of stroke when drawing
  - text attributes including size, style (e.g. underline, bold), alignment,
  - gradients

Gradients

- 3 kinds of gradients
- LinearGradeint
- RadialGradeint
- SweepGradient
- at least 2 color, but possibly more
- flows from one color to another

Linear Gradient

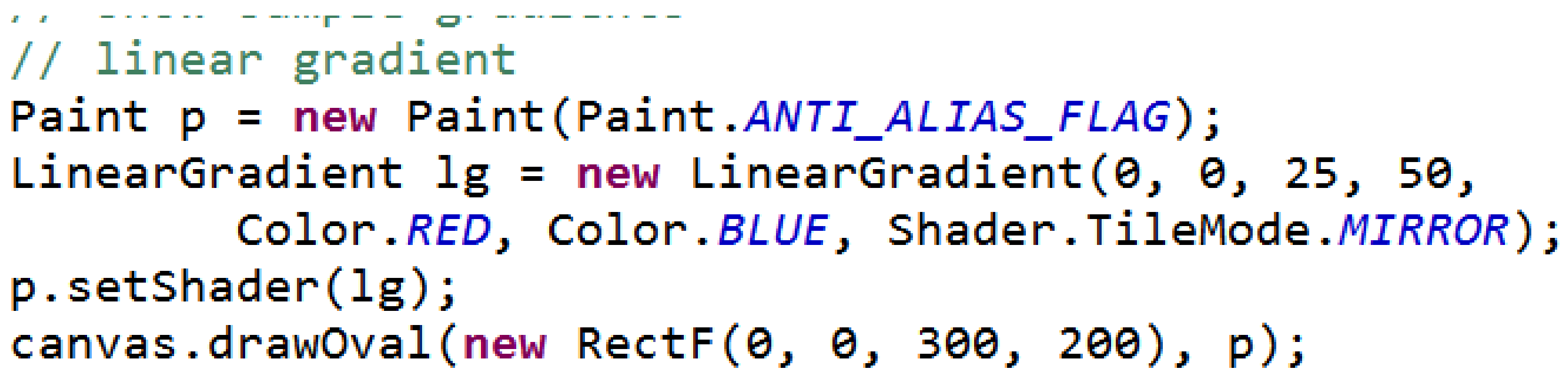


public **LinearGradient** (float x0, float y0, float x1, float y1, int color0, int color1, Shader.TileMode tile)

Create a shader that draws a linear gradient along a line.

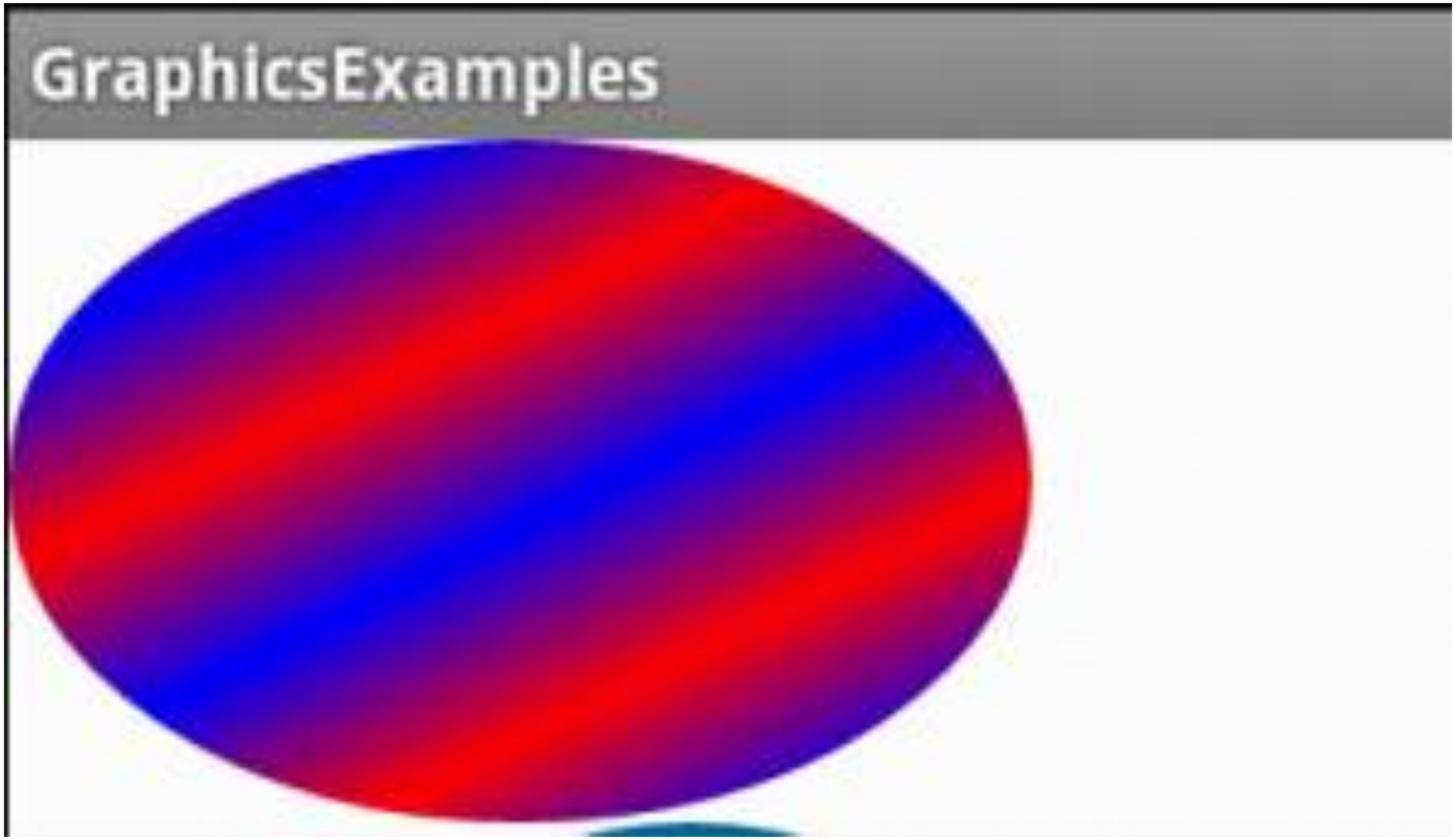
Parameters

x0	The x-coordinate for the start of the gradient line
y0	The y-coordinate for the start of the gradient line
x1	The x-coordinate for the end of the gradient line
y1	The y-coordinate for the end of the gradient line
color0	The color at the start of the gradient line.
color1	The color at the end of the gradient line.
tile	The Shader tiling mode

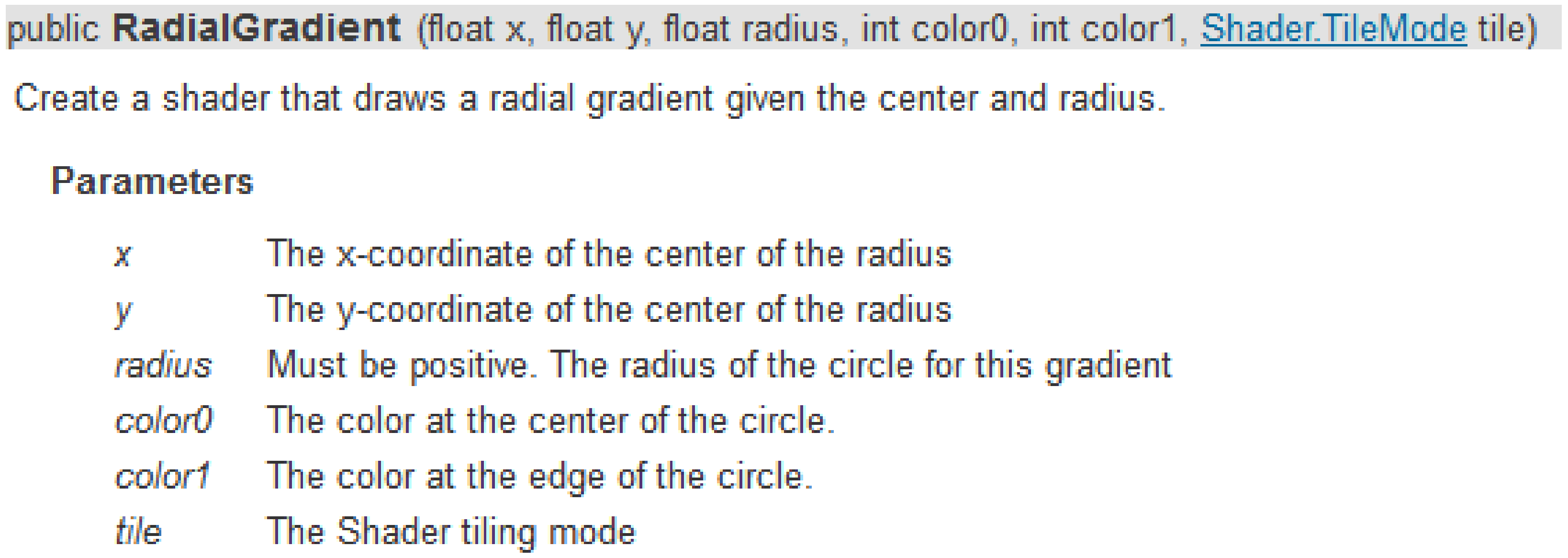


```java
// linear gradient
Paint p = new Paint(Paint.ANTI_ALIAS_FLAG);
LinearGradient lg = new LinearGradient(0, 0, 25, 50,
        Color.RED, Color.BLUE, Shader.TileMode.MIRROR);
p.setShader(lg);
canvas.drawOval(new RectF(0, 0, 300, 200), p);
```

LinearGradient



RadialGradient



public **RadialGradient** (float x, float y, float radius, int color0, int color1, Shader.TileMode tile)

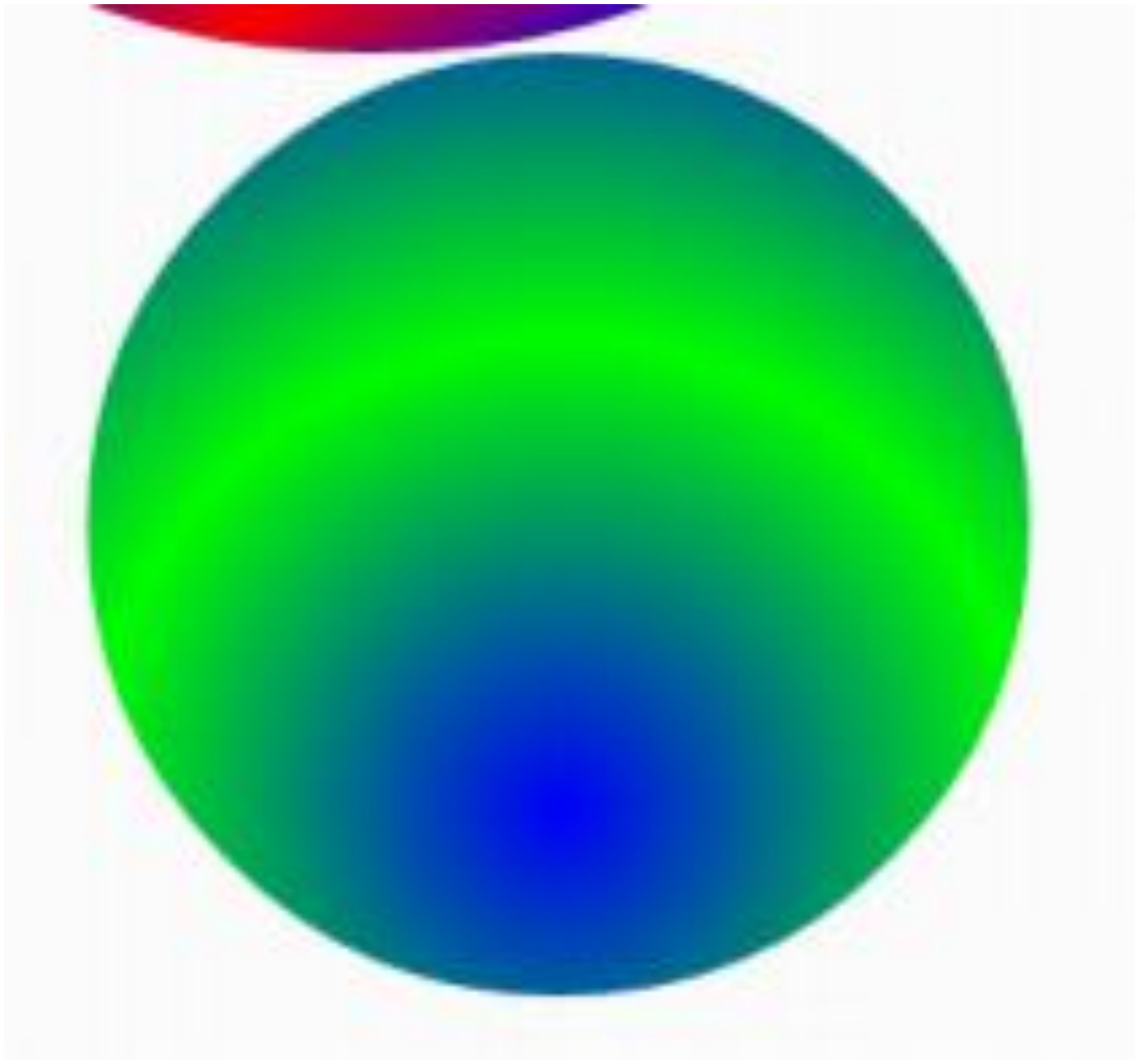
Create a shader that draws a radial gradient given the center and radius.

Parameters

x	The x-coordinate of the center of the radius
y	The y-coordinate of the center of the radius
radius	Must be positive. The radius of the circle for this gradient
color0	The color at the center of the circle.
color1	The color at the edge of the circle.
tile	The Shader tiling mode

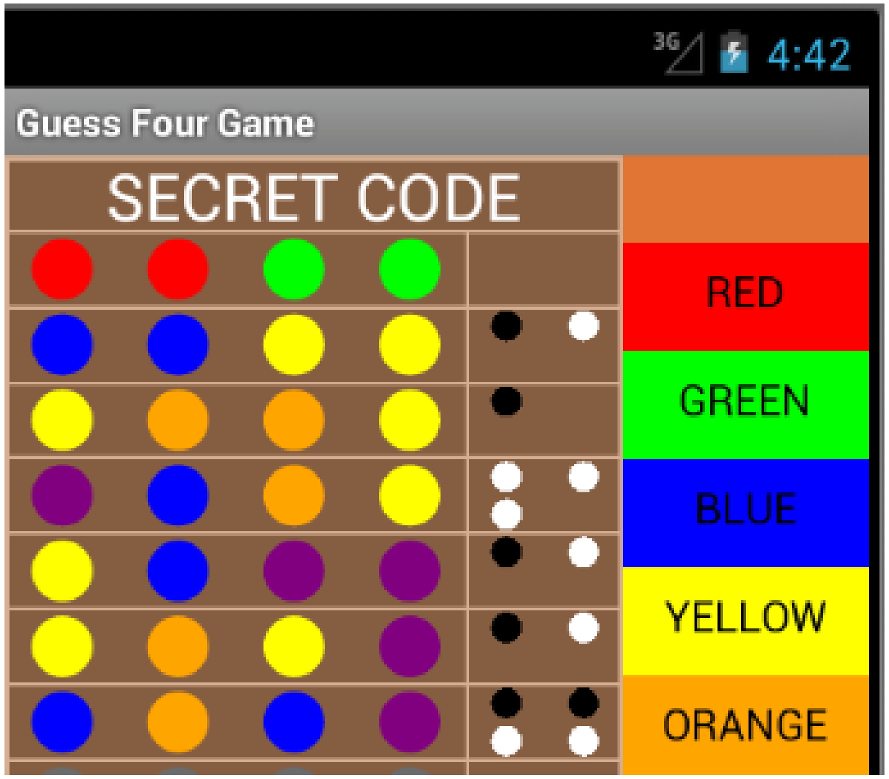
```
// radial gradient
RadialGradient rg = new RadialGradient(200, 400, 125,
        Color.BLUE, Color.GREEN, Shader.TileMode.MIRROR);
p.setShader(rg);
canvas.drawCircle(200, 325, 125, p);
```

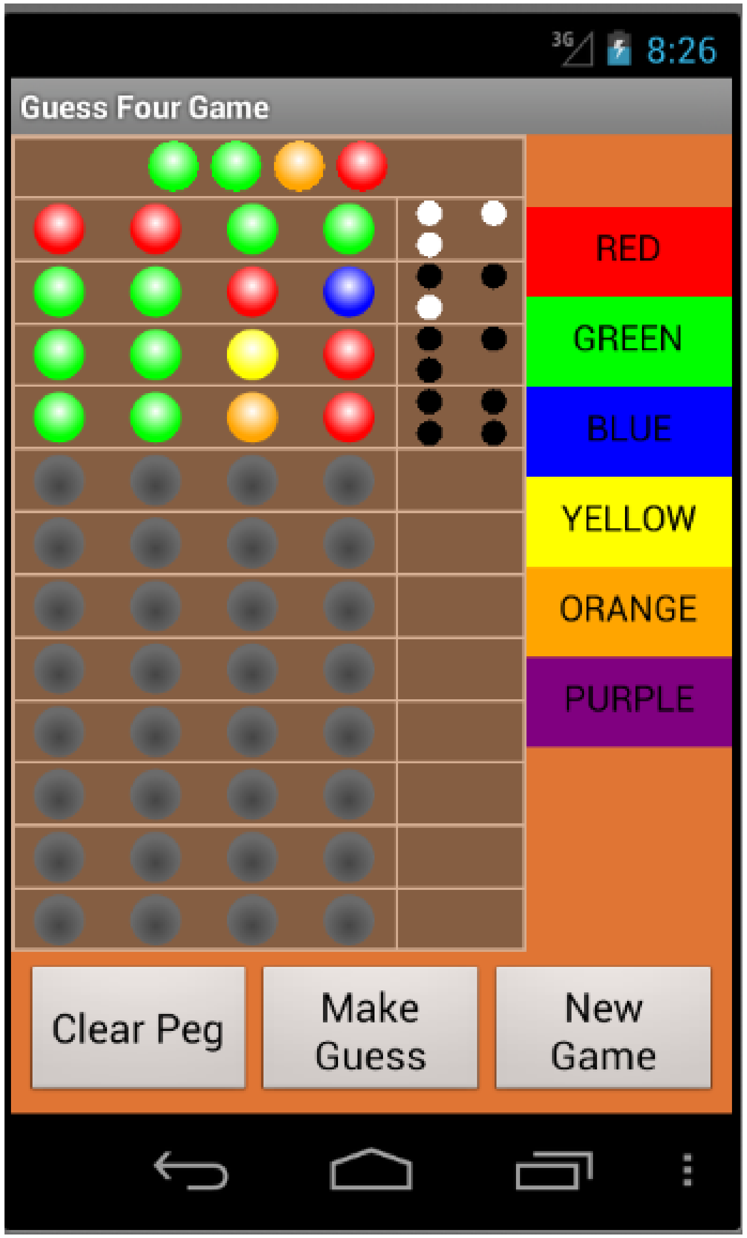
RadialGradient



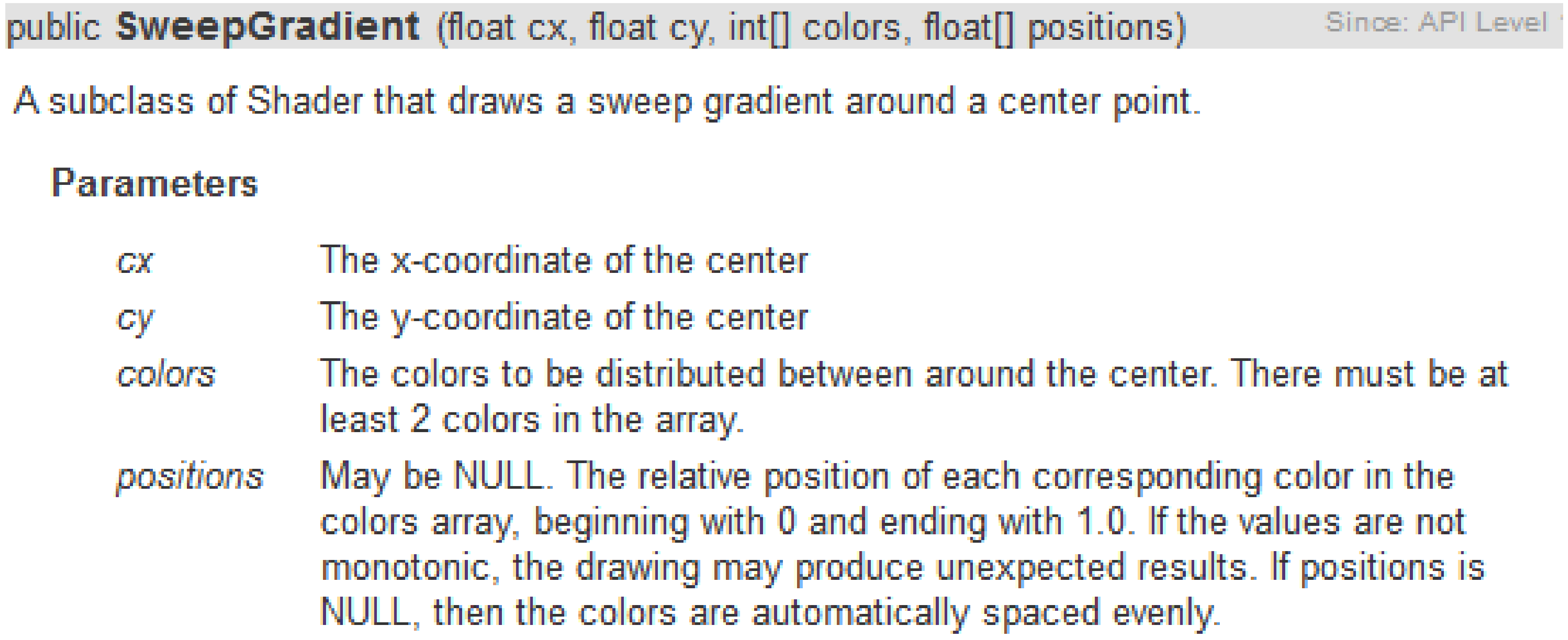
RadialGradeint

- add depth to pegs and open spots in Guess Four game





SweepGradient



public **SweepGradient** (float cx, float cy, int[] colors, float[] positions) Since: API Level

A subclass of Shader that draws a sweep gradient around a center point.

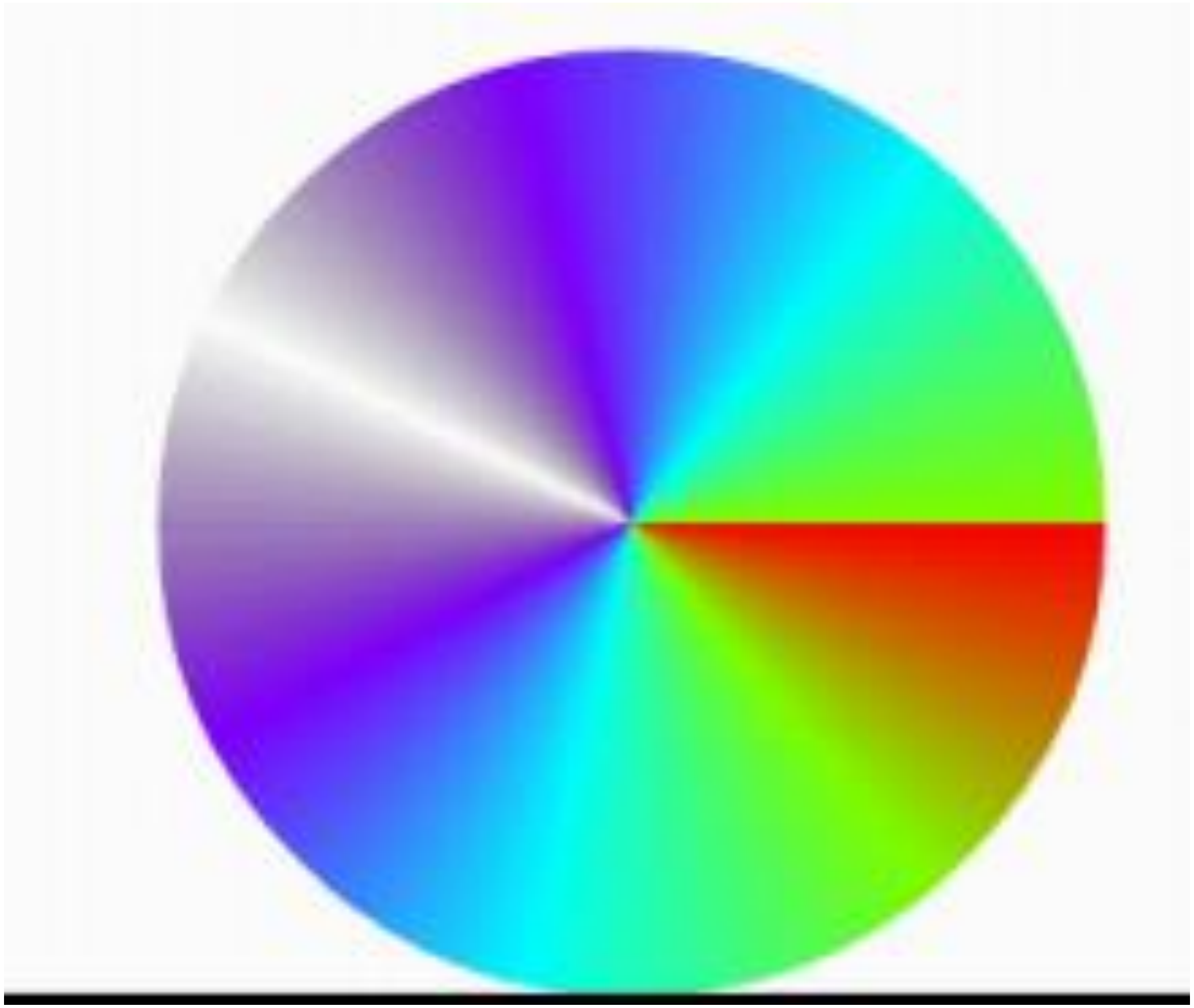
Parameters

cx	The x-coordinate of the center
cy	The y-coordinate of the center
colors	The colors to be distributed between around the center. There must be at least 2 colors in the array.
positions	May be NULL. The relative position of each corresponding color in the colors array, beginning with 0 and ending with 1.0. If the values are not monotonic, the drawing may produce unexpected results. If positions is NULL, then the colors are automatically spaced evenly.

SweepGradient

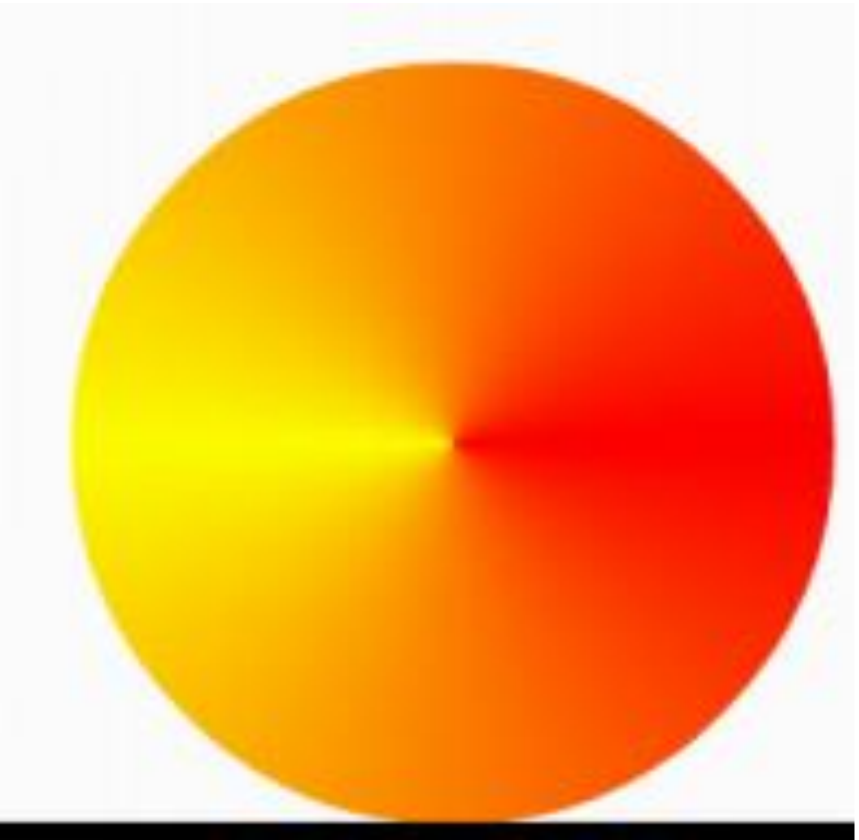
```java
// sweep gradient
int numColors = 4;
int angleIncrement = 360 / numColors;
int[] rainbow = new int[numColors * 2];
float[] hsv = {0, 1, 1};
for(int i = 0; i < rainbow.length / 2; i++) {
    rainbow[i] = Color.HSVToColor(hsv);
    hsv[0] += angleIncrement;
}
for(int i = rainbow.length / 2; i < rainbow.length; i++) {
    rainbow[i] = rainbow[rainbow.length - i];
}
SweepGradient sg = new SweepGradient(300, 600, rainbow, null);
p.setShader(sg);
canvas.drawCircle(300, 600, 125, p);
```

SweepGradient



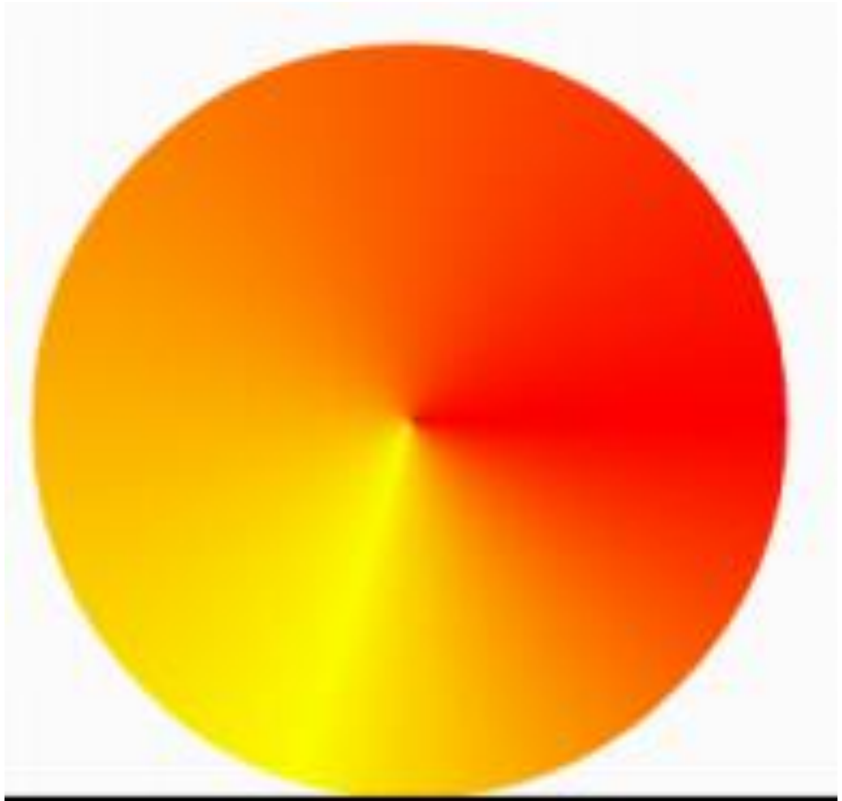
SweepGradient

```
SweepGradient sg = new SweepGradient(300, 600,
        new int[] {Color.RED, Color.YELLOW, Color.RED},
        null);
p.setShader(sg);
canvas.drawCircle(300, 600, 125, p);
```



SweepGradient

```
SweepGradient sg = new SweepGradient(300, 600,
        new int[] {Color.RED, Color.YELLOW, Color.RED},
        new float[]{0, 0.3f, 1});
p.setShader(sg);
canvas.drawCircle(300, 600, 125, p);
```

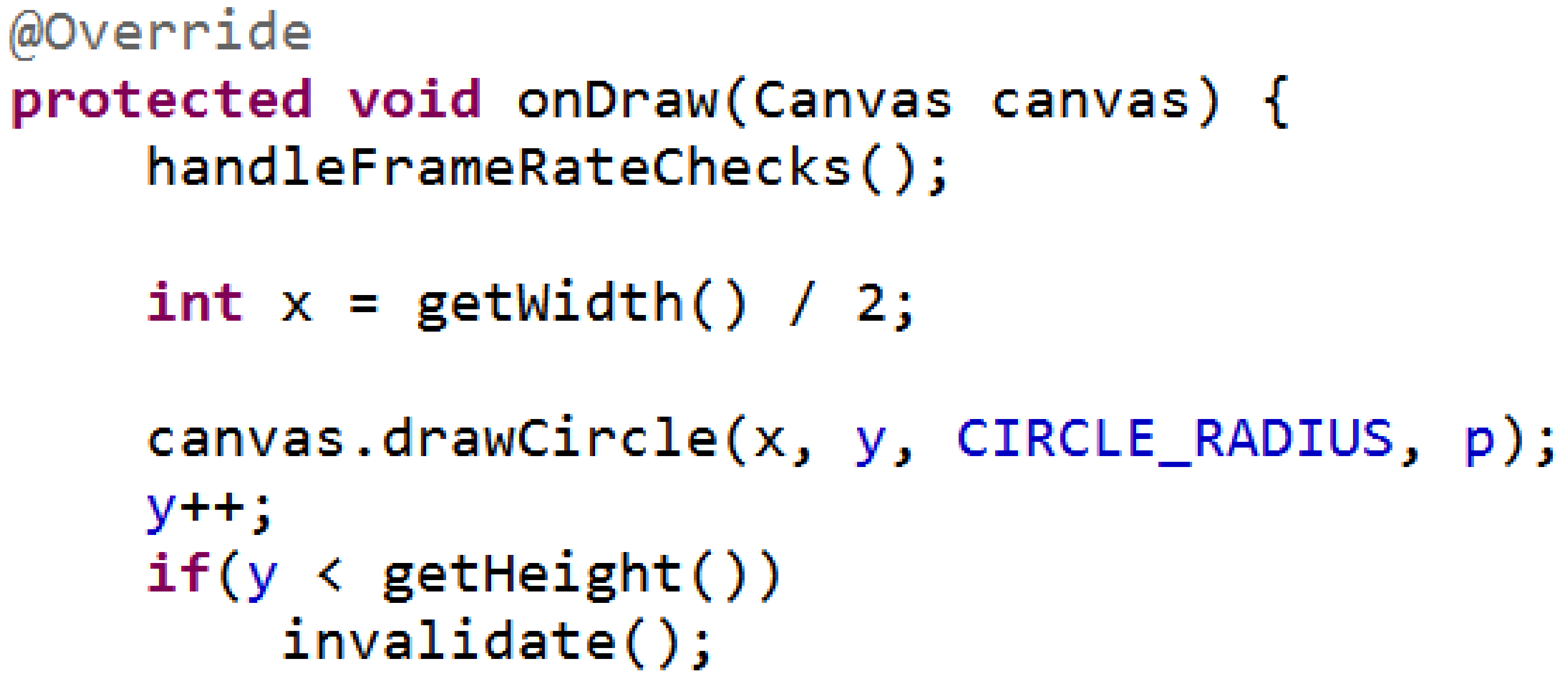


SIMPLE ANDROID ANIMATION LOOP

Simple Animation

- Animation altering some property of a 2D primitive
  - position
  - size
  - color
  - alpha
- Simplest animation loop, after onDraw, call invalidate
  - at the mercy of the UI frame rate

Simple (And Poor) Animation Approach



```
@Override
protected void onDraw(Canvas canvas) {
    handleFrameRateChecks();

    int x = getWidth() / 2;

    canvas.drawCircle(x, y, CIRCLE_RADIUS, p);
    y++;
    if(y < getHeight())
        invalidate();
```

- draw as fast as possible
  - emulator frame rate (on my machine) 12 fps
  - dev device frame rate, high 50s fps

Better Animation Loop

- Create a Handler to delay / sleep
- update method will call sleep method on the Handler

```
private RefreshHandler mRedrawHandler = new RefreshHandler();

class RefreshHandler extends Handler {

    @Override
    public void handleMessage(Message msg) {
        GraphicsView.this.update();
        GraphicsView.this.invalidate();
    }

    public void sleep(long delayMillis) {
        this.removeMessages(0);
        sendMessageDelayed(obtainMessage(0), delayMillis);
    }
};
```

update method

```
public void update() {

    if (mode == RUNNING) {
        handleFrameRateChecks();
        long now = System.currentTimeMillis();
        if (now - prevTime > moveDelay) {
            prevTime = now;
            x = getWidth() / 2;
            y += SPEED;
            if(y > getHeight())
                mode = STOPPED;
        }
        mRedrawHandler.sleep(moveDelay);
    }

}
```

- animation loop
  - update tells handler to sleep
  - handler calls update when it wakes up …

Simple Animation Example
Add Custom View to XML

- in main.xml
- add custom View as element in LinearLayout

Class Name

```
<scottm.examples.BalloonView
    android:id="@+id/graphics_view"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:text="@string/hello"
    android:background="#fff"/>
```

RGB Color, using hexadecimal

Simple Example - BalloonView

```
public class BalloonView extends View {

    private static final String TAG = "BalloonView";


    private static final int MAX_BALLOONS = 20;
    private static final
        double newBalloonsPerFrame = 1.0 / 25;

    private Paint paint;

    private HashSet<Balloon> balloons;
    private Random random;
```

BalloonView Constructors

```
public BalloonView(Context context) {
    super(context);
    initialize();
}

public BalloonView (Context context, AttributeSet attrs) {
    super(context, attrs);
    initialize();

}

public BalloonView(Context context, AttributeSet attrs,
                        int defStyle) {
    super(context, attrs, defStyle);
    initialize();
}
```

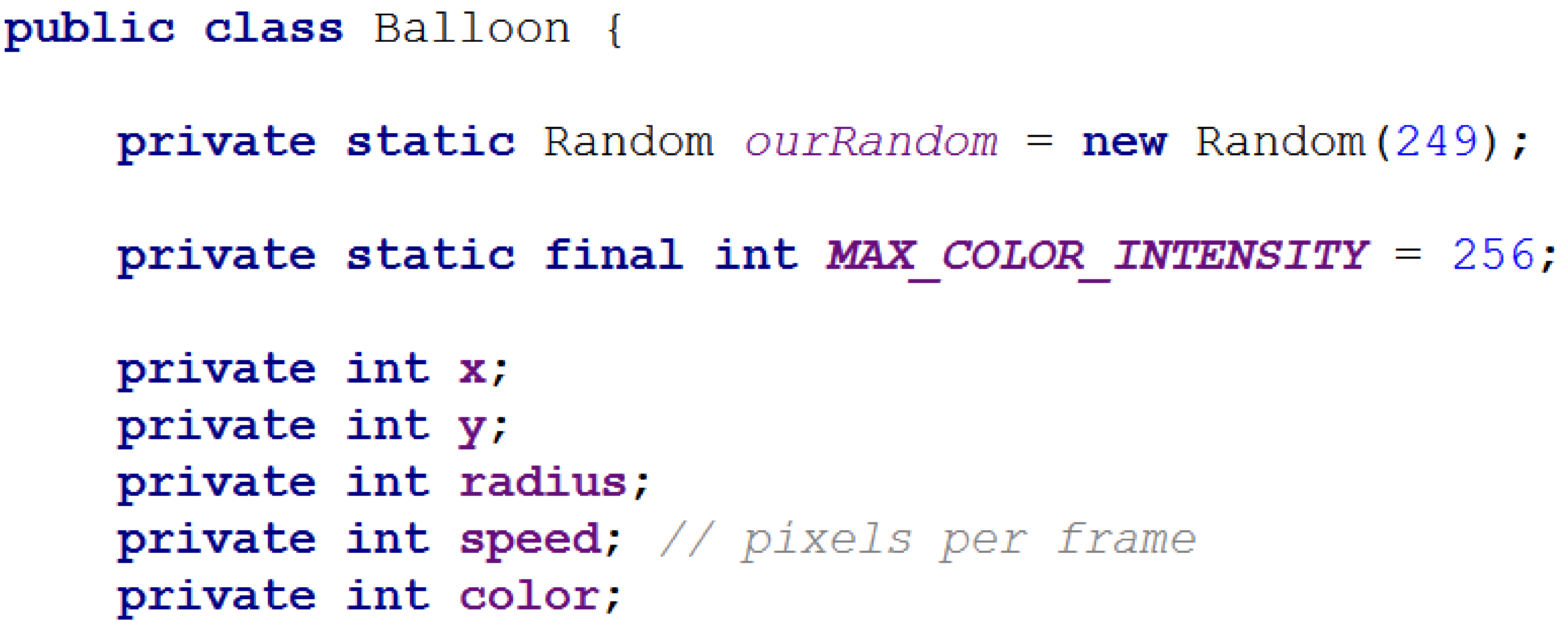
BalloonView

- Tracks "balloons" on screen
  - Balloon class to track state of each balloon
- Add balloons randomly
- Update balloons when instructed
  - change position
  - remove if off screen

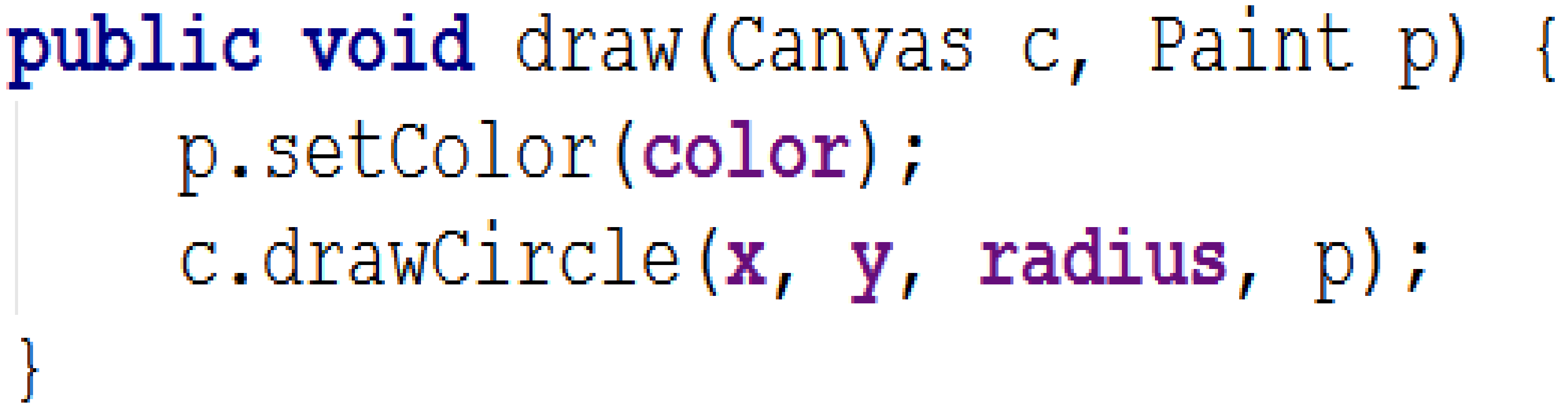
BalloonView onDraw Method

```
@Override
protected void onDraw(Canvas canvas) {
    int width = canvas.getWidth();
    int height = canvas.getHeight();
    updateBalloons();
    double randomValue = random.nextDouble();
    if (balloons.size() == 0
            || randomValue < newBalloonsPerFrame) {
        addBalloon(height, width);
    }
    for (Balloon b : balloons) {
        b.draw(canvas, paint);
    }
}
```

Balloon Class and draw Method

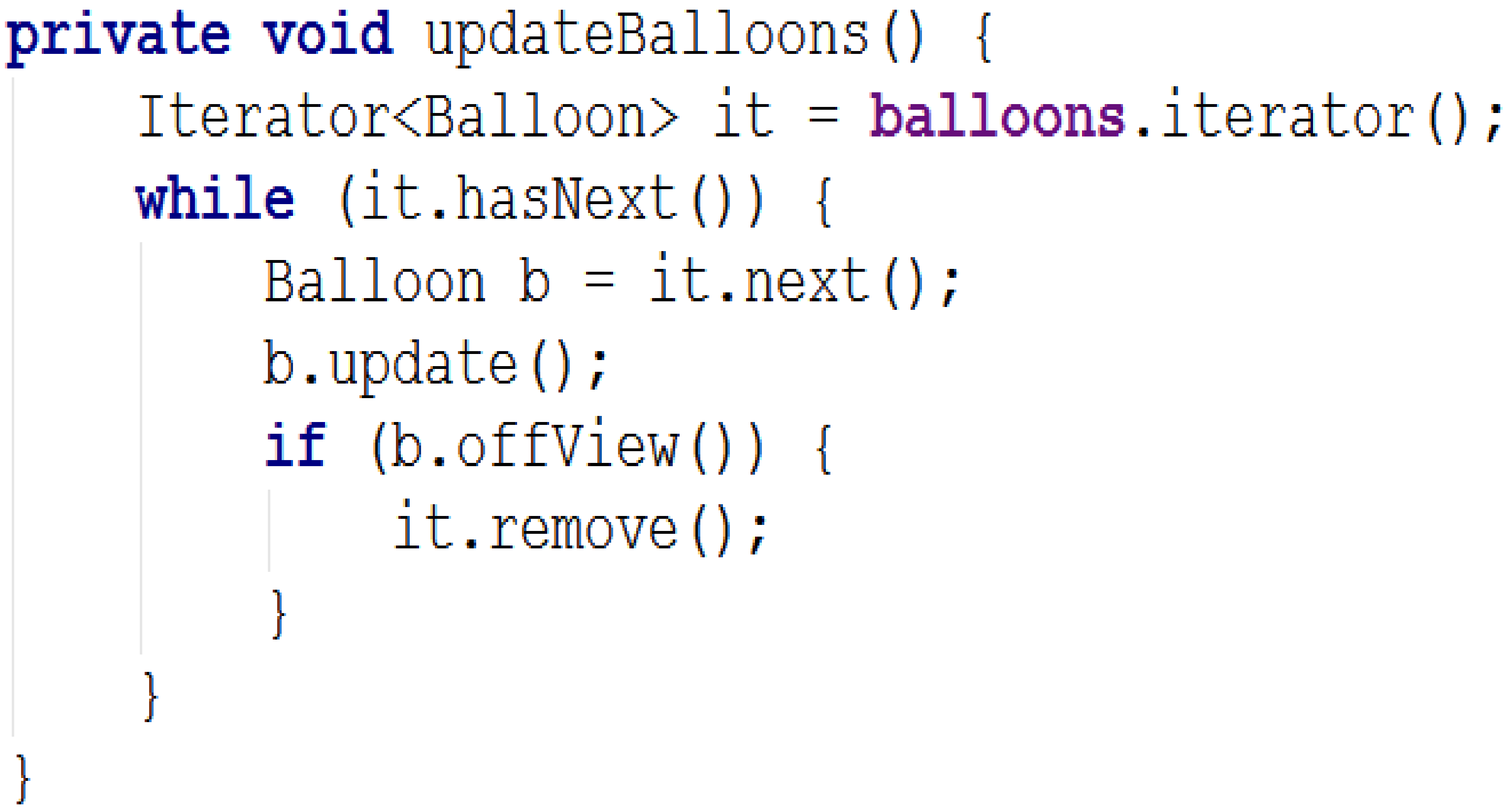


```
public class Balloon {

    private static Random ourRandom = new Random(249);

    private static final int MAX_COLOR_INTENSITY = 256;

    private int x;
    private int y;
    private int radius;
    private int speed; // pixels per frame
    private int color;
```



```
public void draw(Canvas c, Paint p) {
    p.setColor(color);
    c.drawCircle(x, y, radius, p);
}
```

BalloonView update Method



```
private void updateBalloons() {
    Iterator<Balloon> it = balloons.iterator();
    while (it.hasNext()) {
        Balloon b = it.next();
        b.update();
        if (b.offView()) {
            it.remove();
        }
    }
}
```

Animation Loop

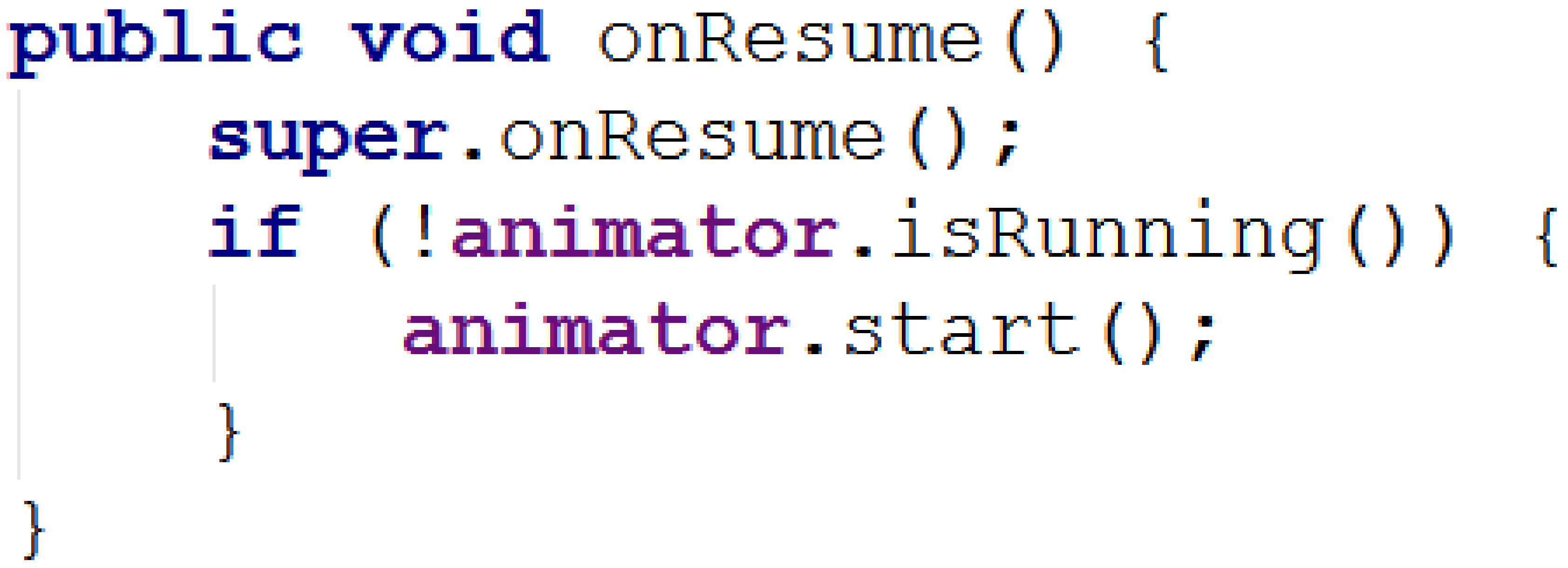
- Activity has AnimationLoop object
  - non standard Android class to simplify animation loops
- AnimationLoop given target frames per second and View to update

```
private BalloonView bv;
private AnimationLoop animator;
private final int FPS = 50;
```

```
// in onCreate for Activity
animator = new AnimationLoop(bv, FPS);
```

Animation Loop

- When Activity resumes, animation loop started



```java
public void onResume() {
    super.onResume();
    if (!animator.isRunning()) {
        animator.start();
    }
}
```

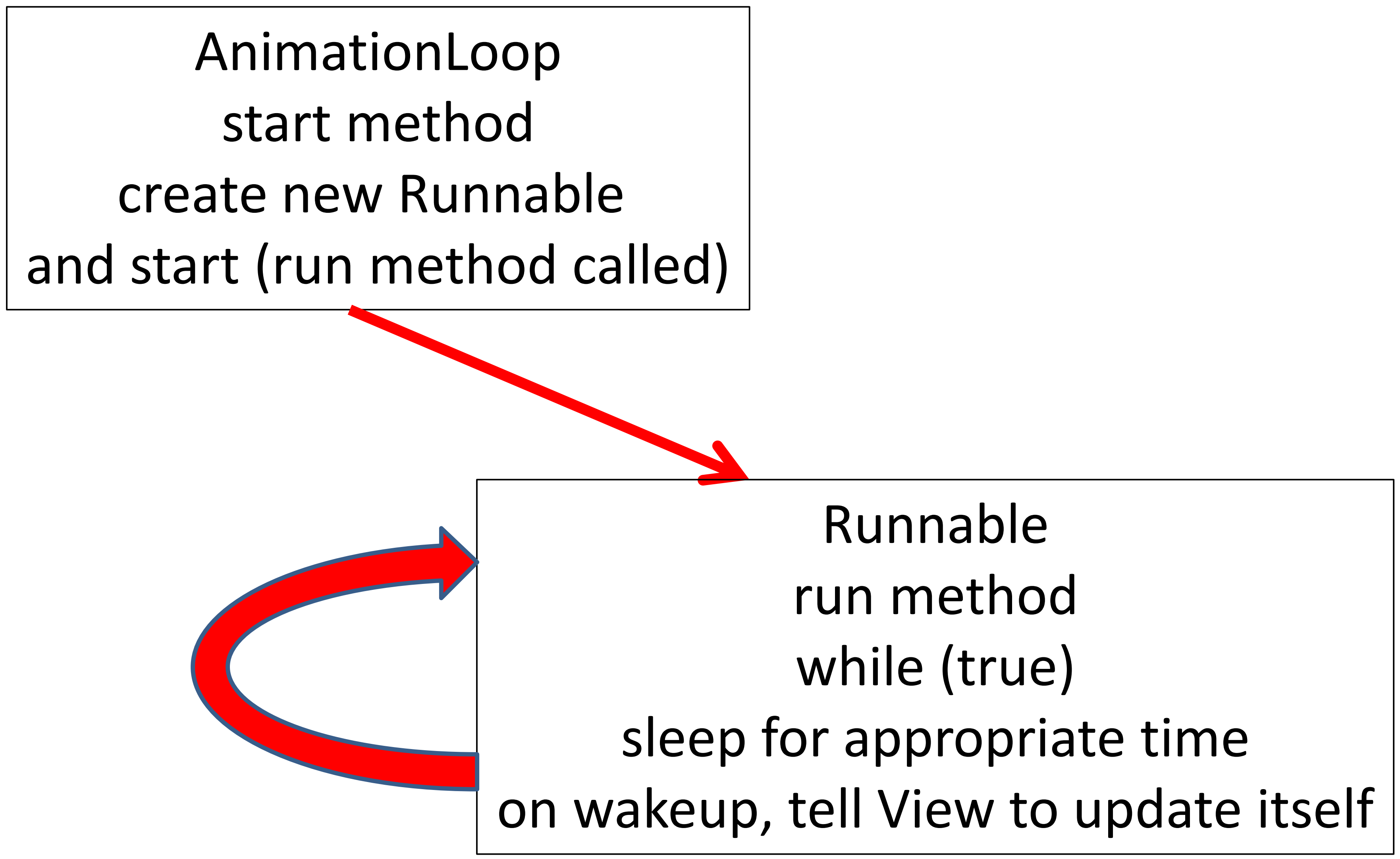
Pausing

- Animation Loop object paused when Activity paused
- toggle animation when activity is clicked
  - create on click listener for activity

Animation Loop

- Animation Loop has a Runnable and a Handler
- Runnable: Any class whose instances are intended to be executed by a thread
  - standard Java class
- Handler: Allows you to send and process Message and Runnable objects associated with a thread's Message queue.
  - schedule messages and runnables to be executed as some point in the future
  - Android specific class (there is a class named Handler in the Java standard library)

AnimationLoop


AnimationLoop
start method
create new Runnable
and start (run method called)
Runnable
run method
while (true)
sleep for appropriate time
on wakeup, tell View to update itself

TWEENED ANIMATIONS

Simple Animations

- Tweened Animations
  - also know as View Animations
- provide a way to perform simple animations on Views, Bitmaps, TextViews, Drawables
- provide start point, end point, size, rotation, transparency, other properties
- Can set up tweened animation in XML or programmatically

Tweened Animations

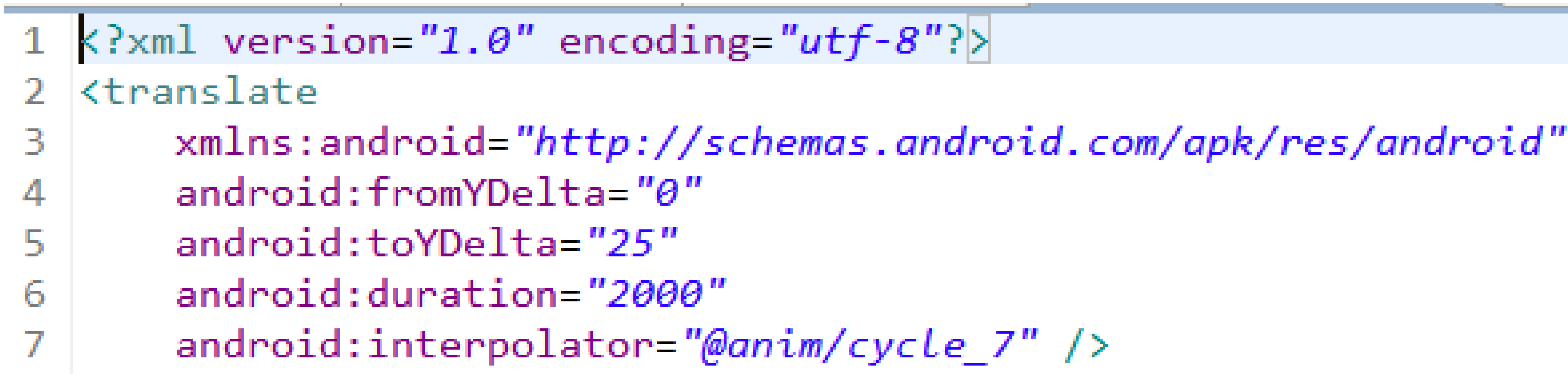
- Used to alter one of four properties of a single View affect
  - Alpha (transparency / opaqueness)
  - Rotation
  - Scale (size)
  - Location (movement, Translate)

Tweened Animations

- define interpolator in XML (optional)
  - allow animation to be repeated, accelerate, decelerate, "bounce", and more
  - can use built in interpolators or create your own
- define animation in XML
  - alpha, rotate, scale, translate
  - define **from value** and **to value** and **duration**
- In program load and start animation in response to some event
  - Guess Four, invalid choice, solved puzzle

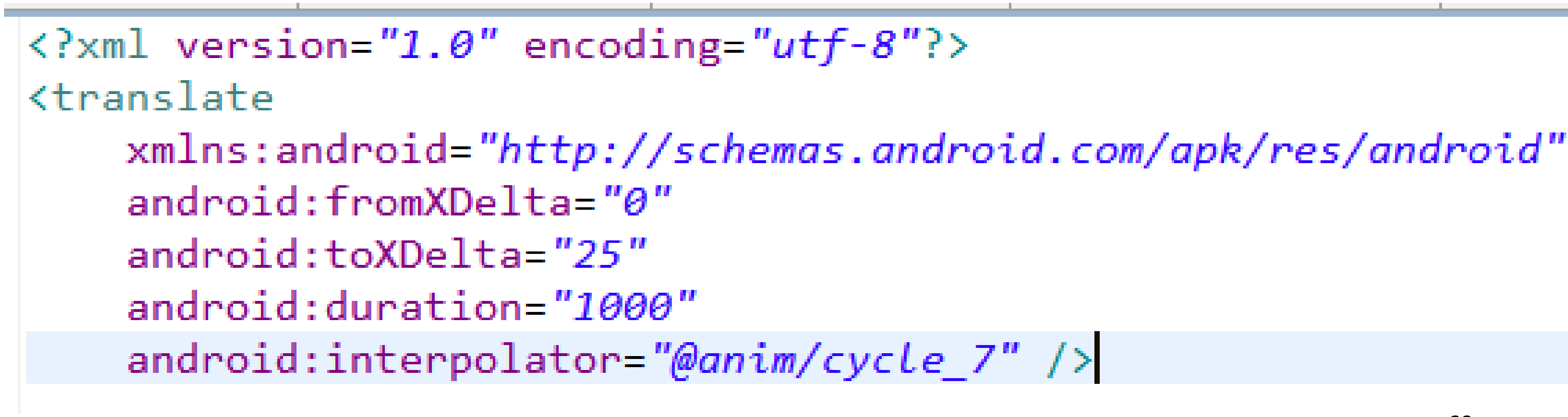
Guess Four res/anim

- shake up down



```
<?xml version="1.0" encoding="utf-8"?>
<translate
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromYDelta="0"
    android:toYDelta="25"
    android:duration="2000"
    android:interpolator="@anim/cycle_7" />
```

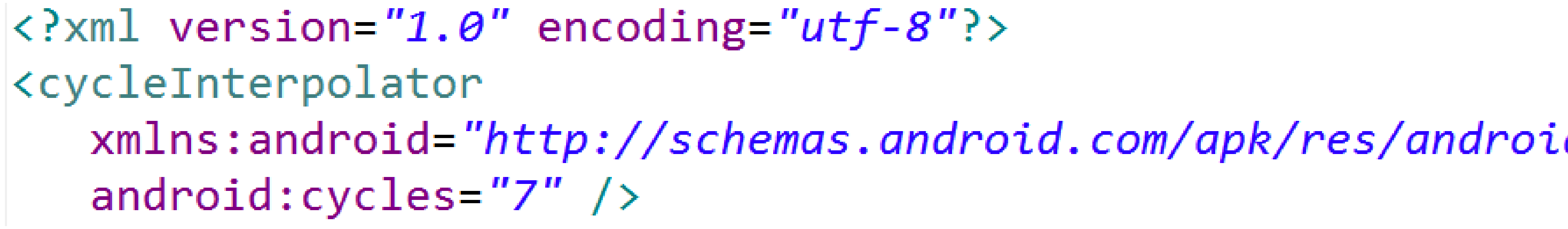
shake left right



```
<?xml version="1.0" encoding="utf-8"?>
<translate
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromXDelta="0"
    android:toXDelta="25"
    android:duration="1000"
    android:interpolator="@anim/cycle_7" />
```

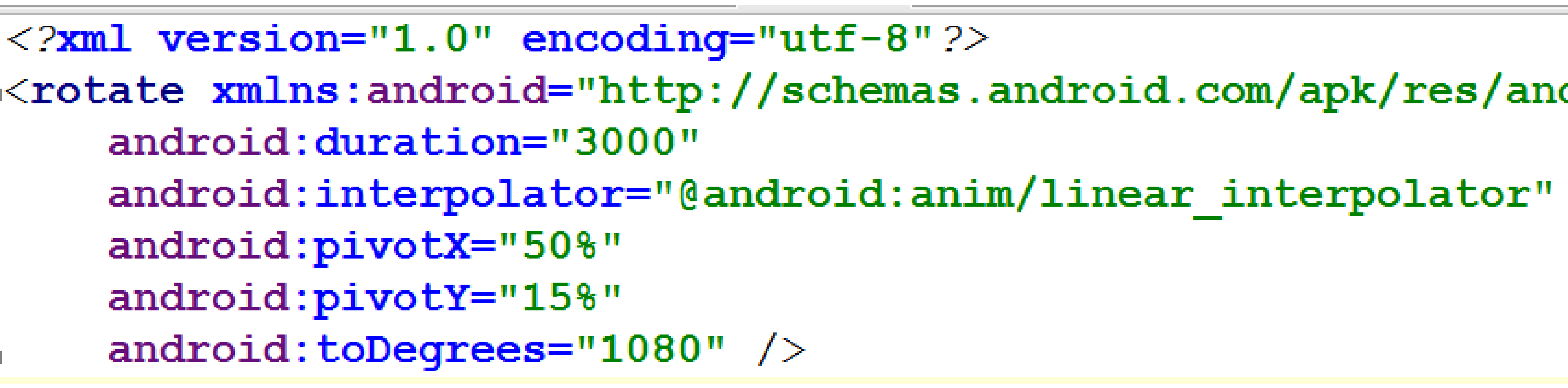
GuessFour res/anim

cycle_7.xml



```
<?xml version="1.0" encoding="utf-8"?>
<cycleInterpolator
   xmlns:android="http://schemas.android.com/apk/res/androi
   android:cycles="7" />
```

spin.xml



```
<?xml version="1.0" encoding="utf-8"?>
<rotate xmlns:android="http://schemas.android.com/apk/res/and
    android:duration="3000"
    android:interpolator="@android:anim/linear_interpolator"
    android:pivotX="50%"
    android:pivotY="15%"
    android:toDegrees="1080" />
```

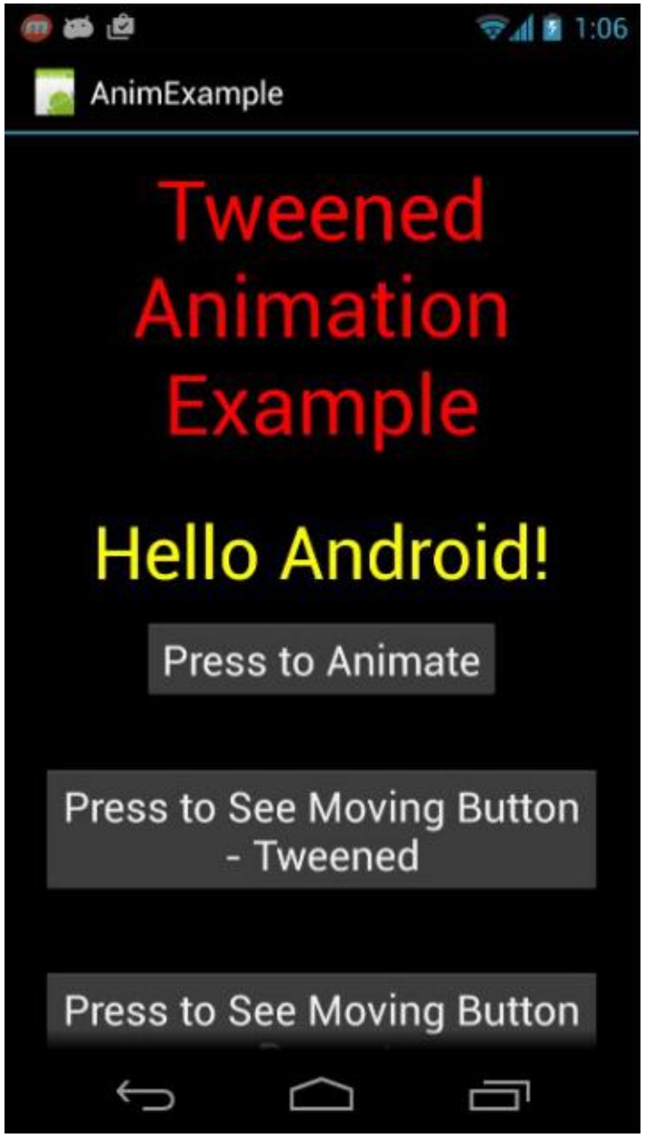
GuessFour Example

- On error board shakes back and forth
- On win board spins
- From BoardView in GuessFour

```
public void shakeLeftRight() {
    Log.d(TAG, "in shake! Trying to start animation!");
    startAnimation(AnimationUtils.loadAnimation(game, R.anim.shake));
}

public void shakeUpDown() {
    Log.d(TAG, "in shake! Trying to start animation!");
    startAnimation(AnimationUtils.loadAnimation(game, R.anim.shake_up_down));
}

public void spin() {
    Log.d(TAG, "in shake! Trying to start animation!");
    startAnimation(AnimationUtils.loadAnimation(game, R.anim.spin));
}
```

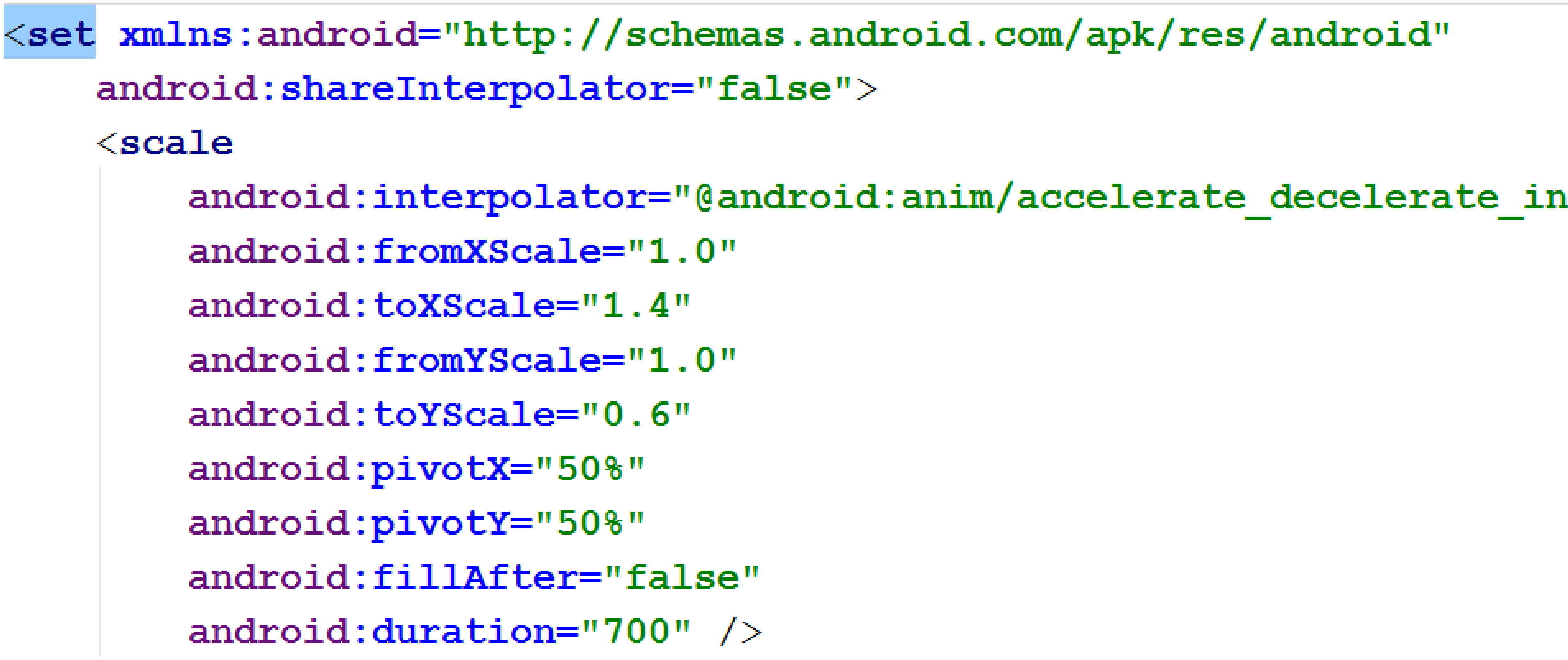
More Tweened Examples

- hyperspace example from android dev site
- rotate and change alpha
- animation types:
  - alpha
  - scale
  - translate
  - rotate


1:06
AnimExample
Tweened Animation Example
Hello Android!
Press to Animate
Press to See Moving Button - Tweened
Press to See Moving Button

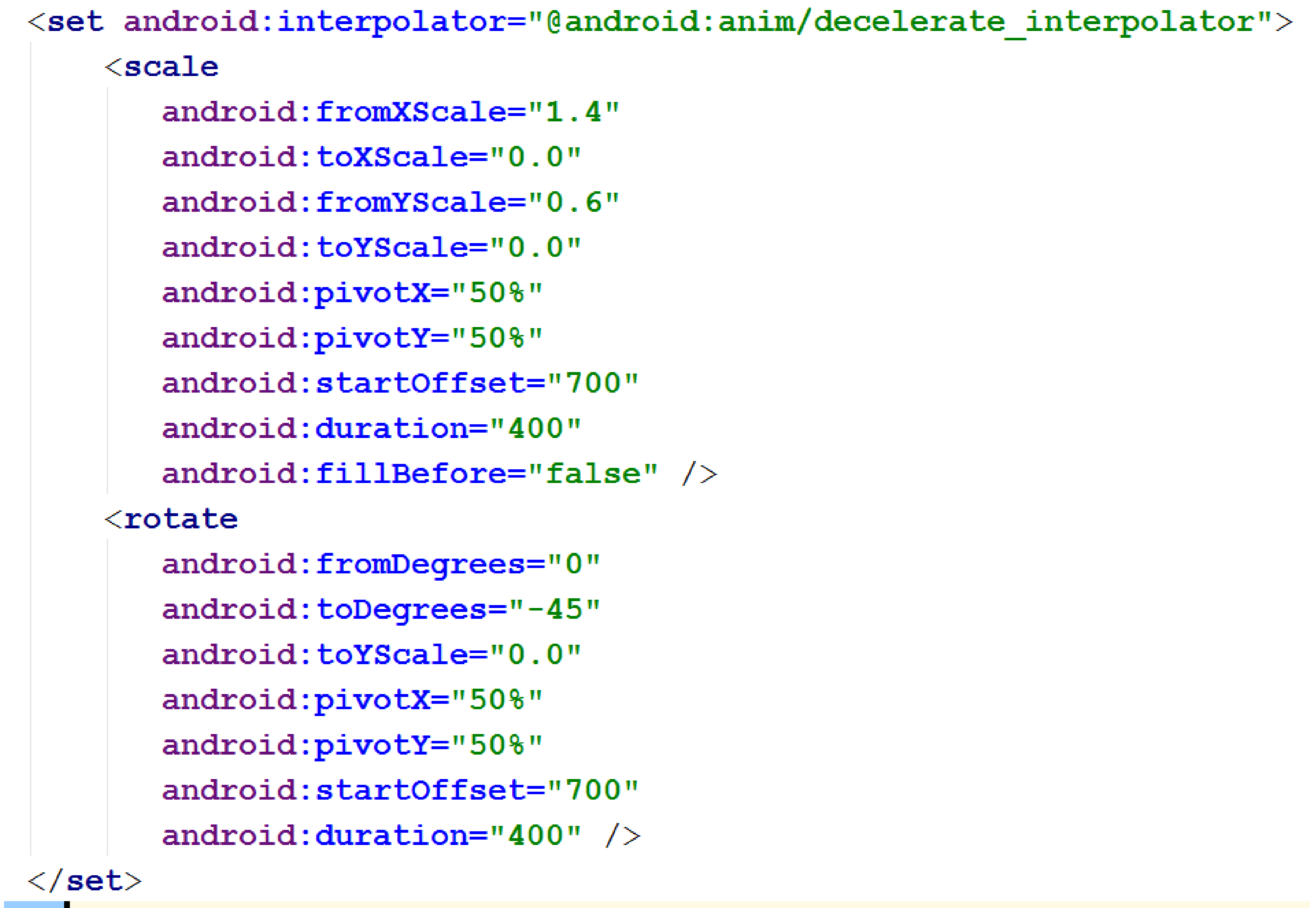
http://developer.android.com/guide/topics/resources/animation-resource.html

Hyperspace Part 1



```
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="false">
    <scale
        android:interpolator="@android:anim/accelerate_decelerate_in
        android:fromXScale="1.0"
        android:toXScale="1.4"
        android:fromYScale="1.0"
        android:toYScale="0.6"
        android:pivotX="50%"
        android:pivotY="50%"
        android:fillAfter="false"
        android:duration="700" />
```

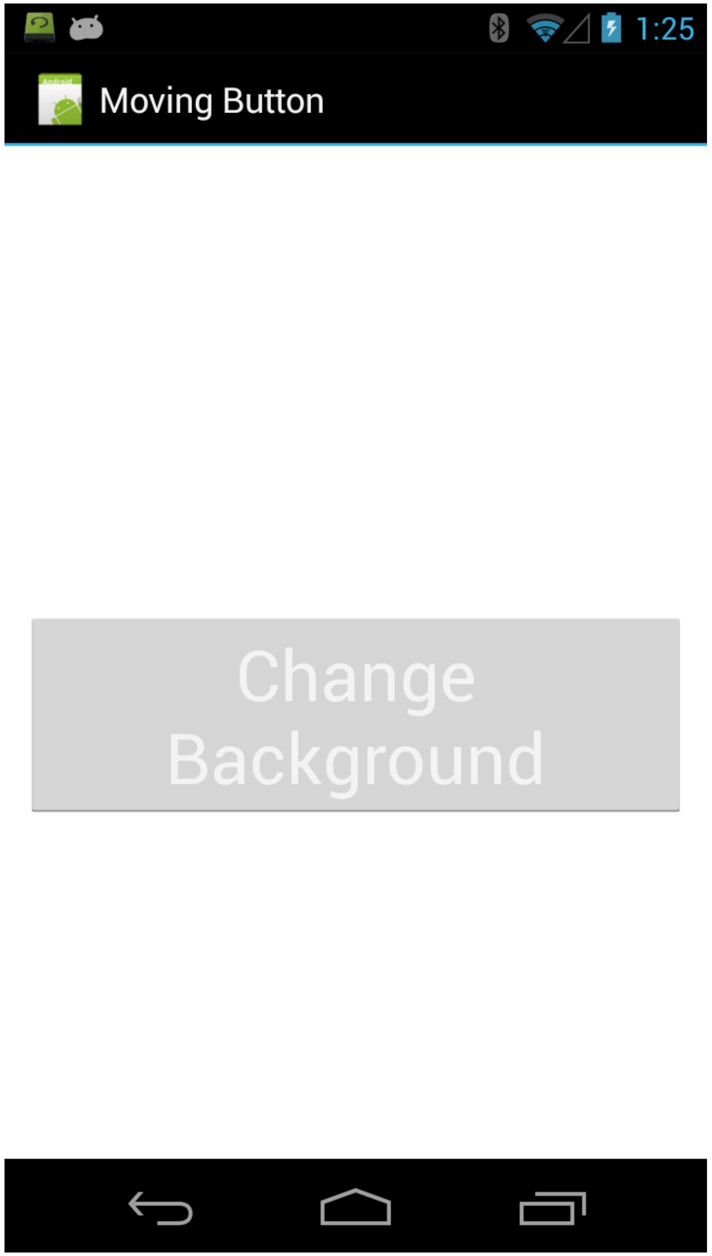
Hyperspace Part 2



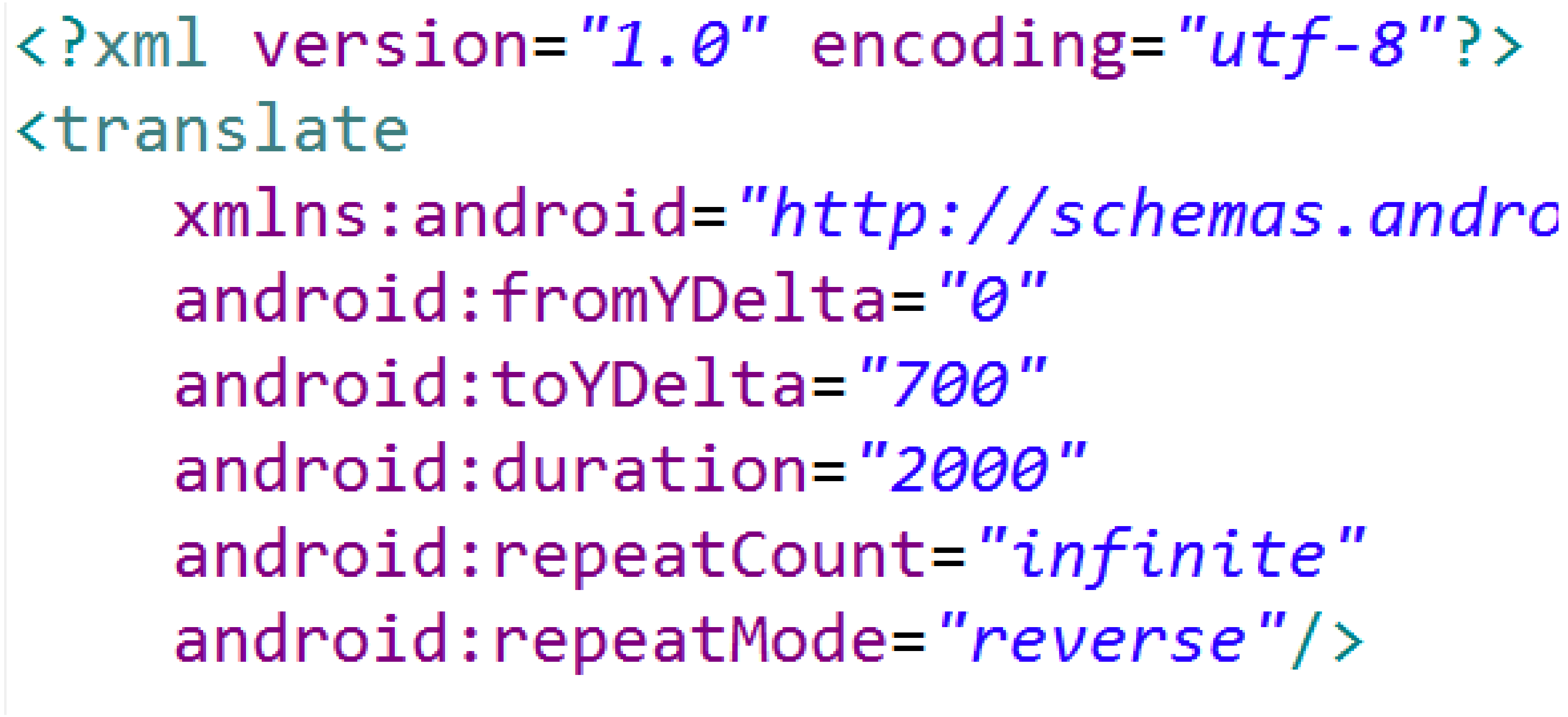
```
<set android:interpolator="@android:anim/decelerate_interpolator">
    <scale
       android:fromXScale="1.4"
       android:toXScale="0.0"
       android:fromYScale="0.6"
       android:toYScale="0.0"
       android:pivotX="50%"
       android:pivotY="50%"
       android:startOffset="700"
       android:duration="400"
       android:fillBefore="false" />
    <rotate
       android:fromDegrees="0"
       android:toDegrees="-45"
       android:toYScale="0.0"
       android:pivotX="50%"
       android:pivotY="50%"
       android:startOffset="700"
       android:duration="400" />
</set>
```

More Tweened Examples

- Moving Button
- **Note**, tweened animations draw the button in a different spot
- But, the button's location does not actually change



Up and Down Animation



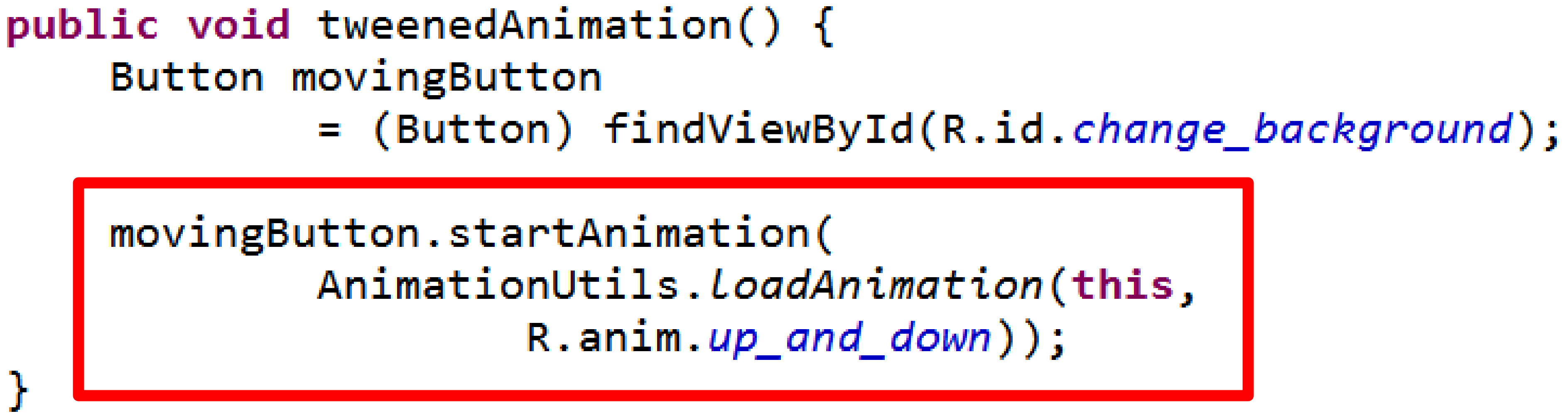
```
<?xml version="1.0" encoding="utf-8"?>
<translate
    xmlns:android="http://schemas.andro
    android:fromYDelta="0"
    android:toYDelta="700"
    android:duration="2000"
    android:repeatCount="infinite"
    android:repeatMode="reverse"/>
```

Moving Button Activity

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.button_move);
    randNumGen = new Random();
    Intent intent = getIntent();
    int animationType = TWEEN;
    if(intent.hasExtra(ANIMATION_TAG)) {
        animationType = intent.getExtras().getInt(ANIMATION_TAG);
    }
    if(animationType == TWEEN)
        tweenedAnimation();
    else
        propertyAnimation();
}
```

Use extra from Intent to determine what type of animation to perform.

Tweened Animation



```
public void tweenedAnimation() {
    Button movingButton
            = (Button) findViewById(R.id.change_background);
    movingButton.startAnimation(
            AnimationUtils.loadAnimation(this,
                    R.anim.up_and_down));
}
```

Change Background Color

- Called when button clicked
  - onClick attribute

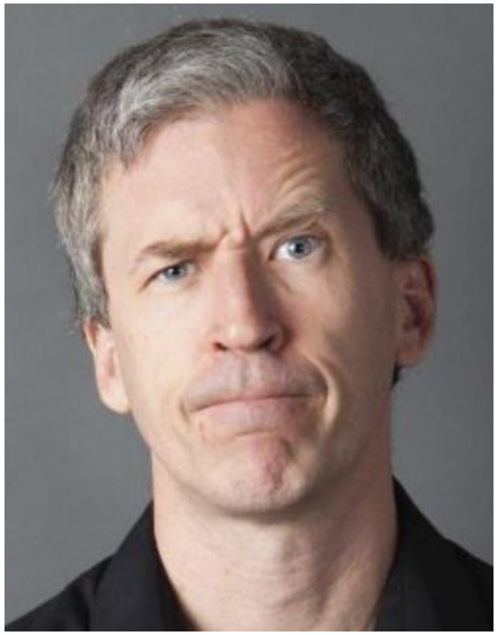
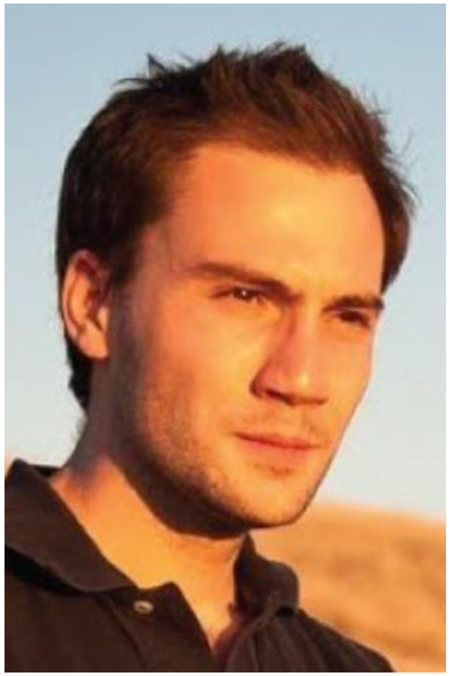
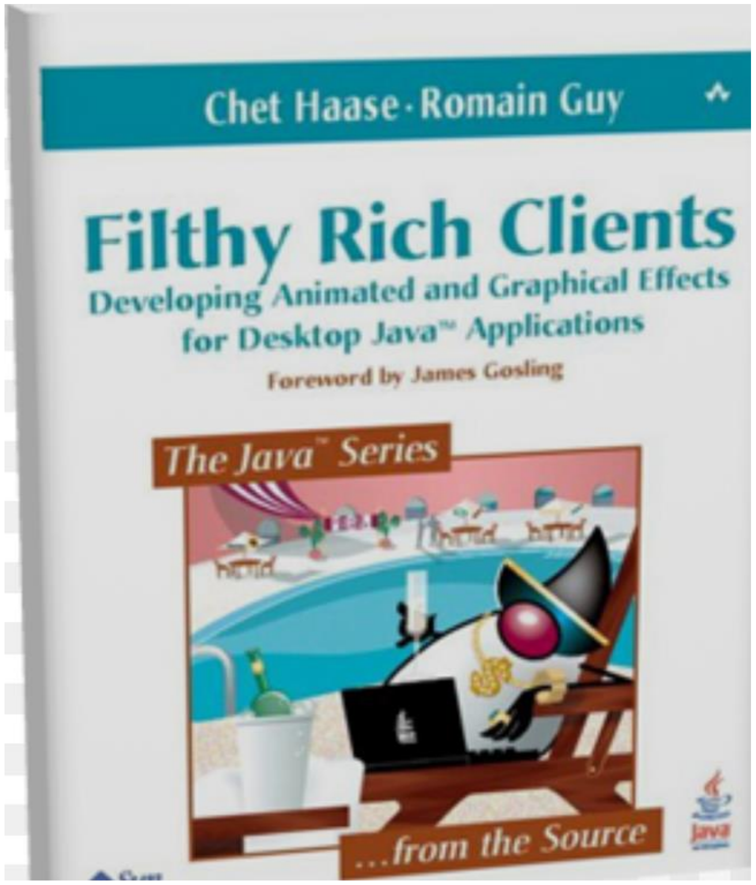
```
public void changeBackground(View v) {
    View target = (View) findViewById(R.id.linear_layout_button);
    int red = randNumGen.nextInt(NUM_SHADES );
    int green = randNumGen.nextInt(NUM_SHADES );
    int blue = randNumGen.nextInt(NUM_SHADES );
    target.setBackgroundColor(Color.argb(255, red, green, blue));
}
```

Result?

PROPERTY ANIMATIONS

-AVAILABLE POST GINGERBREAD, ANDOID 3.0, API LEVEL 11

- Developer of new animation framework for Android, Chet Haase and Romain Guy




Chet Haase · Romain Guy
Filthy Rich Clients
Developing Animated and Graphical Effects for Desktop Java™ Applications
Foreword by James Gosling
The Java™ Series
...from the Source

Property Animations

- A more general animation framework
- Tweened Animations can only affect alpha, scale, rotation, and position
- Property Animations can affect any property of an object
  - typically a View or Drawable
- can be defined in sets, change multiple properties at a time
- animation created separate from target object, reuse with different objects

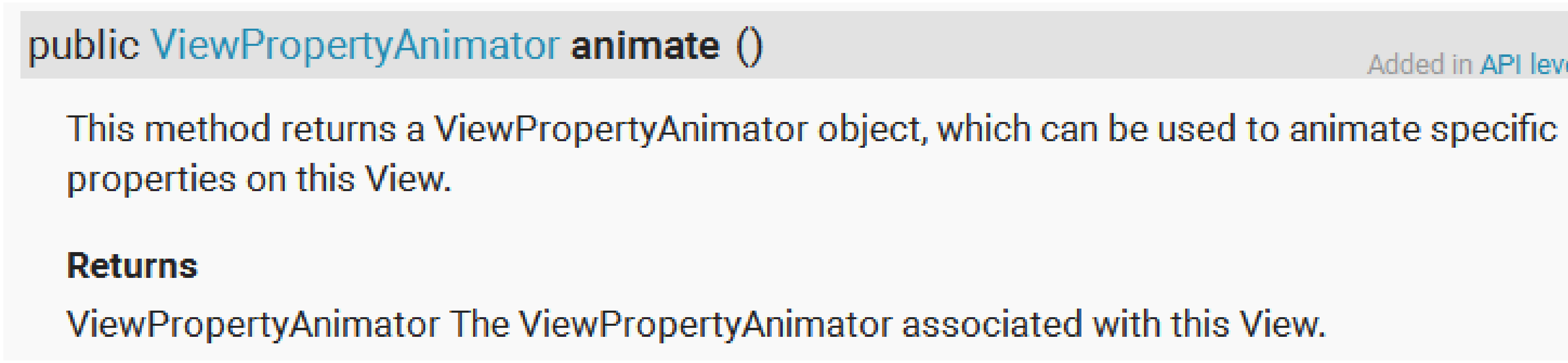
http://developer.android.com/guide/topics/graphics/prop-animation.html

Property Animation - Classes

- ValueAnimator
  - base class for property animations
- Object Animator
  - convenience class for animating specific object and property
- ViewPropertyAnimator
  - optimized, simple animation of View objects
- evaluator classes such as ArgbEvaluator to animate property by defining how it changes over time

Some Animations Simple

- API levels 11+
- View objects have animate() method



public ViewPropertyAnimator **animate** ()

Added in API lev

This method returns a ViewPropertyAnimator object, which can be used to animate specific properties on this View.

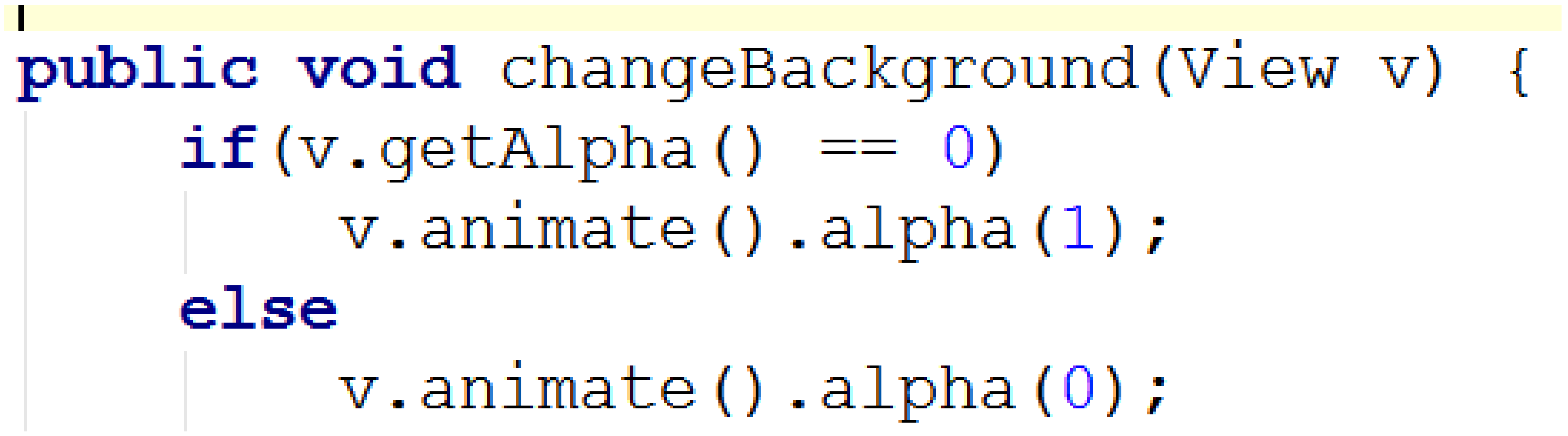
Returns

ViewPropertyAnimator The ViewPropertyAnimator associated with this View.

- ViewPropertyAnimator
- methods for alpha, rotation, scale (size), translation (position)

ViewPropertyAnimator Example

- onClick method for a button:

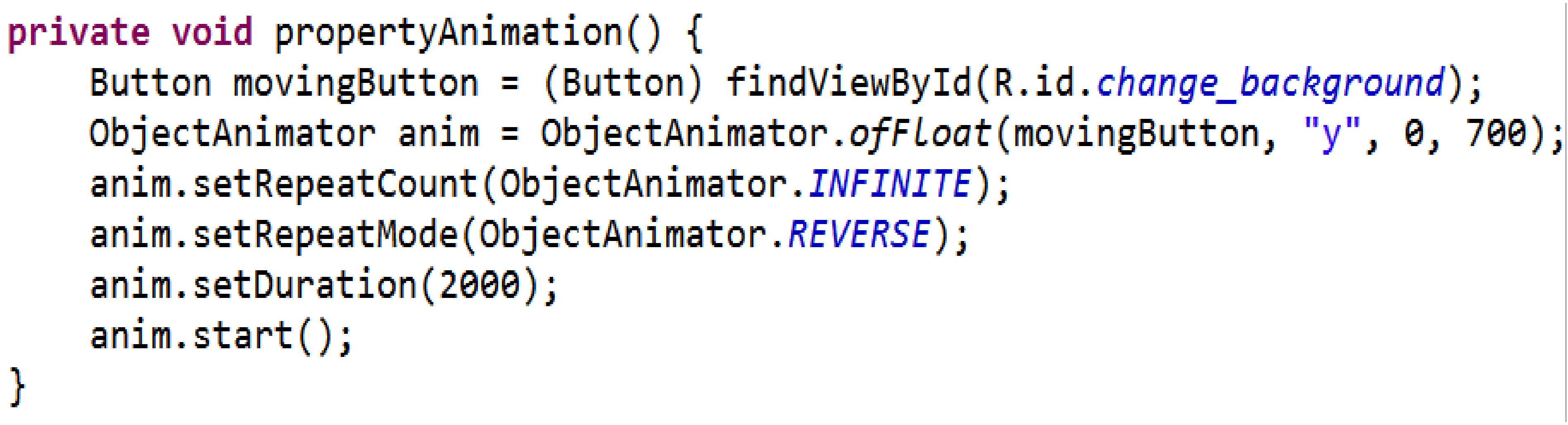


```
public void changeBackground(View v) {
    if(v.getAlpha() == 0)
        v.animate().alpha(1);
    else
        v.animate().alpha(0);
```

- button will disappear and then reappear next time clicked

More Complex Property Animation

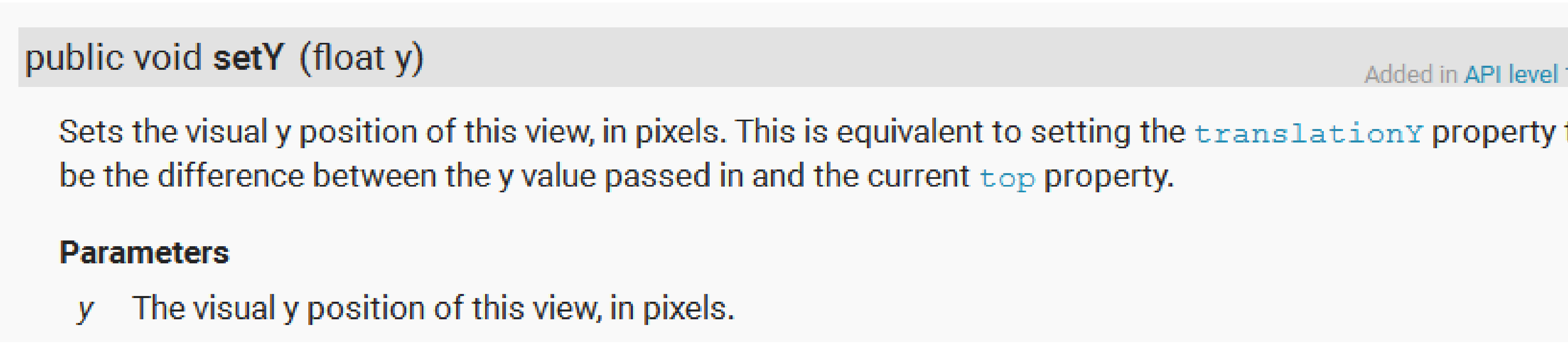
- Object animation example
- from moving button example



```
private void propertyAnimation() {
    Button movingButton = (Button) findViewById(R.id.change_background);
    ObjectAnimator anim = ObjectAnimator.ofFloat(movingButton, "y", 0, 700);
    anim.setRepeatCount(ObjectAnimator.INFINITE);
    anim.setRepeatMode(ObjectAnimator.REVERSE);
    anim.setDuration(2000);
    anim.start();
}
```

- **<u>animated class must have a "set<Property>" method</u>**

Button Class



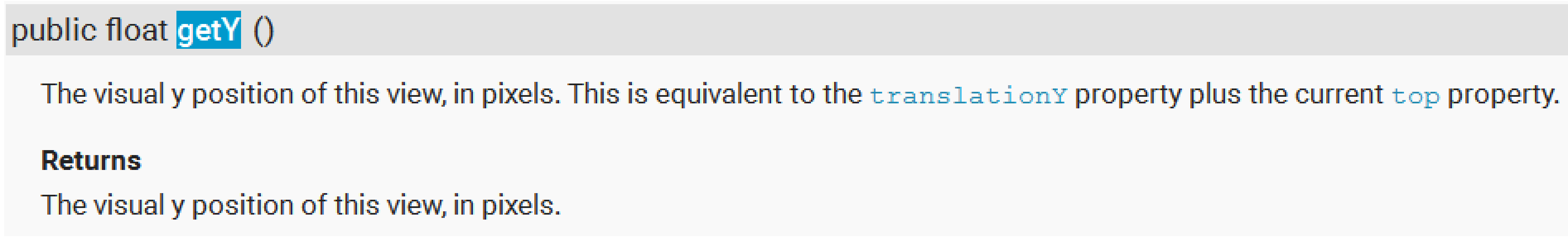
public void **setY** (float y)

Added in API level

Sets the visual y position of this view, in pixels. This is equivalent to setting the `translationY` property t
be the difference between the y value passed in and the current `top` property.

Parameters

y The visual y position of this view, in pixels.



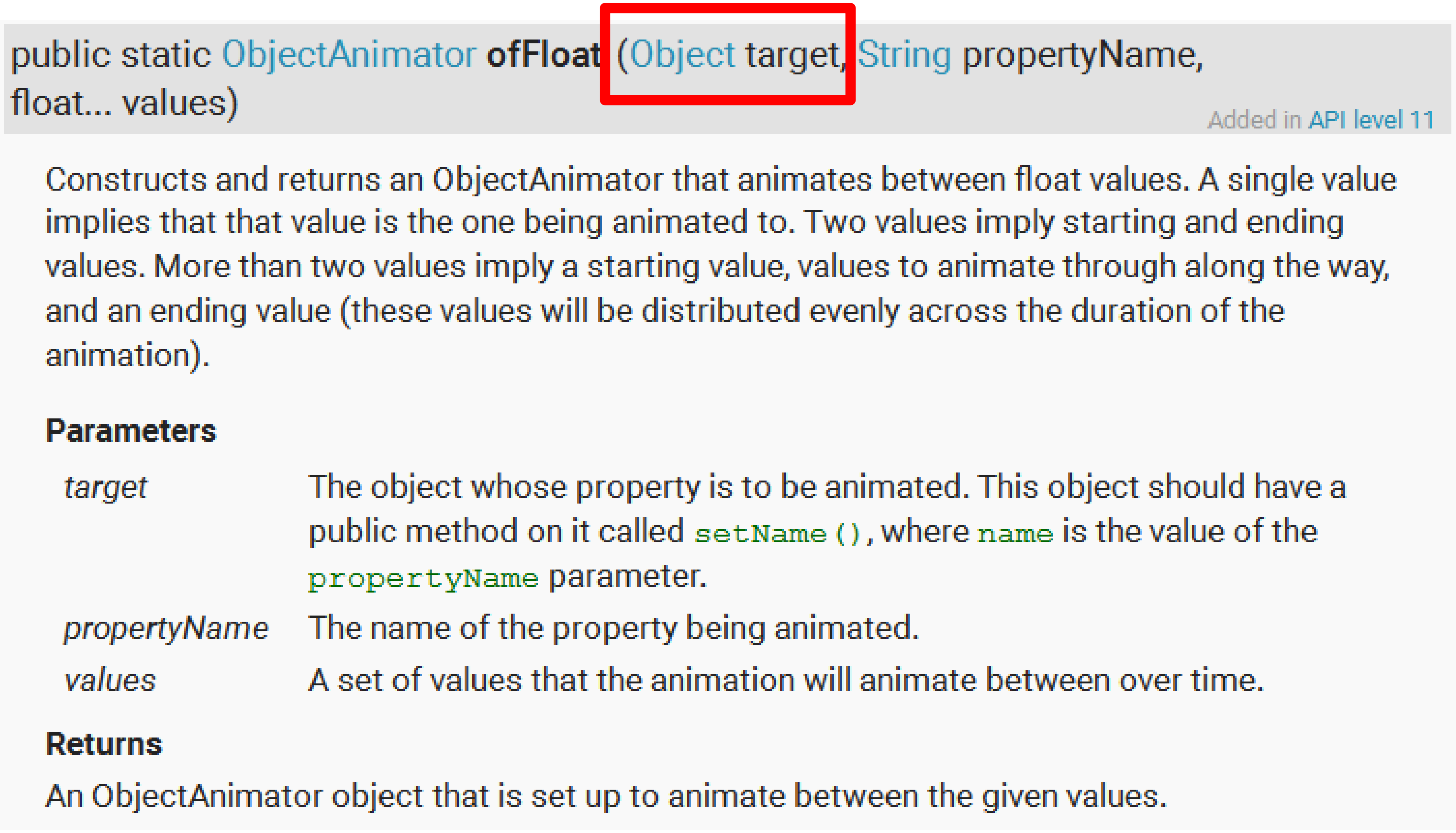
public float getY ()

The visual y position of this view, in pixels. This is equivalent to the `translationY` property plus the current `top` property.

Returns

The visual y position of this view, in pixels.

ObjectAnimator



public static ObjectAnimator **ofFloat** (Object target, String propertyName, float... values)

Added in API level 11

Constructs and returns an ObjectAnimator that animates between float values. A single value implies that that value is the one being animated to. Two values imply starting and ending values. More than two values imply a starting value, values to animate through along the way, and an ending value (these values will be distributed evenly across the duration of the animation).

Parameters

target	The object whose property is to be animated. This object should have a public method on it called `setName()`, where `name` is the value of the `propertyName` parameter.
propertyName	The name of the property being animated.
values	A set of values that the animation will animate between over time.

Returns

An ObjectAnimator object that is set up to animate between the given values.

HOW ??????

- How can the ObjectAnimator call the right methods on the object passed?

A. reflection
B. open graphics library
C. static
D. xml
E. generics

- Declared type is Object
- must be calling setY method, right?

A Sidetrack on Reflection

Reflection

- Advanced feature of Java.
- Commonly used by programs that "examine or modify the runtime behavior of applications running in the Java virtual machine"
- The Android Property Animation framework uses reflection

Why Reflection

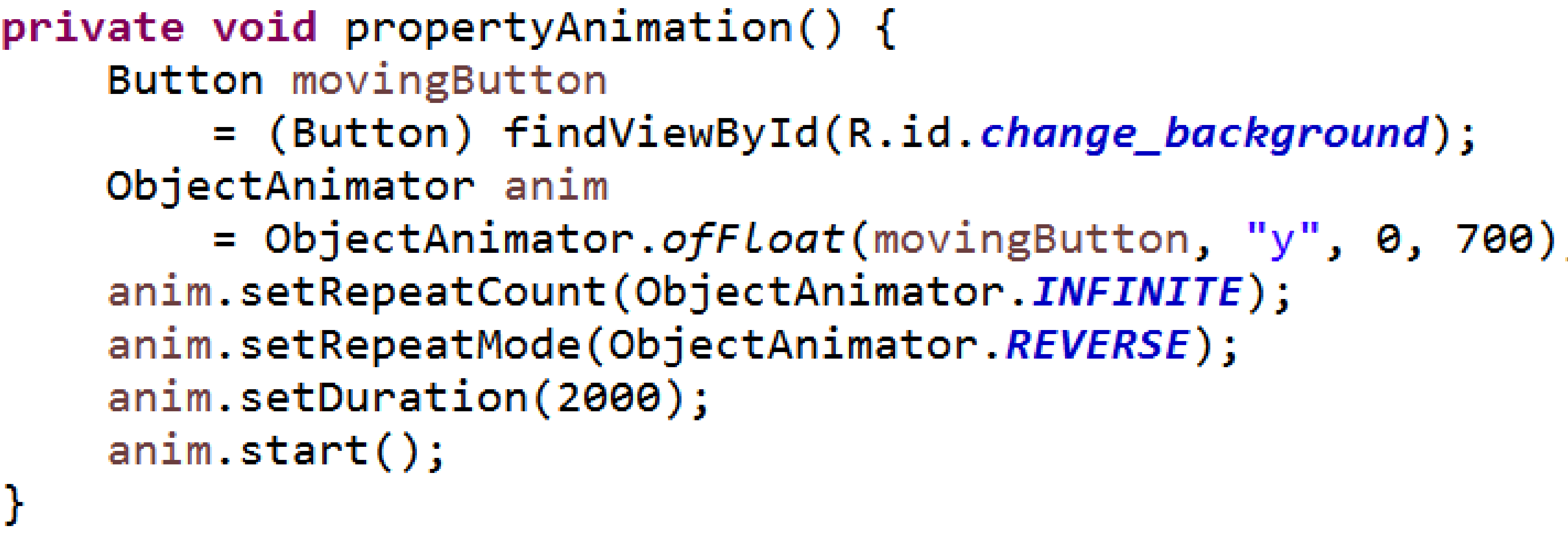
- Extensible features
  - like Android Property Animator framework
- Class Browsers and Visual Development Environments
- Debugger and Testing Tools
  - am I testing all the public methods?
  - coverage

Recall: Property Animation

- ObjectAnimator class a subclass of ValueAnimator
- Convenience class for property animation
- When animator created set animation time, ***property to animate***, and the starting and ending values
- "The constructors of this class take parameters to define the target object that will be animated as well as the name of the property that will be animated. **Appropriate set/get functions are then determined internally** and the animation will call these functions as necessary to animate the property. "

ObjectAnimator Example

- Button class must have getY and setY methods that return and accept a float

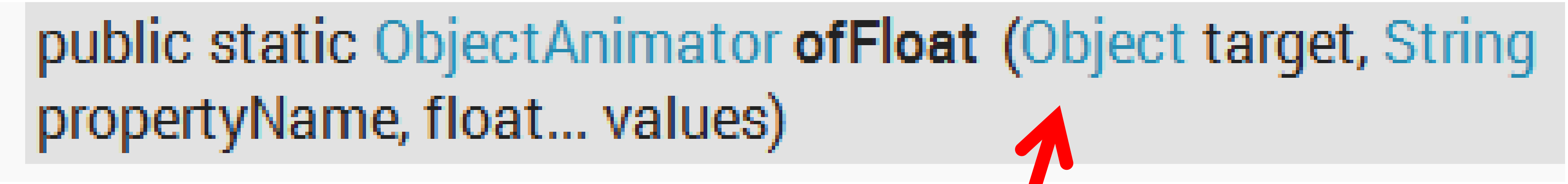


```
private void propertyAnimation() {
    Button movingButton
        = (Button) findViewById(R.id.change_background);
    ObjectAnimator anim
        = ObjectAnimator.ofFloat(movingButton, "y", 0, 700)
    anim.setRepeatCount(ObjectAnimator.INFINITE);
    anim.setRepeatMode(ObjectAnimator.REVERSE);
    anim.setDuration(2000);
    anim.start();
}
```

- ofFloat, ofInt, ofObject, ofMulti…

Object Animator

- How does the Object animator affect the y value of the Button?



```
public static ObjectAnimator ofFloat (Object target, String propertyName, float... values)
```

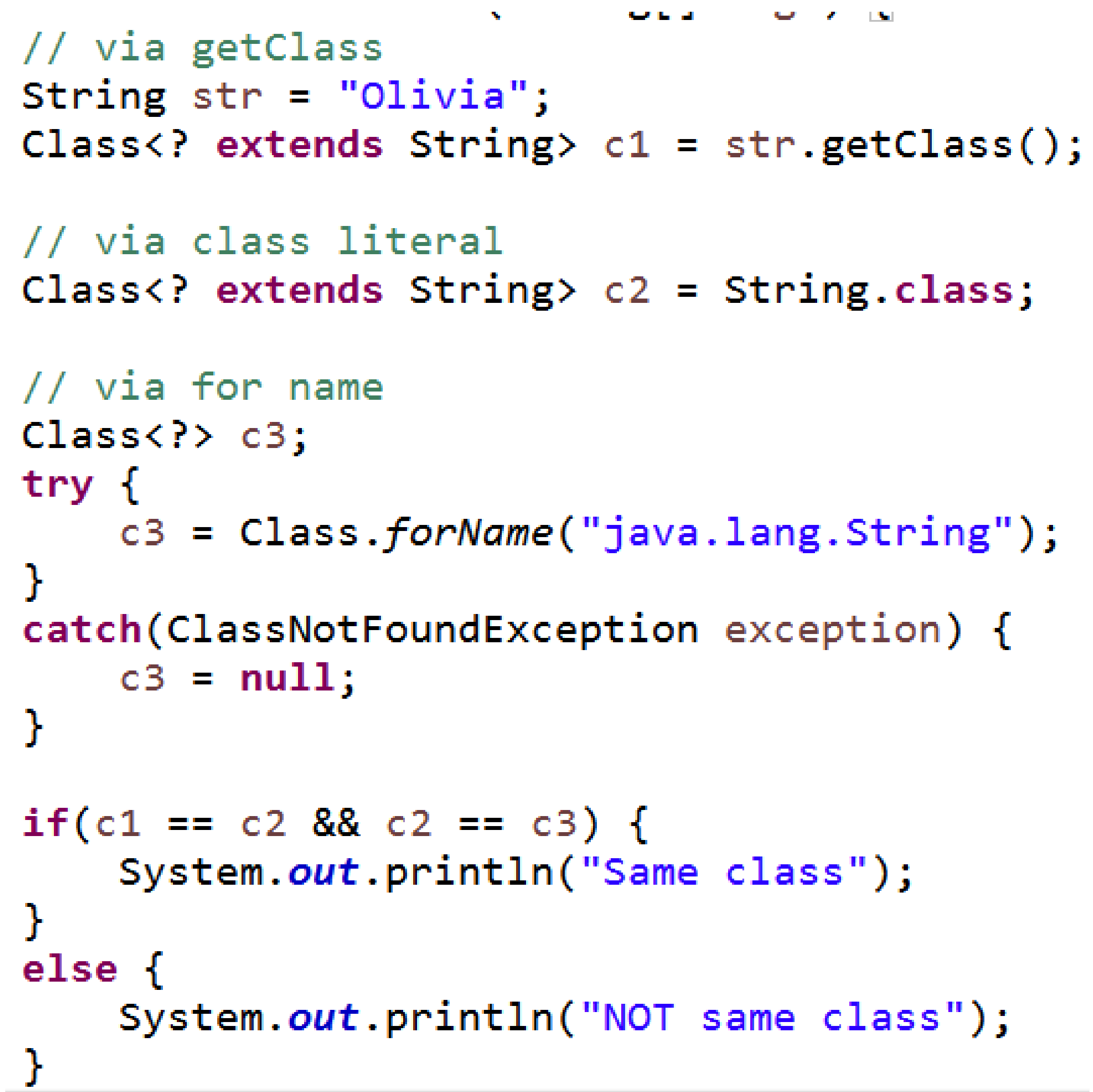
Not a button

- Recall Java, declared type, actual type
- What methods does allow compiler allow?

Class objects

- Everything in Java is a primitive (byte, short, int, long, float, double, char, boolean) or an Object
  - arrays and Enums are Objects
- The Java virtual machine instantiates an immutable instance of java.lang.Class for every type of Object necessary in a running program
- Entry point for reflection

Getting the Class object



```
// via getClass
String str = "Olivia";
Class<? extends String> c1 = str.getClass();

// via class literal
Class<? extends String> c2 = String.class;

// via for name
Class<?> c3;
try {
    c3 = Class.forName("java.lang.String");
}
catch(ClassNotFoundException exception) {
    c3 = null;
}

if(c1 == c2 && c2 == c3) {
    System.out.println("Same class");
}
else {
    System.out.println("NOT same class");
}
```

Accessing Internals

- Class object may be used to access fields, methods, and constructors

  … including **<u>private</u>** members
- methods that enumerate members (instance variables) and methods that search for a member given a name
- Like a spy

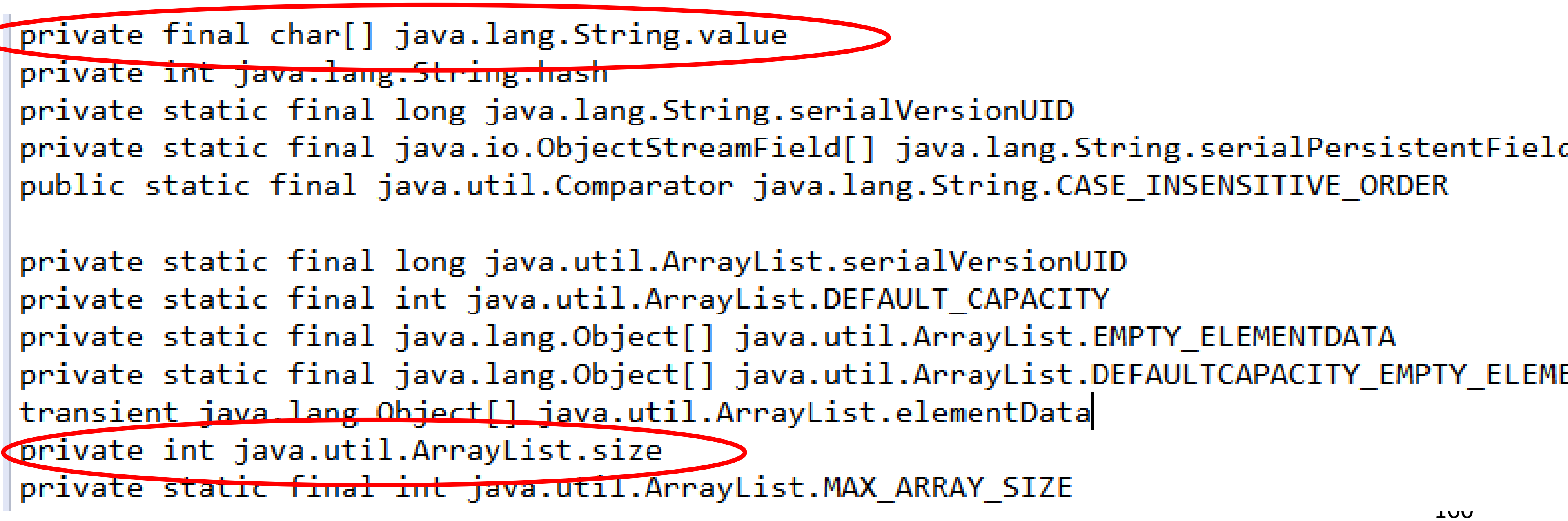
Security

- This appears to be dangerous stuff
- The ability to find out about private methods and fields

  … and even change them
- Thus many of the methods in the Class class and Reflection API throw SecurityExceptions
- If a SecurityManager is present and permission for reflection is not granted, exceptions occur
- **"If you remove this sticker, the warranty is void"**

Enumerating Fields

```java
public static void listFields(Object obj) {
    Class<?> c = obj.getClass();
    for(Field f : c.getDeclaredFields()) {
        System.out.println(f);
    }
}
```



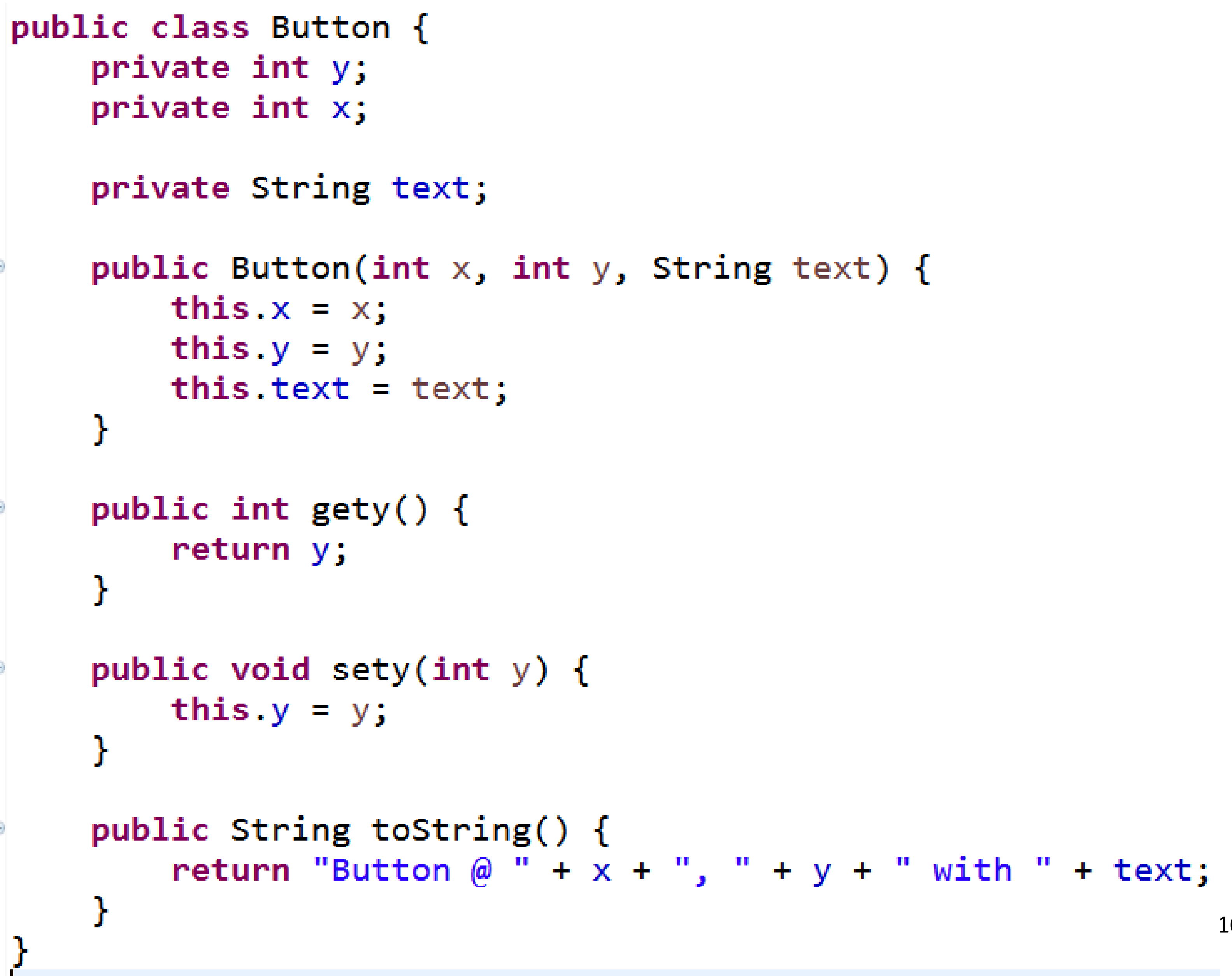
```
private final char[] java.lang.String.value
private int java.lang.String.hash
private static final long java.lang.String.serialVersionUID
private static final java.io.ObjectStreamField[] java.lang.String.serialPersistentFiel
public static final java.util.Comparator java.lang.String.CASE_INSENSITIVE_ORDER

private static final long java.util.ArrayList.serialVersionUID
private static final int java.util.ArrayList.DEFAULT_CAPACITY
private static final java.lang.Object[] java.util.ArrayList.EMPTY_ELEMENTDATA
private static final java.lang.Object[] java.util.ArrayList.DEFAULTCAPACITY_EMPTY_ELEME
transient java.lang.Object[] java.util.ArrayList.elementData
private int java.util.ArrayList.size
private static final int java.util.ArrayList.MAX_ARRAY_SIZE
```

Enumerating Methods

```java
public static void listMethods(Object obj) {
    Class<?> c = obj.getClass();
    for(Method m : c.getDeclaredMethods()) {
        System.out.println(m);
    }
}
```

```
public boolean java.lang.String.equals(java.lang.Object)
public java.lang.String java.lang.String.toString()
public int java.lang.String.hashCode()
public int java.lang.String.compareTo(java.lang.String)
public int java.lang.String.compareTo(java.lang.Object)
public int java.lang.String.indexOf(java.lang.String,int)
public int java.lang.String.indexOf(java.lang.String)
public int java.lang.String.indexOf(int,int)
public int java.lang.String.indexOf(int)
static int java.lang.String.indexOf(char[],int,int,char[],int,
static int java.lang.String.indexOf(char[],int,int,java.lang.S
```

Calling Methods

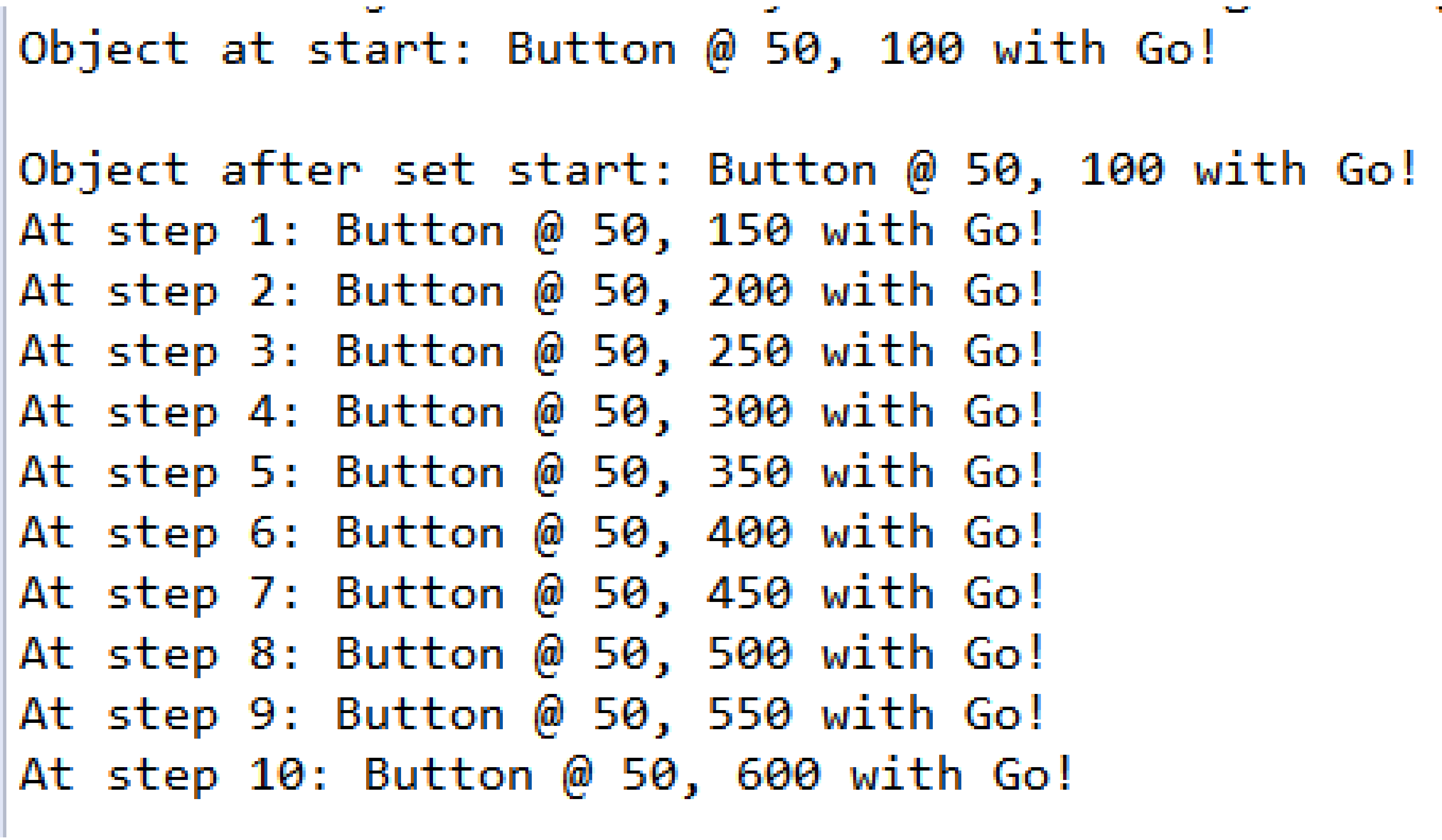


```
public class Button {
    private int y;
    private int x;

    private String text;

    public Button(int x, int y, String text) {
        this.x = x;
        this.y = y;
        this.text = text;
    }

    public int gety() {
        return y;
    }

    public void sety(int y) {
        this.y = y;
    }

    public String toString() {
        return "Button @ " + x + ", " + y + " with " + text;
    }
}
```

Calling Methods

```
System.out.println("Object at start: " + obj);
Class<?> c = obj.getClass();
Method getMethod = c.getMethod("get" + property);
int current = (Integer) getMethod.invoke(obj);

Method setMethod = c.getMethod("set" + property, int.class)
System.out.println();
System.out.println("Object after set start: " + obj);
System.out.println();
int step = (stop - current) / steps;
for(int i = 0; i < steps; i++) {
    current += step;
    setMethod.invoke(obj, current);
    System.out.println("At step " + (i + 1) + ": " + obj);
}
```

Results of Method Calls - Animate!



```
Object at start: Button @ 50, 100 with Go!

Object after set start: Button @ 50, 100 with Go!
At step 1: Button @ 50, 150 with Go!
At step 2: Button @ 50, 200 with Go!
At step 3: Button @ 50, 250 with Go!
At step 4: Button @ 50, 300 with Go!
At step 5: Button @ 50, 350 with Go!
At step 6: Button @ 50, 400 with Go!
At step 7: Button @ 50, 450 with Go!
At step 8: Button @ 50, 500 with Go!
At step 9: Button @ 50, 550 with Go!
At step 10: Button @ 50, 600 with Go!
```

Clicker Question

- Can we alter the value stored in an object via reflection if there is not set method?

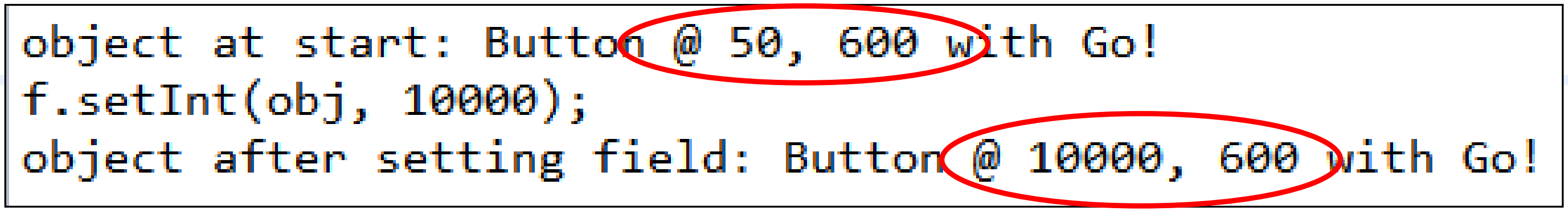
A. No

B. Yes

C. Maybe

And Then the Unthinkable Happened

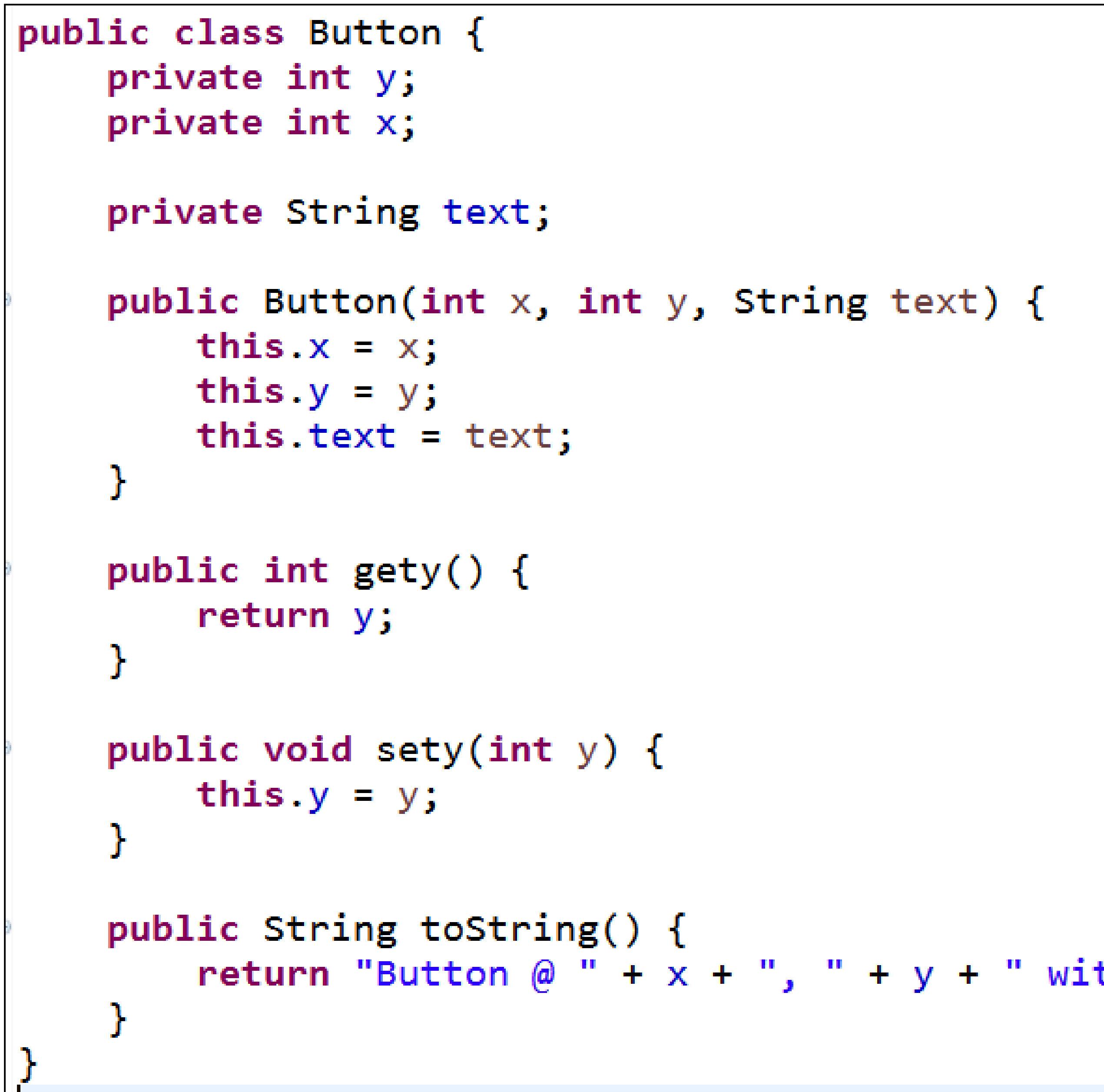
```java
public static void alterField(Object obj, String property) {
    try {
        System.out.println();
        System.out.println("object at start: " + obj);
        Class<?> c = obj.getClass();
        Field f = c.getDeclaredField(property);
        f.setAccessible(true);
        f.setInt(obj, 10000);
        System.out.println("f.setInt(obj, 10000);");
        System.out.println("object after setting field: "
                + obj);
    }
    catch(Exception e) {
        System.out.println("Sample code failed");
        e.printStackTrace();
    }
```



```
object at start: Button @ 50, 600 with Go!
f.setInt(obj, 10000);
object after setting field: Button @ 10000, 600 with Go!
```

Recall Button Class

- x is private
- no methods to access or alter x
- YIKES



```
public class Button {
    private int y;
    private int x;

    private String text;

    public Button(int x, int y, String text) {
        this.x = x;
        this.y = y;
        this.text = text;
    }

    public int gety() {
        return y;
    }

    public void sety(int y) {
        this.y = y;
    }

    public String toString() {
        return "Button @ " + x + ", " + y + " wit
    }
}
```

Clicker Question

- Can we alter the value stored in an object via reflection if there is not set method and the value is declared to be final?

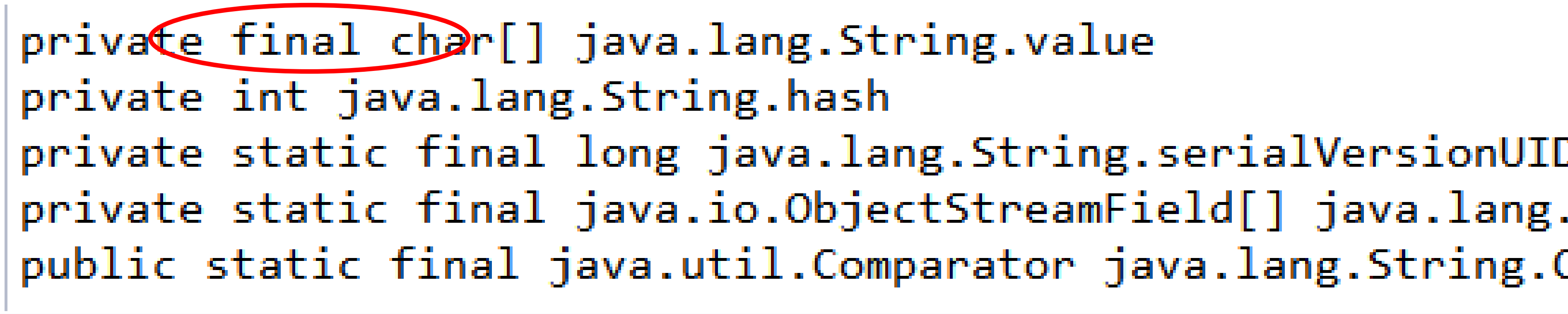
A. No

B. Yes

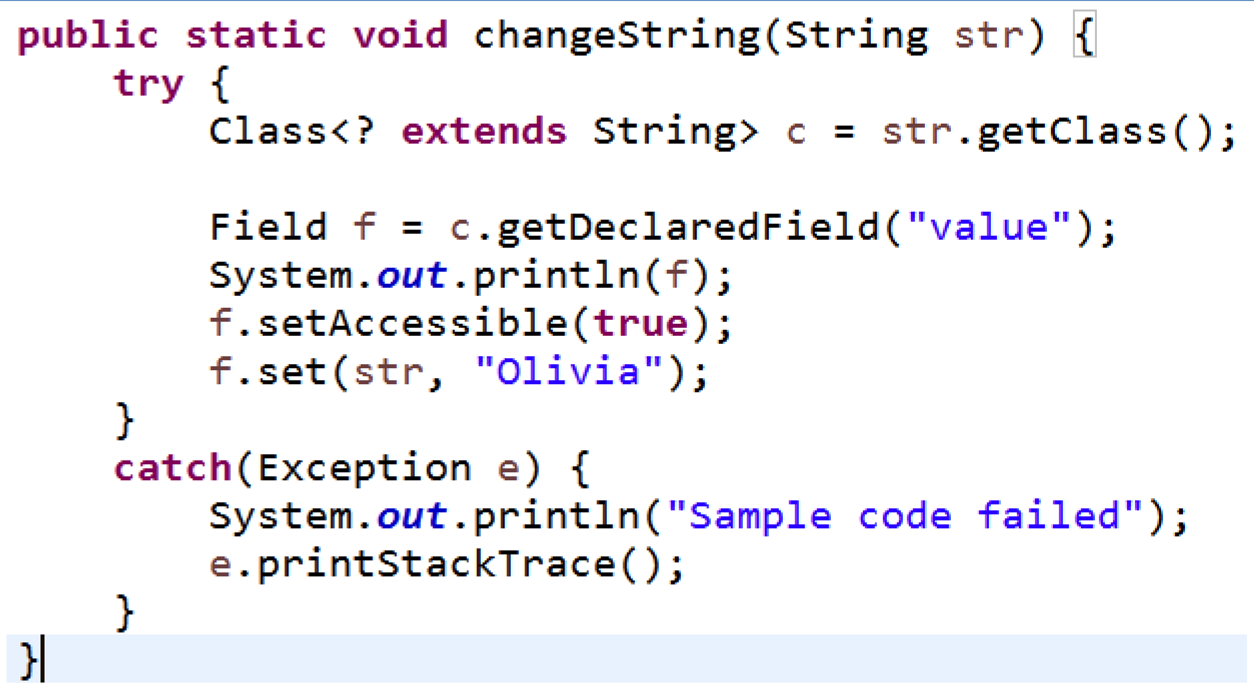
C. Maybe

But final is safe, right????

- String fields in Java 8

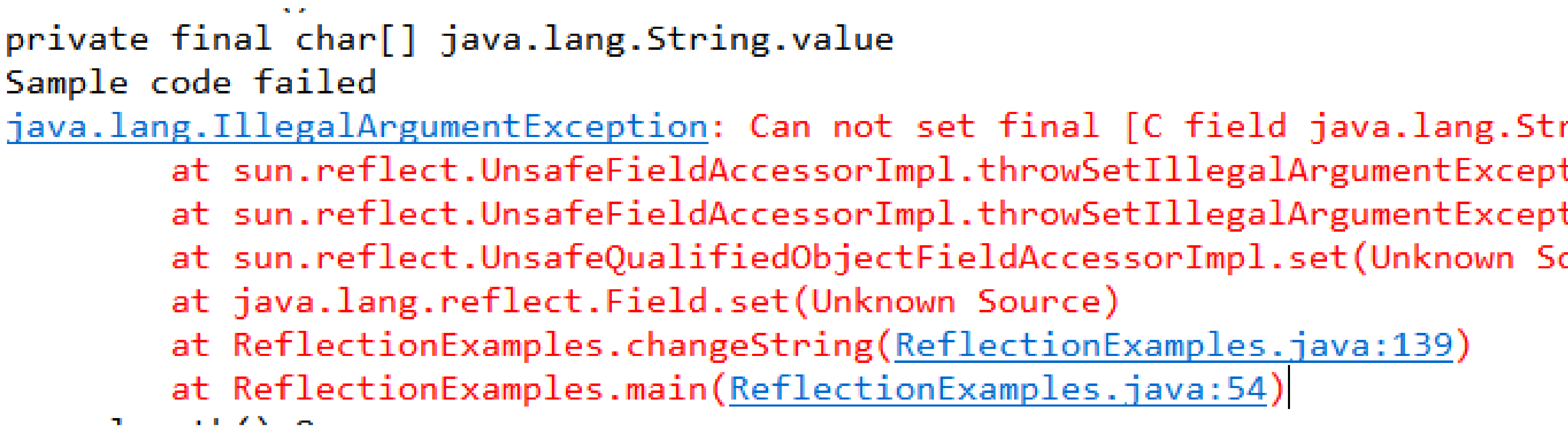


```
private final char[] java.lang.String.value
private int java.lang.String.hash
private static final long java.lang.String.serialVersionUID
private static final java.io.ObjectStreamField[] java.lang.
public static final java.util.Comparator java.lang.String.C
```



```
    public static void changeString(String str) {
        try {
            Class<? extends String> c = str.getClass();

            Field f = c.getDeclaredField("value");
            System.out.println(f);
            f.setAccessible(true);
            f.set(str, "Olivia");
        }
        catch(Exception e) {
            System.out.println("Sample code failed");
            e.printStackTrace();
        }
    }
```

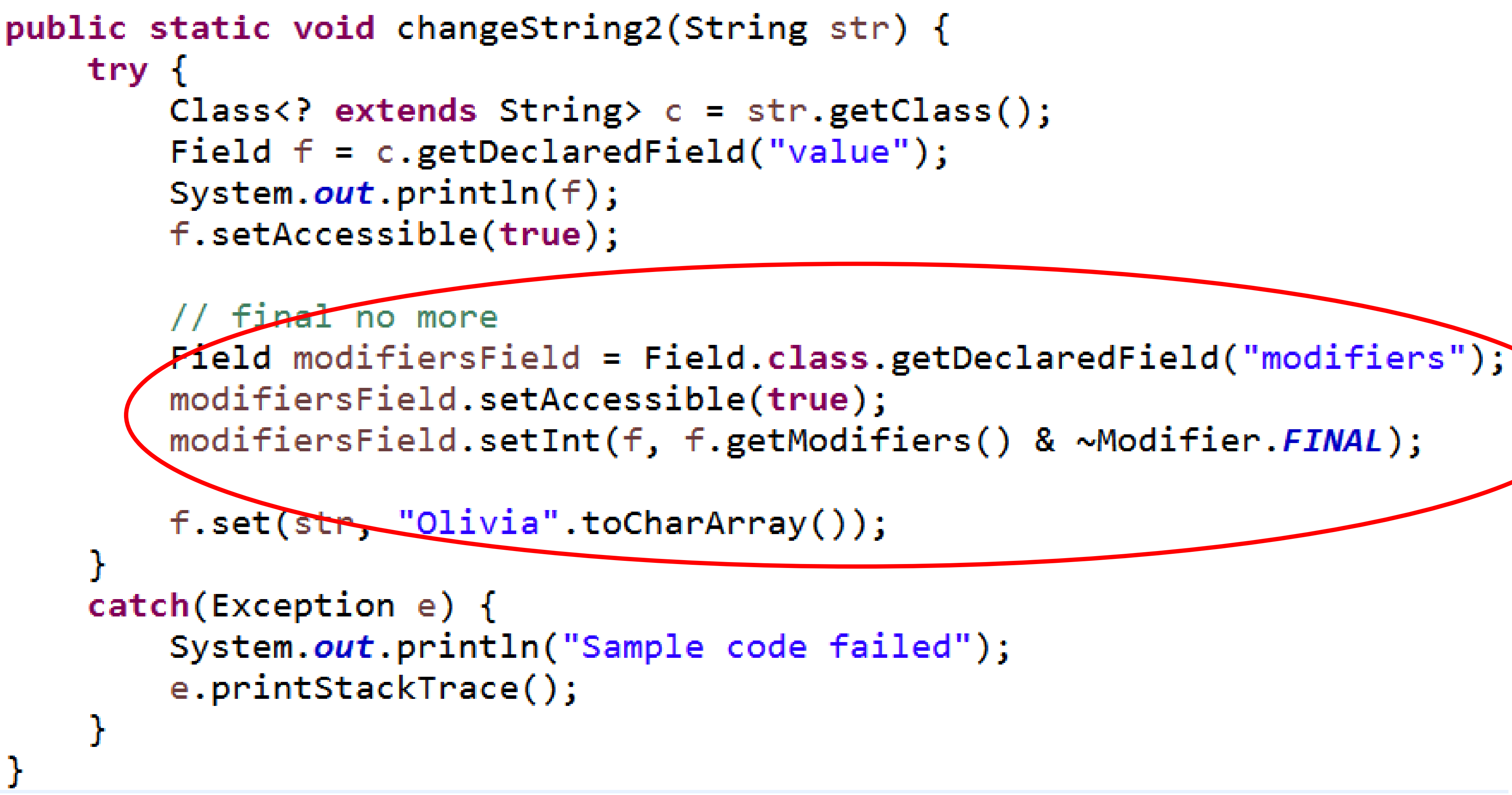
Result of changeString



```
private final char[] java.lang.String.value
Sample code failed
java.lang.IllegalArgumentException: Can not set final [C field java.lang.Str
	at sun.reflect.UnsafeFieldAccessorImpl.throwSetIllegalArgumentExcept
	at sun.reflect.UnsafeFieldAccessorImpl.throwSetIllegalArgumentExcept
	at sun.reflect.UnsafeQualifiedObjectFieldAccessorImpl.set(Unknown So
	at java.lang.reflect.Field.set(Unknown Source)
	at ReflectionExamples.changeString(ReflectionExamples.java:139)
	at ReflectionExamples.main(ReflectionExamples.java:54)
```

- We are good right?

Oh the Humanity

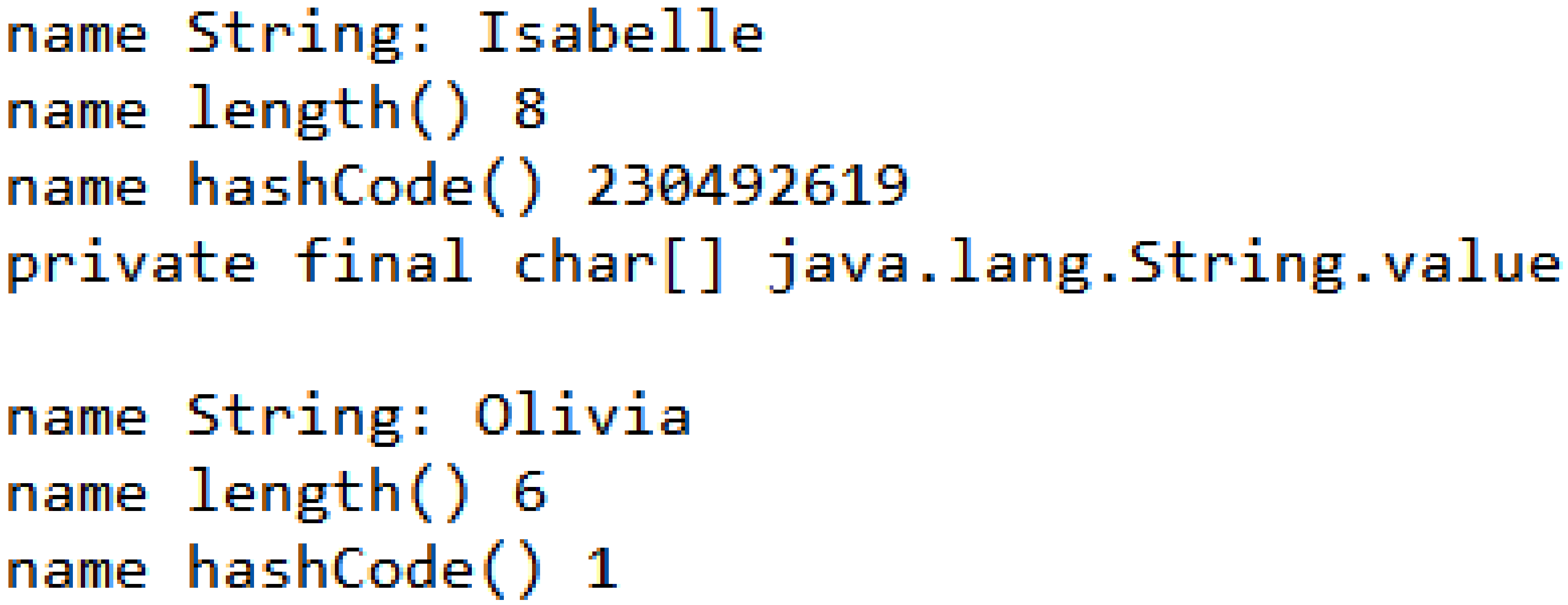


```
public static void changeString2(String str) {
    try {
        Class<? extends String> c = str.getClass();
        Field f = c.getDeclaredField("value");
        System.out.println(f);
        f.setAccessible(true);

        // final no more
        Field modifiersField = Field.class.getDeclaredField("modifiers");
        modifiersField.setAccessible(true);
        modifiersField.setInt(f, f.getModifiers() & ~Modifier.FINAL);

        f.set(str, "Olivia".toCharArray());
    }
    catch(Exception e) {
        System.out.println("Sample code failed");
        e.printStackTrace();
    }
}
```

Client Code

```java
String name = "Isabelle";
System.out.println("name String: " + name);
System.out.println("name length() " + name.length());
System.out.println("name hashCode() " + name.hashCode());
oneInts(name);
changeString2(name);
System.out.println();
System.out.println("name String: " + name);
System.out.println("name length() " + name.length());
System.out.println("name hashCode() " + name.hashCode());
```

Result



```
name String: Isabelle
name length() 8
name hashCode() 230492619
private final char[] java.lang.String.value

name String: Olivia
name length() 6
name hashCode() 1
```

RENDERING WITH COMPLEX CALCULATIONS

More Complex Graphics

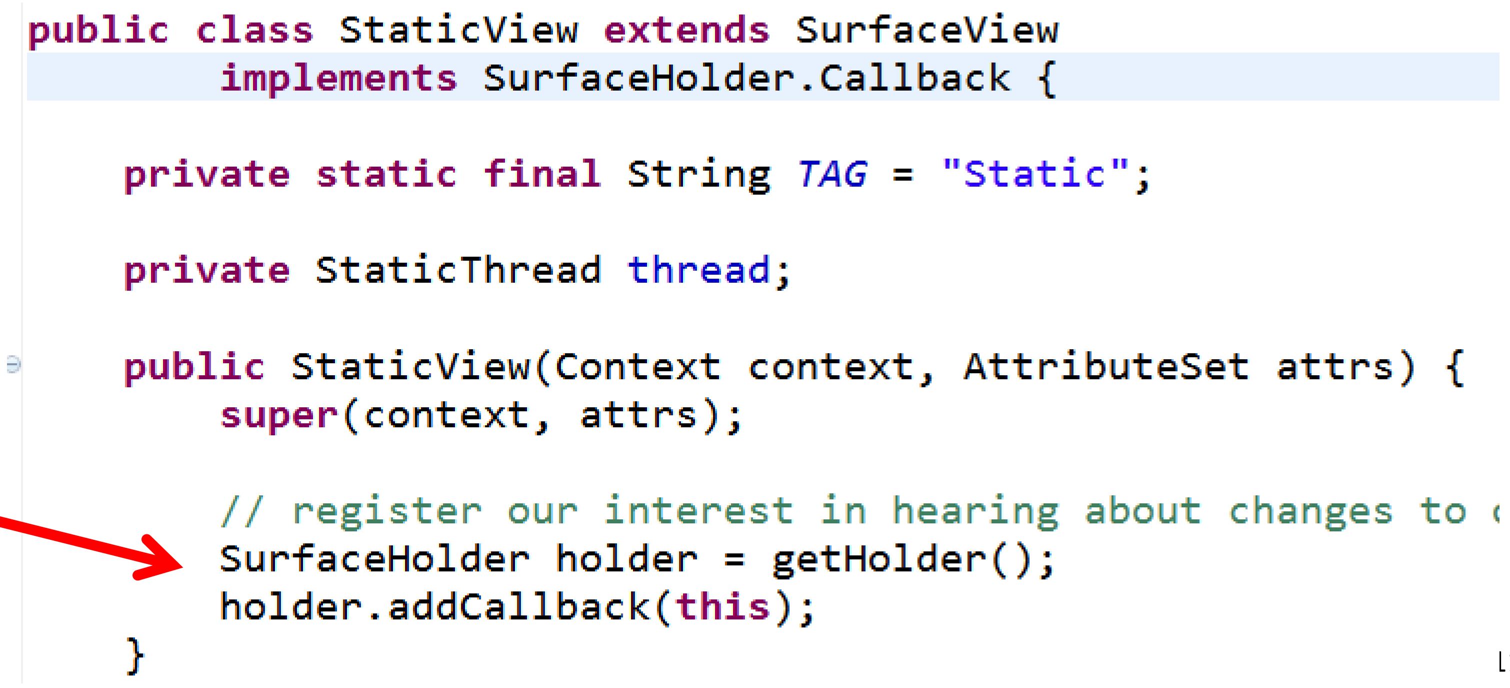
- Don't want apps to become unresponsive
- If complex graphics or animation use SurfaceView class
- Main view not waiting on onDraw to finish
- secondary thread with reference to SurfaceView
- SrufaceView draws and when done display result

Using a SurfaceView

- extend SurfaceView
- implement SurfaceHolder.Callback
  - methods to notify main View when SurfaceView is created, changed or destroyed

Simple Example

- Static Screen
- continuously draw several hundred small rectangles (points, with stroke = 10)
  - slowly fill screen and then keep changing



```
public class StaticView extends SurfaceView
        implements SurfaceHolder.Callback {

    private static final String TAG = "Static";

    private StaticThread thread;

    public StaticView(Context context, AttributeSet attrs) {
        super(context, attrs);

        // register our interest in hearing about changes to
        SurfaceHolder holder = getHolder();
        holder.addCallback(this);
    }
```

Implement SurfaceHolder.Callback methods

```
// called when surface changes size
@Override
public void surfaceChanged(SurfaceHolder holder, int format,
        int width, int height) {

}

// called when surface is first created
@Override
public void surfaceCreated(SurfaceHolder holder)
{
    thread = new StaticThread(holder);
    thread.setRunning(true);
    thread.start(); //start the animation
}
```

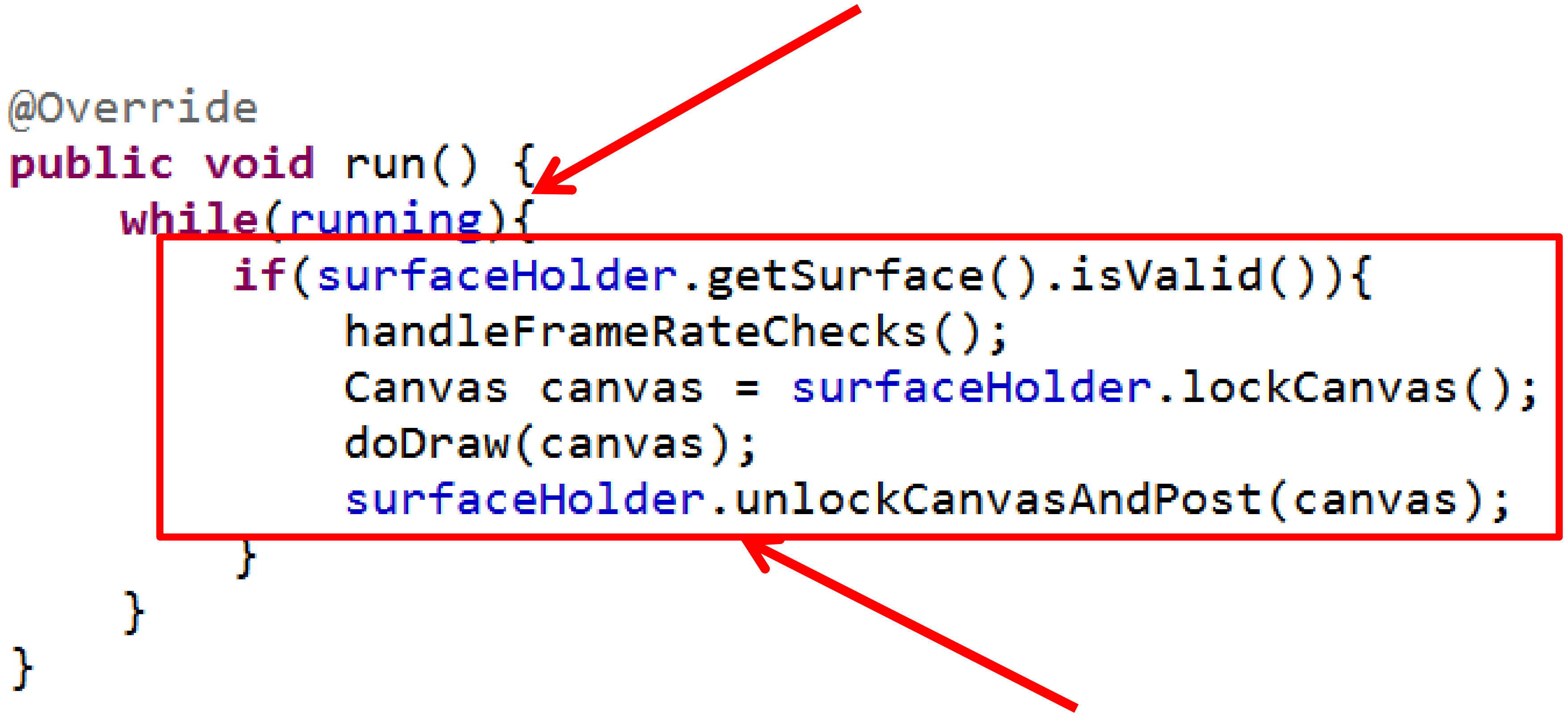
Prevent Runaway Threads!

```
// called when the surface is destroyed
@Override
public void surfaceDestroyed(SurfaceHolder holder) {
    // ensure that thread terminates properly
    boolean retry = true;
    thread.setRunning(false);

    while (retry) {
        try {
            thread.join();
            Log.d(TAG, "Thread stopped! " + thread);
            retry = false;
        }
        catch (InterruptedException e) {}
    }
}
```

Inner Class for Thread

```
private class StaticThread extends Thread {

    private boolean running;
    private SurfaceHolder surfaceHolder;
    private Bitmap image;
    private Random random;
    private Paint paint;

    public StaticThread(SurfaceHolder sh) {
        surfaceHolder = sh;
        random = new Random();
        paint = new Paint();
        paint.setStyle(Paint.Style.STROKE);
        paint.setStrokeWidth(10);
        image = Bitmap.createBitmap(getWidth(), getHeight(),
                Bitmap.Config.ARGB_8888);
        Log.d(TAG, "width: " + image.getWidth());
        Log.d(TAG, "height: " + image.getHeight());
        Log.d(TAG, "image: " + image);
    }

    public void setRunning(boolean status) {
        running = status;
    }
```

Run Method in StaticThread

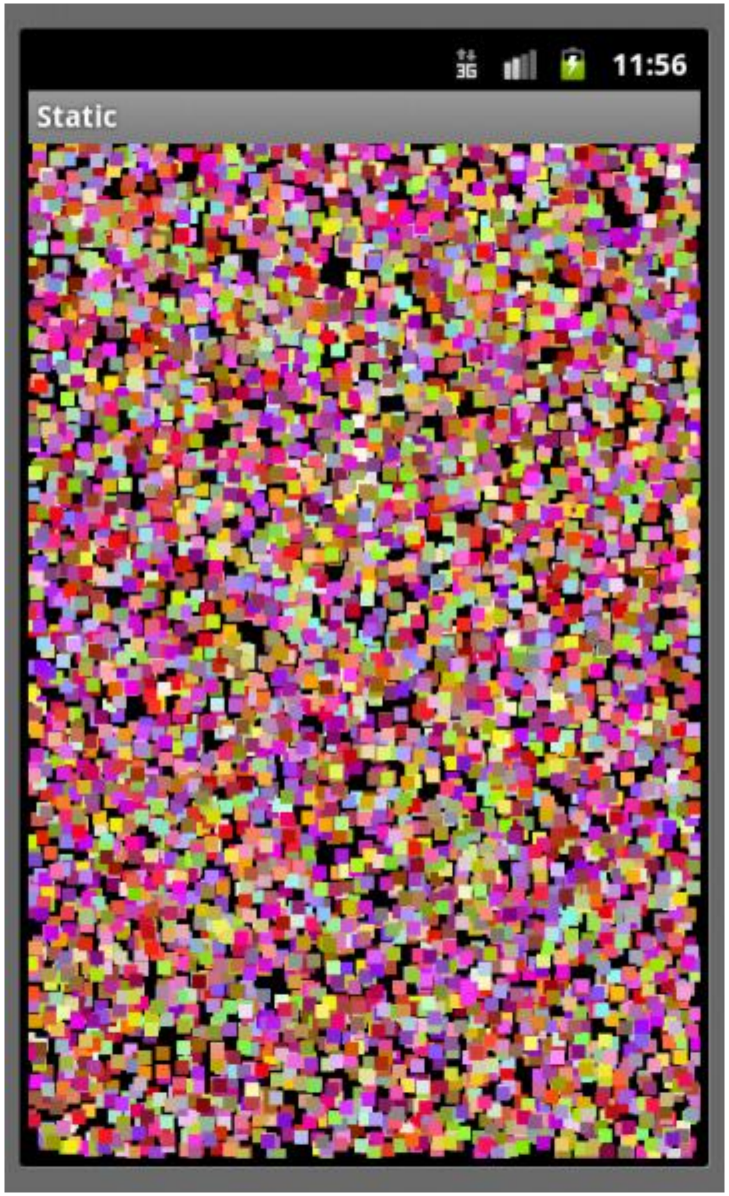


```
@Override
public void run() {
    while(running){
        if(surfaceHolder.getSurface().isValid()){
            handleFrameRateChecks();
            Canvas canvas = surfaceHolder.lockCanvas();
            doDraw(canvas);
            surfaceHolder.unlockCanvasAndPost(canvas);
        }
    }
}
```

Standard Approach for Drawing on SurfaceView

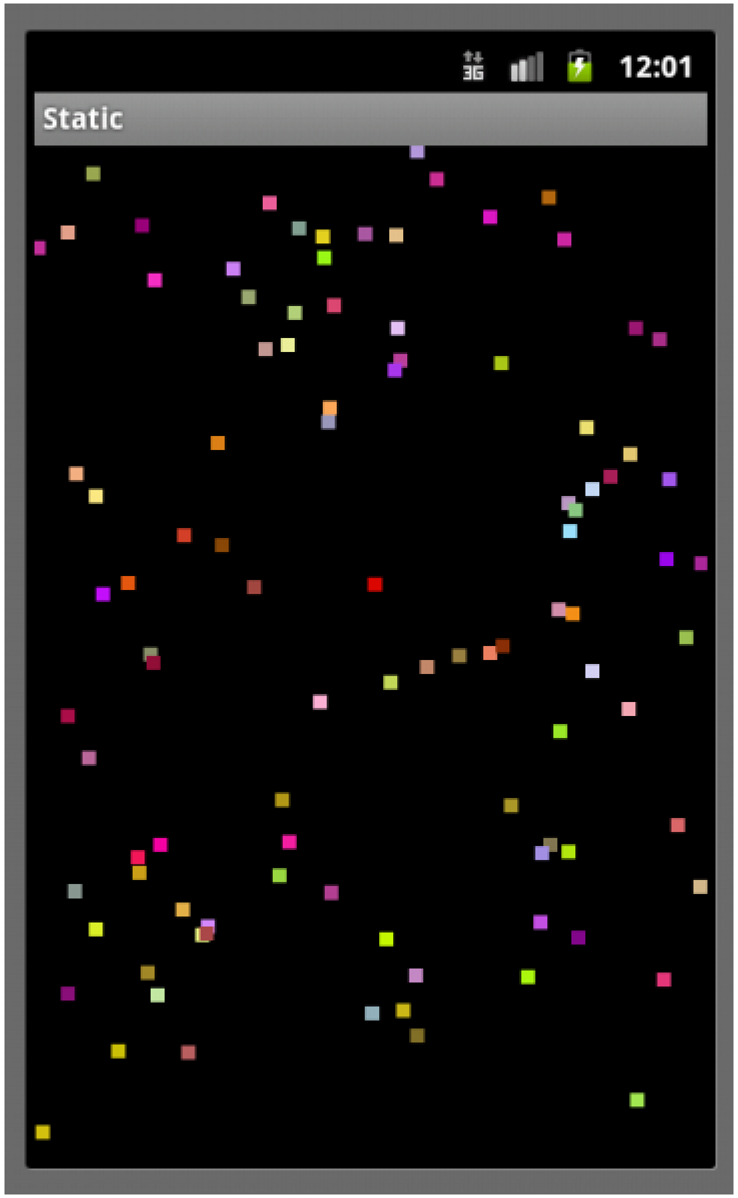
Demo run()

- Pronounced flicker and jitter
- Double buffer under the hood
- We are drawing on two different Bitmaps
- Canvas does drawing onto Bitmap



Remove Flicker / Jitter

- If we draw background each "frame" then we don't redraw previous rectangles
- How about "saving" all the data?
  - points, colors

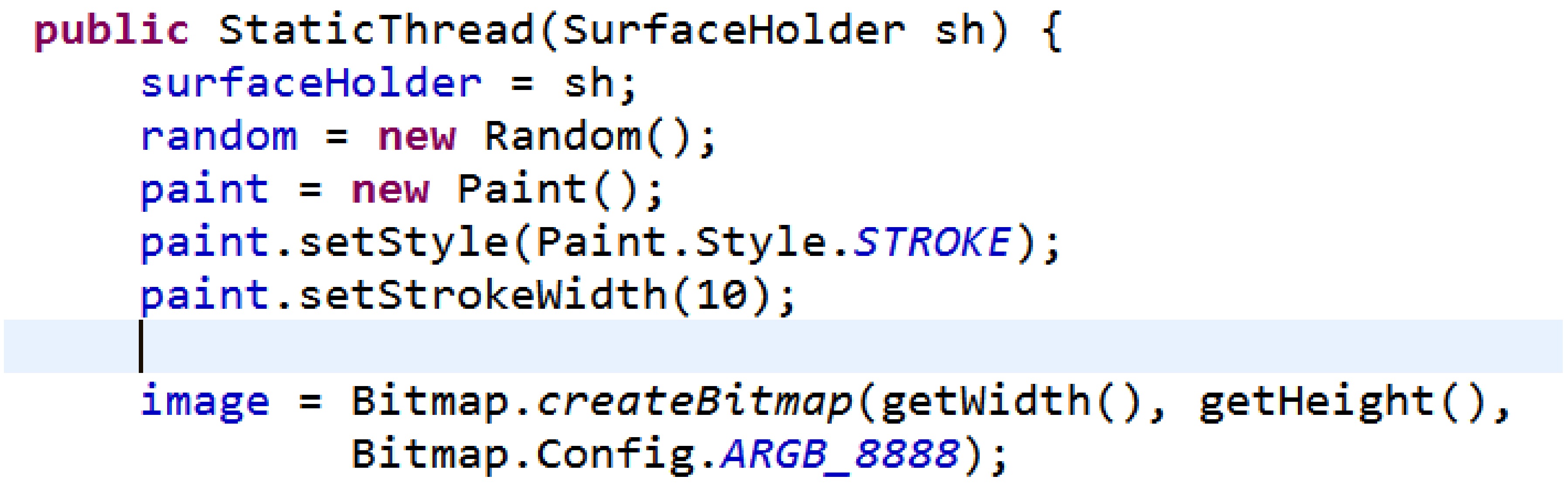


Alternative

- Recall two approaches:
  - draw on UI thread by overriding onDraw
    - create custom View (tutorial 4)
    - okay if not a lot of drawing
  - must keep UI thread responsive
    - complex drawing or animations using SurfaceView
- Third approach, possible variation on the above two approaches
  - maintain a separate Bitmap

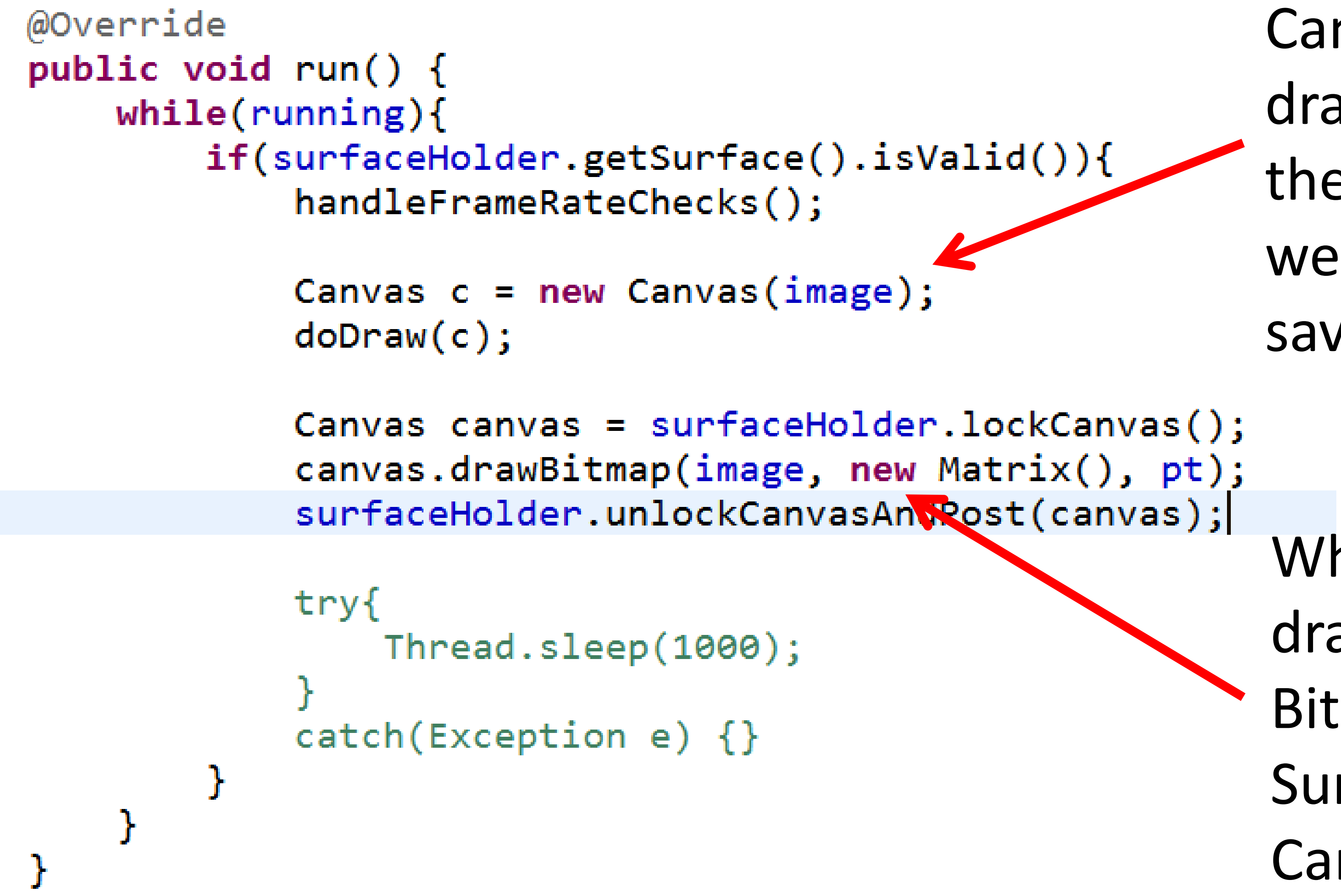
Separate Bitmap

- StaticThread has a Bitmap instance var
- Initialize in constructor



```java
public StaticThread(SurfaceHolder sh) {
    surfaceHolder = sh;
    random = new Random();
    paint = new Paint();
    paint.setStyle(Paint.Style.STROKE);
    paint.setStrokeWidth(10);

    image = Bitmap.createBitmap(getWidth(), getHeight(),
            Bitmap.Config.ARGB_8888);
```

Updates to Bitmap



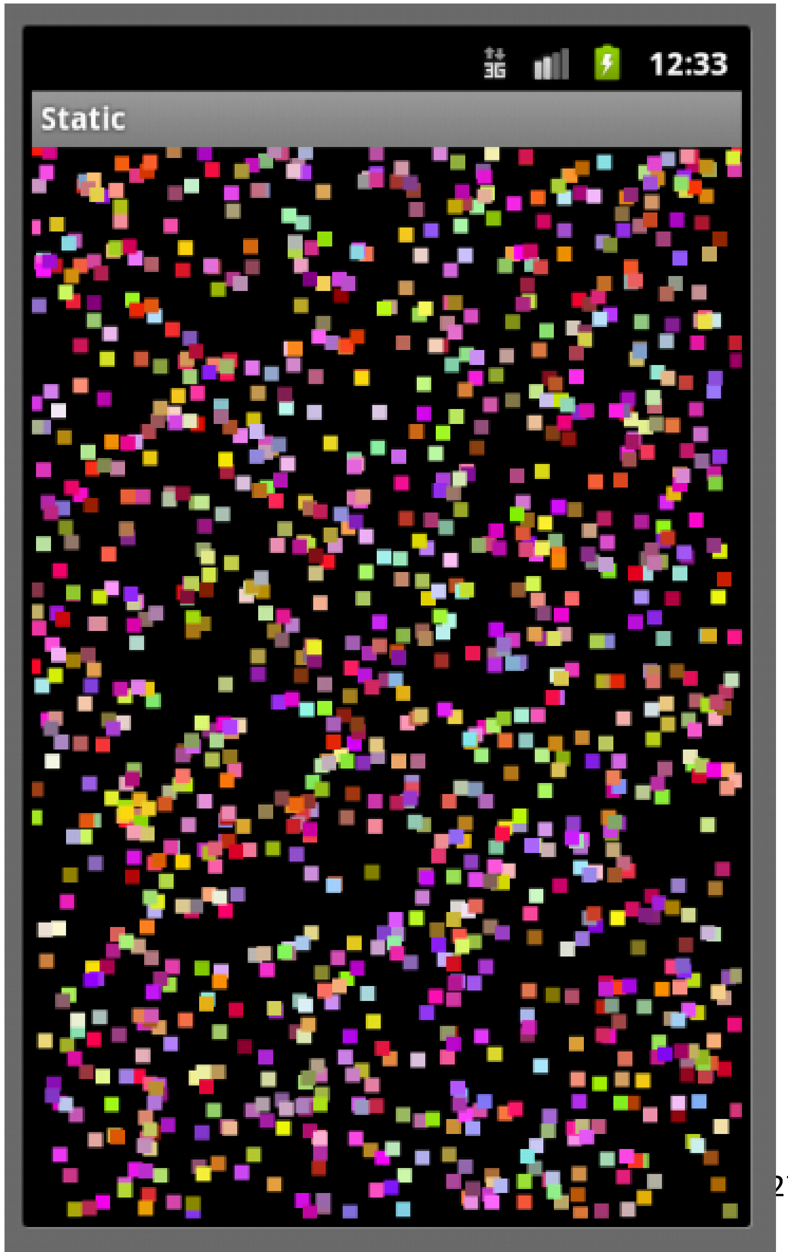
```
@Override
public void run() {
    while(running){
        if(surfaceHolder.getSurface().isValid()){
            handleFrameRateChecks();

            Canvas c = new Canvas(image);
            doDraw(c);

            Canvas canvas = surfaceHolder.lockCanvas();
            canvas.drawBitmap(image, new Matrix(), pt);
            surfaceHolder.unlockCanvasAndPost(canvas);

            try{
                Thread.sleep(1000);
            }
            catch(Exception e) {}
        }
    }
}
```

Create a Canvas to draw on the Bitmap we are saving

When done drawing to Bitmap use SurfaceView Canvas to draw

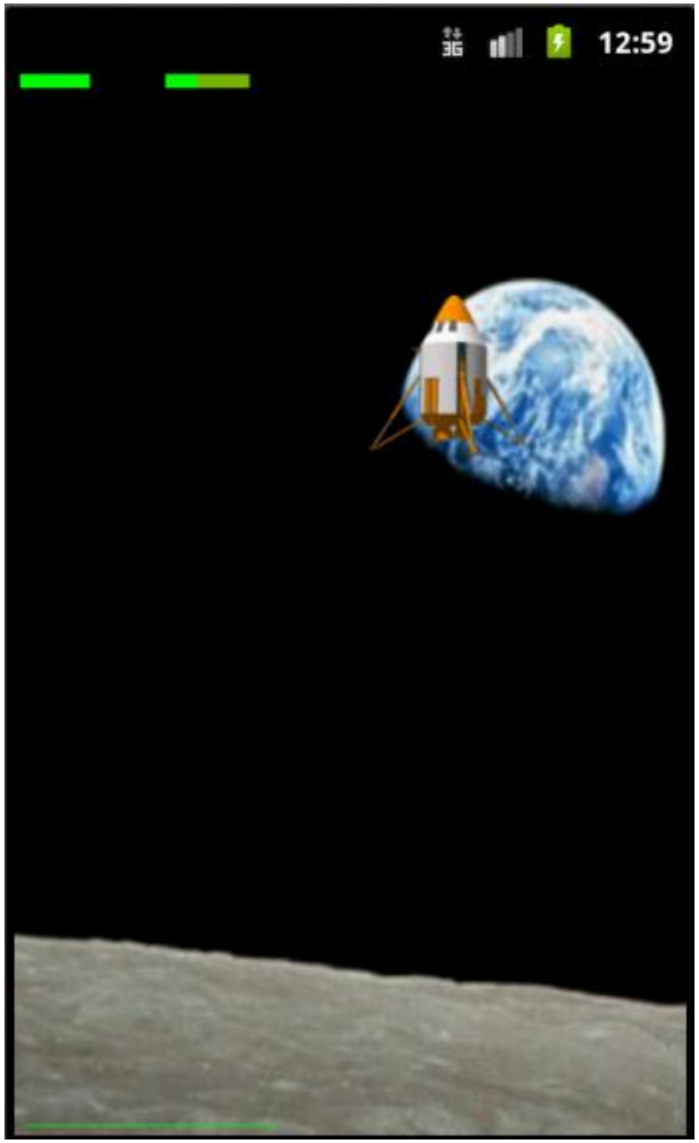
Demo Alt Version of run()

- Flicker and jitter?
- Also possible to save Bitmap to file for later use



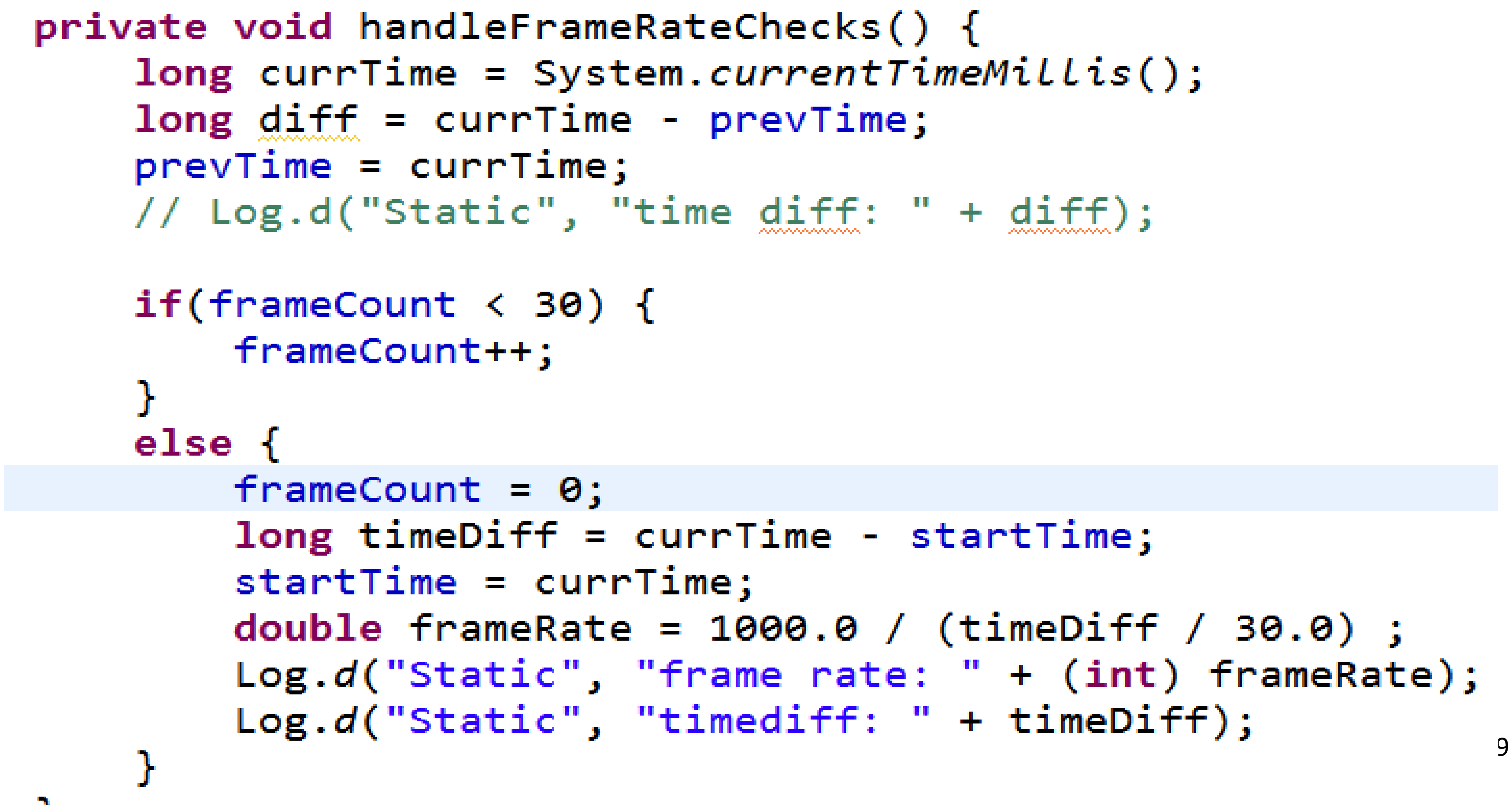
Animations

- Frame based vs. Time based
- Frame based:
  - update every frame
  - simple, but difference in frame rates
- Time based
  - update every frame but based on time elapsed since last frame
  - more work, more accurate
  - sdk example lunar lander



Checking Frame Rate

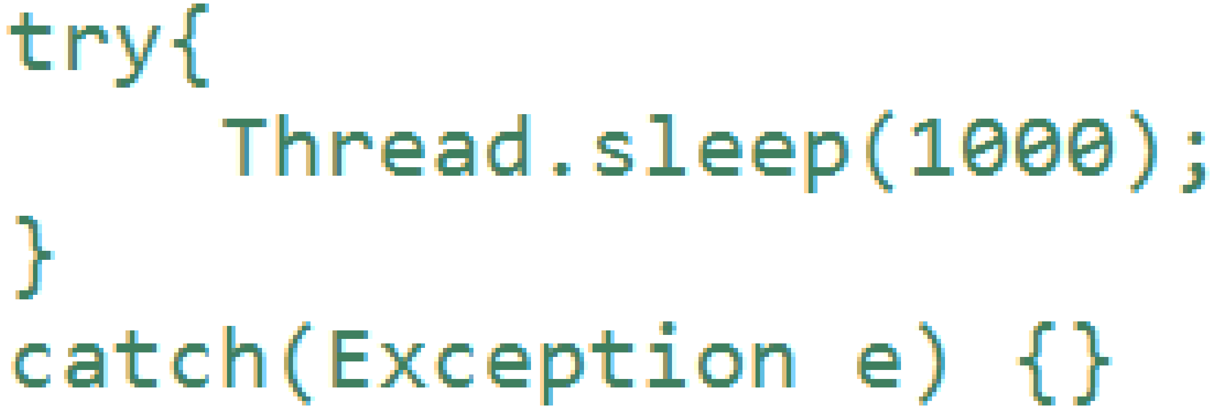
- From StaticView
- Emulator 6-7 fps, dev phone 40 -45 fps



```java
private void handleFrameRateChecks() {
    long currTime = System.currentTimeMillis();
    long diff = currTime - prevTime;
    prevTime = currTime;
    // Log.d("Static", "time diff: " + diff);

    if(frameCount < 30) {
        frameCount++;
    }
    else {
        frameCount = 0;
        long timeDiff = currTime - startTime;
        startTime = currTime;
        double frameRate = 1000.0 / (timeDiff / 30.0) ;
        Log.d("Static", "frame rate: " + (int) frameRate);
        Log.d("Static", "timediff: " + timeDiff);
    }
```

Controlling Frame Rate

- Sleep after completing work in loop of run
- More complex than shown, use previous time and current time



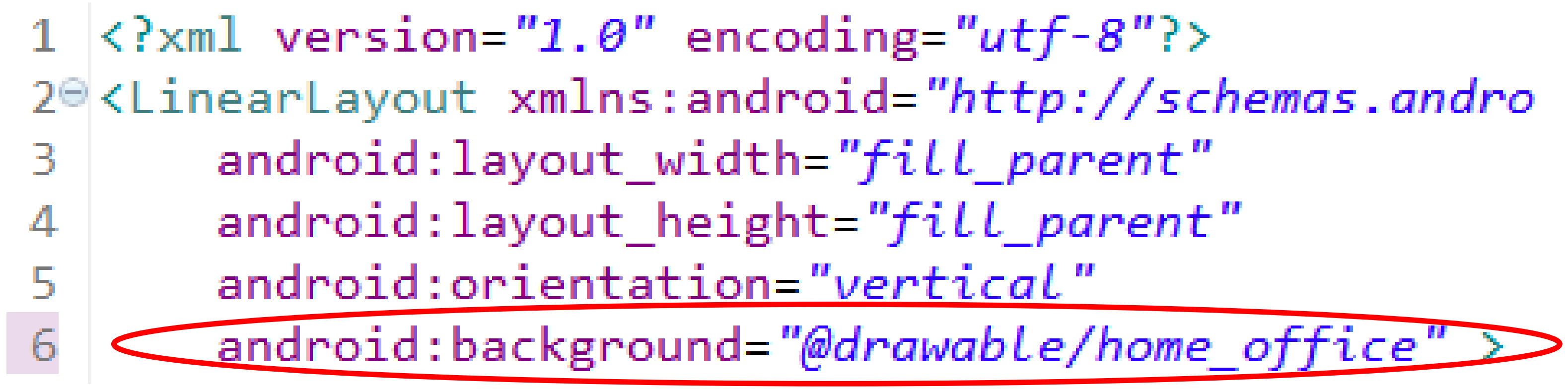
```
try{
        Thread.sleep(1000);
}
catch(Exception e) {}
```

Two Methods for Displaying 2D Graphics

- Two approaches
- draw graphics or animations into a View object that is part of layout
  - define graphics that go into View
  - simple approach for non dynamic graphics or simple "tweened" animations
  - This is what we did in Tic Tac Toe
- Draw graphics directly to a Canvas
  - the complex way, but with more control

ADDING DRAWABLES TO VIEWS

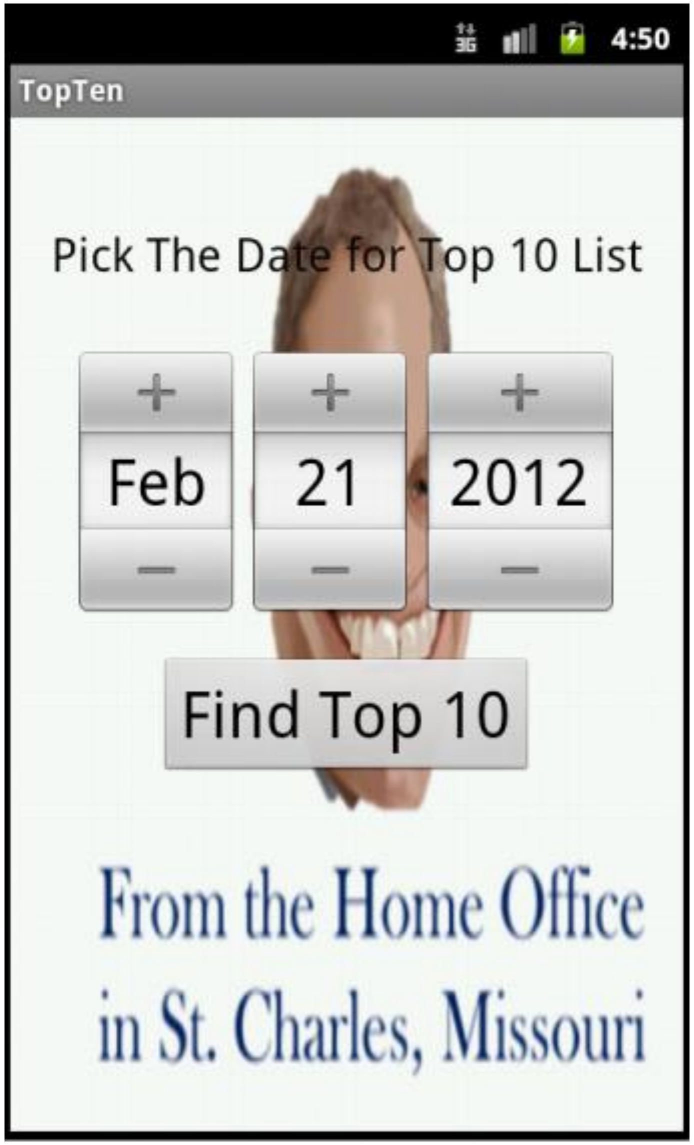
Adding Drawables to Views



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.andro
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:background="@drawable/home_office" >
```

- Change background to an image
  - previously used background colors

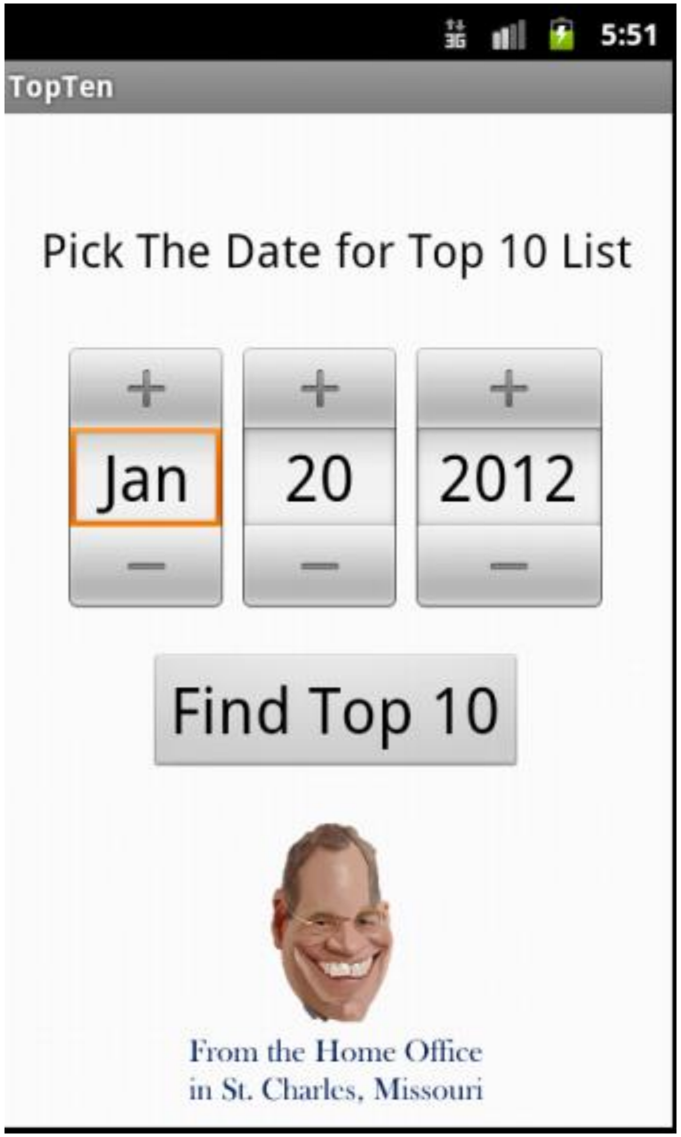
Top Ten With Image Background



Add ImageView to Layout

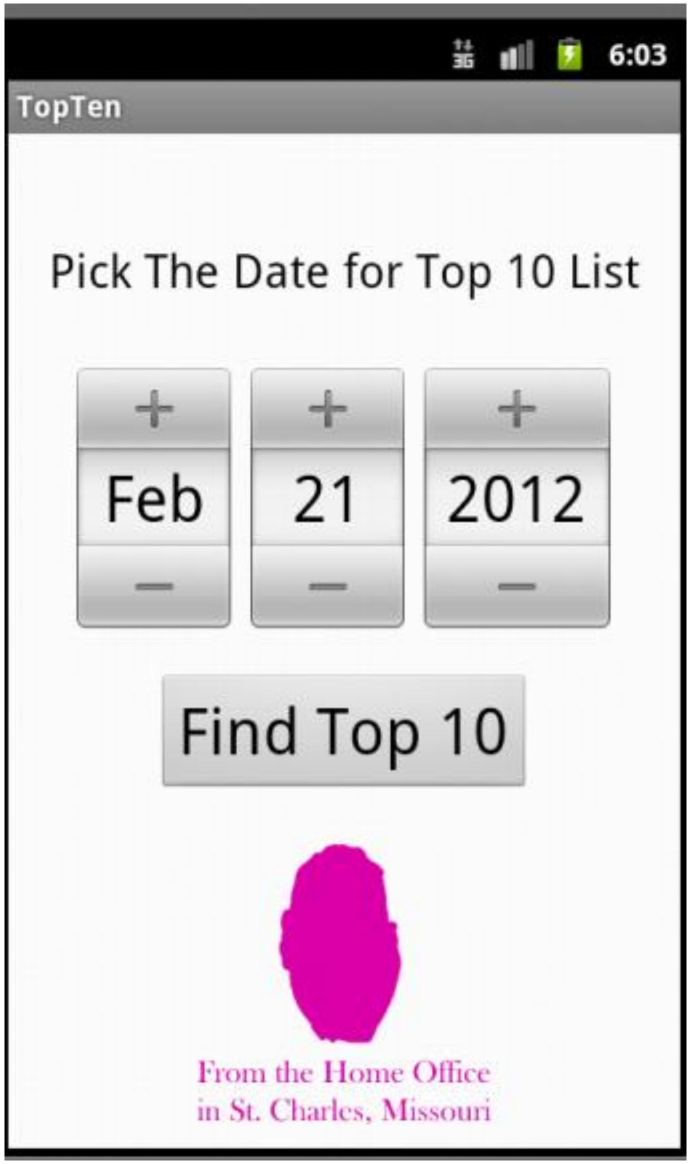
- In the main.xml for top ten

```
<ImageView
    android:id="@+id/imageView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/home_office" />
```



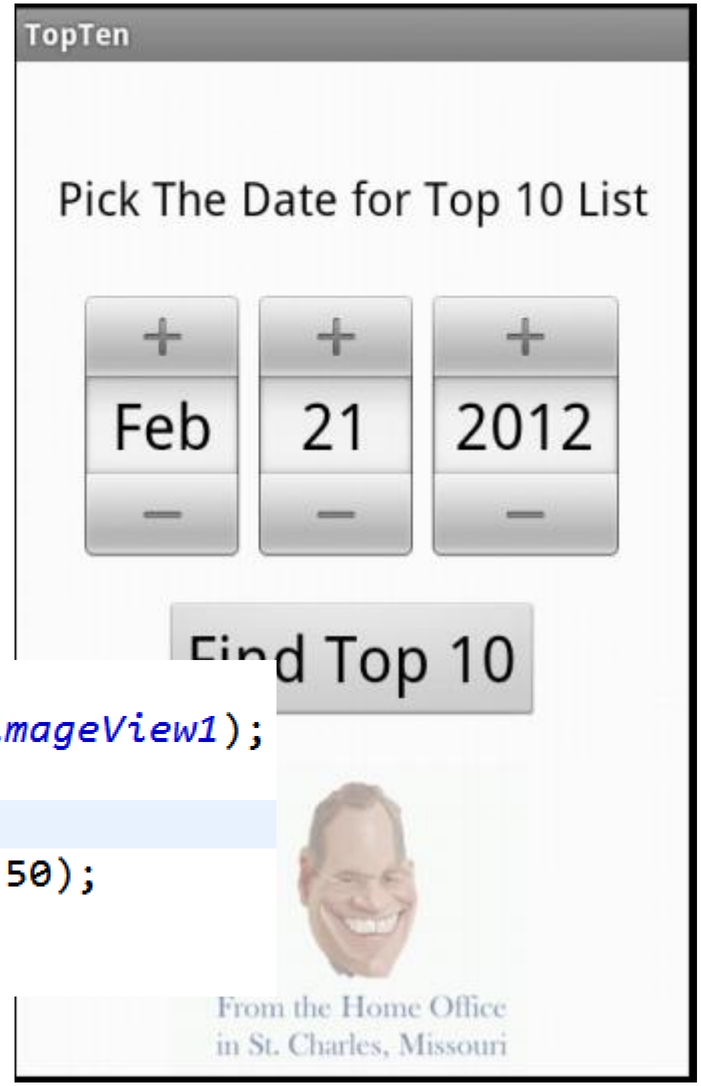
ImageView Attributes

- scaleType: how image should be moved or resized in the ImageView
- tint: affects color of image
- more to position image in ImageView



Changing ImageView Programmatically

- Randomly set the alpha (transparency of the image)
- Or pick an image randomly
- Or set image based on month (Season)


TopTen

Pick The Date for Top 10 List

```
private void changeImage() {
    ImageView iv = (ImageView) findViewById(R.id.imageView1);
    Drawable image = getResources()
            .getDrawable(R.drawable.home_office2);
    image.setAlpha( (int) (Math.random() * 100) + 50);
    iv.setImageDrawable(image);
}
```