Filthy Rich Android Clients

Romain Guy
UI Toolkit Engineer, Android
Google

Overall Presentation Goal

Learn how to apply Filthy Rich Clients techniques to the Android platform.

Speaker’s qualifications

- Romain works on the Android UI toolkit at Google
- Romain co-authored the book Filthy Rich Clients
- Romain enjoy writing Filthy Rich Client applications
- Romain knows how to use Keynote

Filthy Rich Clients are not specific to any particular platform or software stack.

The are a set of techniques applicable across many platforms and toolkits.

Android is a modern mobile operating system offering advanced features for graphical effects.

Let’s discover some of these features.

Agenda

- Architecture
- Graphics
- Animation
- Performance

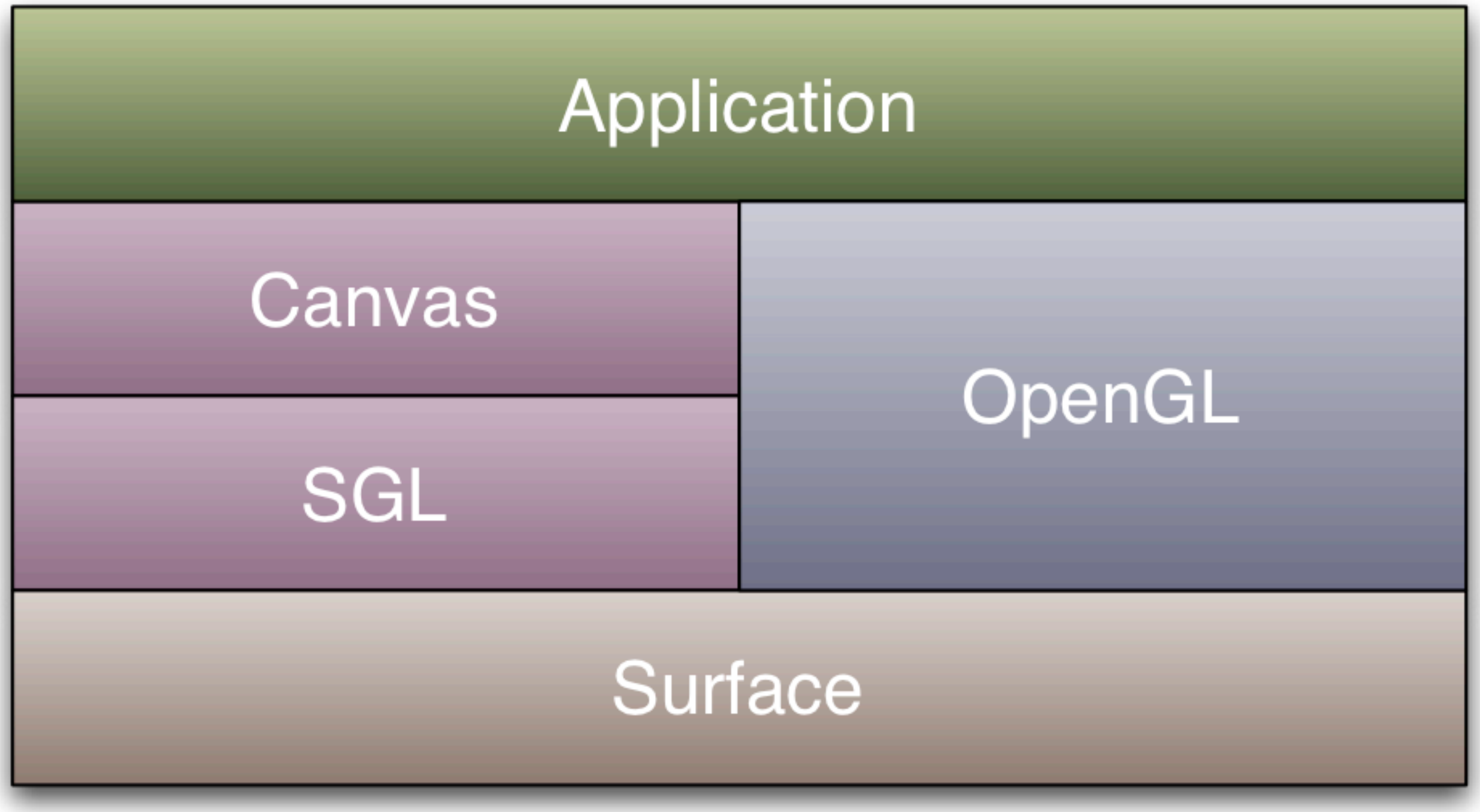
Agenda

- Architecture
- Graphics
- Animation
- Performance

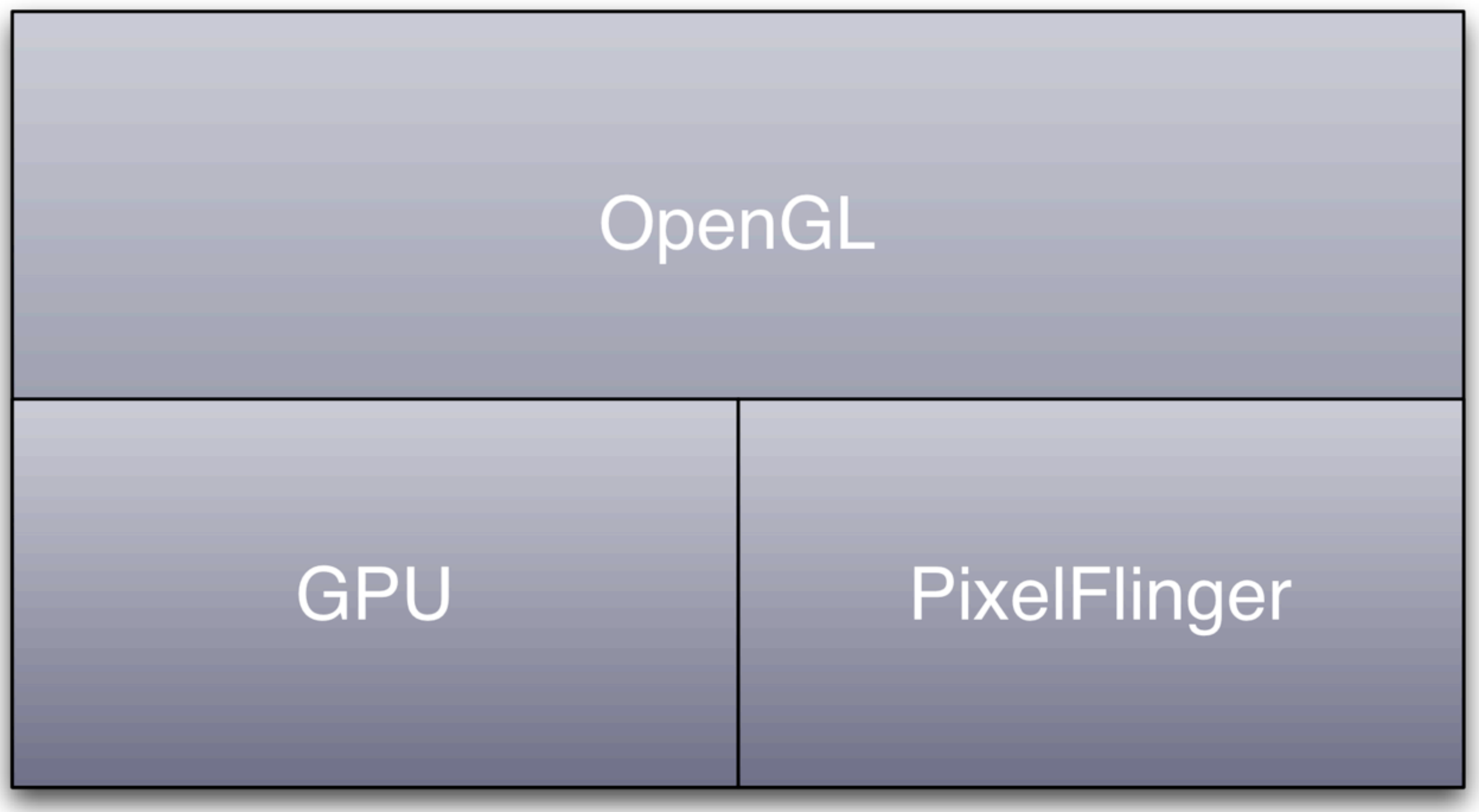
Glossary

- **Canvas**: 2D drawing context
- **Drawable**: Abstract painter
- **PixelFlinger:** Rasterizer (OpenGL JIT for ARM)
- **SGL**: 2D drawing API (Skia)
- **Surface**: Drawing buffer
- **SurfaceFlinger**: Surface manager
- **View**: UI widget
- **ViewGroup/Layout:** UI widget container

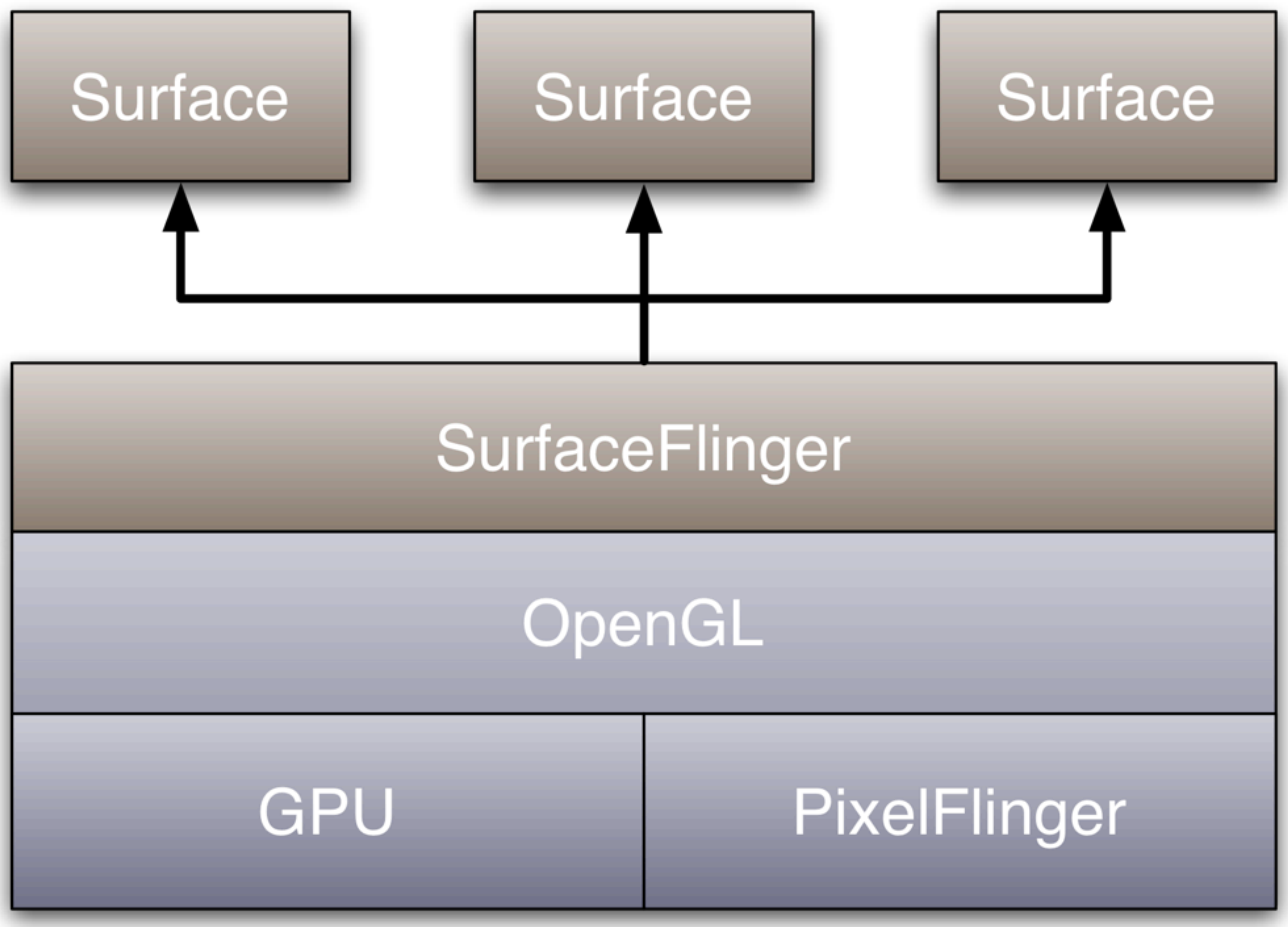
Architecture



Architecture



Architecture



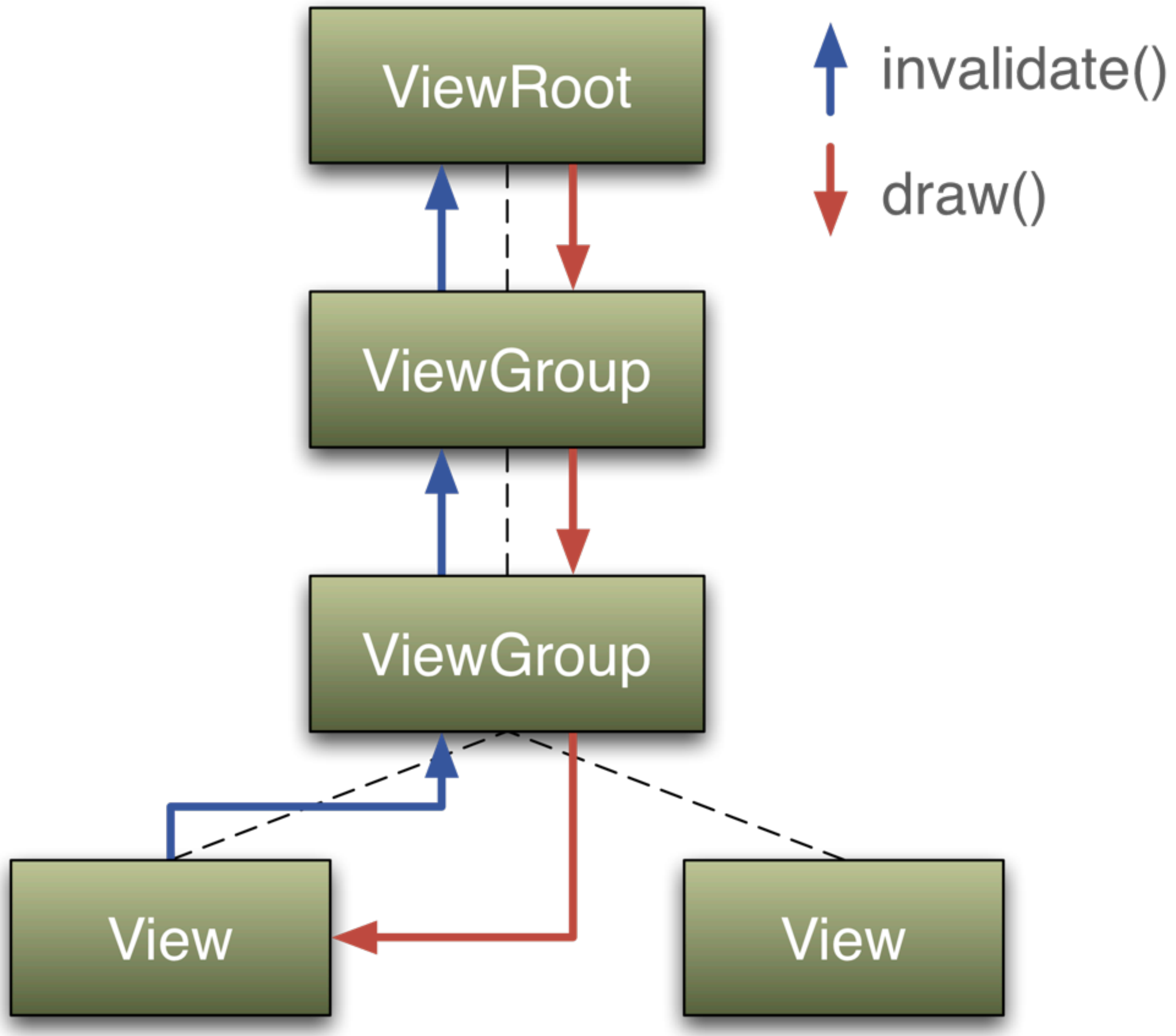
How to draw

```java
public class CustomView extends View {
    @Override
    protected void onDraw(Canvas canvas) {
        // draw stuff
    }
}
```

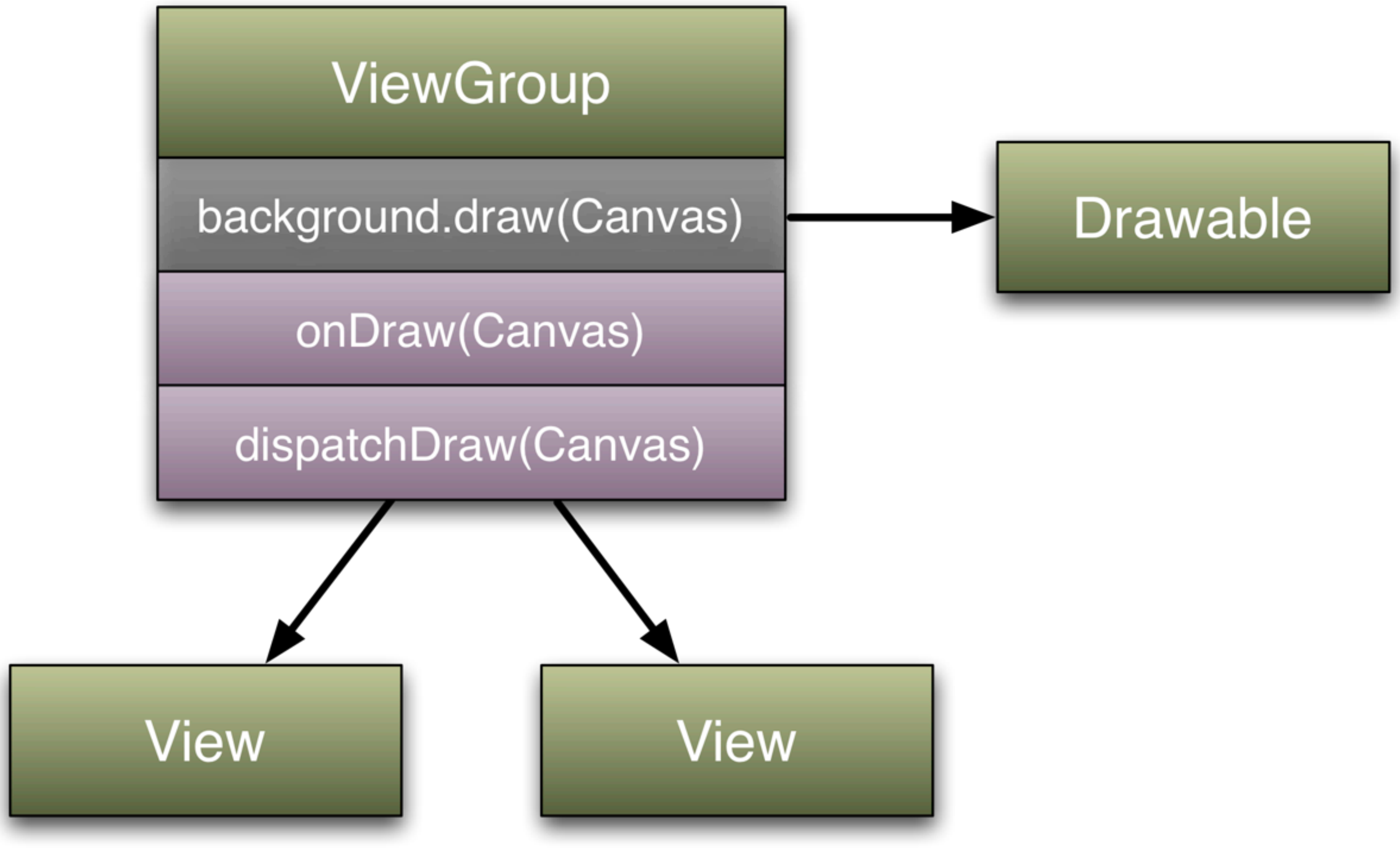
How to draw

```java
public class CustomDrawable extends Drawable {
    @Override
    public void draw(Canvas canvas) {
        // draw stuff
    }
}
```

Redrawing



Drawing sequence



Agenda

- Architecture
- Graphics
- Animation
- Performance

DEMO

3D Reflection

Fundamentals

- Paints
- Gradients
- Transfer modes
- 3D Transformations
- Shadows

About paints

- Canvas is mostly stateless
  - Transformation matrix
- Paint contains the state
  - Opacity, color and color filter
  - Transfer mode, mask filter and shader
  - Anti-aliasing, filtering and dithering
  - Stroke and fill

DEMO

Color filter in Home

Screen transfer mode in Shelves

Faded edges in lists

Gradients

- Shader
  - Horizontal span of colors
- LinearGradient
- RadialGradient
- SweepGradient

Gradients

```
Paint mPaint = new Paint();
mPaint.setShader(new LinearGradient(
      0, 0, 0, 20.0f, 0xFF000000, 0,
     TileMode.CLAMP));

// in onDraw(Canvas)
canvas.drawRect(0.0f, 0.0f,
      20.0f, 20.0f, mPaint);
```

Transfer modes

- In Java2D, AlphaComposite
- Does more
- Modes
  - Porter-Duff (SrcOver, Atop, DstOut, etc.)
  - Color blending (Screen, Darken, Multiply, etc.)

Transfer modes

```java
Shader gradientShader = new LinearGradient(0, 0, 0, 1,
    0xFF000000, 0, TileMode.CLAMP);

Shader bitmapShader = new BitmapShader(mBitmap,
    TileMode.CLAMP, TileMode.CLAMP);

Shader composeShader = new ComposeShader(
    bitmapShader, gradientShader,
    new PorterDuffXfermode(Mode.DST_OUT));

Paint mPaint = new Paint();
mPaint.setShader(composeShader);
```

3D transformations

- 2D Canvas transformations
  - scale(), translate(), rotate()
- Canvas uses a 4x4 transformation matrix
  - 3D transformations
- Use android.graphics.Camera

3D transformations

```
Camera mCamera = new Camera();
// Z translation
mCamera.translate(0.0f, 0.0f, 350.0f);
// rotation around the Y axis in degrees
mCamera.rotateY(45);

// in onDraw(Canvas)
canvas.save();
canvas.concat(mCamera.getMatrix());
canvas.drawBitmap(bitmap, 0.0f, 0.0f, null);
canvas.restore();
```

DEMO

3D transition

Shadows

```
Paint mShadow = new Paint();
// radius=10, y-offset=2, color=black
mShadow.setShadowLayer(10.0f, 0.0f, 2.0f,
    0xFF000000);

// in onDraw(Canvas)
canvas.drawBitmap(bitmap, 0.0f, 0.0f,
    mShadow);
```

Agenda

- Architecture
- Graphics
- Animation
- Performance

Animation

- Why?
  - Better visual feedback
  - UI appears more responsive
- How?
  - Animation
  - LayoutAnimation

Bring life to your application

- Life is restless
  - Transitions, highlights, progress, motion, etc.
- Animate changes
  - Adding/removing views
- Keep animations short and simple

Animation features

- Start delay
- Start time
- Duration
- Repeat mode
- Repeat count
- Interpolation
- Fill before/after
- Defined in XML or code

Inside animations

- Subclass of Animation
- Tied to a View
  - View.setAnimation()/startAnimation()
- Not driven by a timer
  - But time driven
- Driven by the drawing code
  - View.getDrawingTime()

Inside animations

- Fixed set of animated properties
  - AlphaAnimation
  - RotateAnimation
  - ScaleAnimation
  - TranslateAnimation
- View itself is not animated
  - Only a bitmap copy is
  - Drawing cache API

DEMO

Animation in Home

Animation in Shelves

Defining the animation

res/anim/slide_in.xml

```xml
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:fromYDelta="0"
        android:toYDelta="100%"
        android:duration="200" />
    <alpha
        android:fromAlpha="1.0"
        android:toAlpha="0.0"
        android:duration="200" />
</set>
```

Playing the animation

```java
Animation animation;
animation = AnimationUtils.loadAnimation(
        context, R.anim.slide_in);
view.startAnimation(animation);
```

Layout animations

- Apply to a ViewGroup’s children
  - One animation
  - Each child has the same animation
  - Each child has a different start delay
- Layout animation controller
  - Defines the start delay for each child
  - Based on the index, position, column, row, etc.

DEMO

Layout animations

Defining the layout animation

res/anim/layout_fade

```
<gridLayoutAnimation
    android:columnDelay="50%"
    android:directionPriority="row"
    android:direction="right_to_left|bottom_to_top"
    android:animation="@anim/fade" />
```

Playing the layout animation

```
<GridView
    android:layoutAnimation="@anim/layout_fade"

    android:layout_width="fill_parent"
    android:layout_height="fill_parent"/>
```

Transitions

- Long operations
  - Long-press for contextual actions
- Changes
  - Avoid jarring effect
- TransitionDrawable
  - Contains 2 drawables
  - Fade between them

Defining a transition

res/drawable/transition

```
<transition>
    <item android:drawable="@drawable/start"  />
    <item android:drawable="@drawable/end"  />
</transition>
```

Playing a transition

```
TransitionDrawable drawable;
drawable = getDrawable(R.drawable.transition);
view.setBackgroundDrawable(drawable);
drawable.startTransition(1000);
```

DEMO

Transition in Home

Transition in Shelves

Agenda

- Architecture
- Graphics
- Animation
- Performance

Performance

- G1 hardware
  - ~384 Mhz CPU
  - 16 MB of RAM per process
  - ATI Imageon GPU
- Interpreted VM
- Simple Garbage Collector
- SGL is not hardware accelerated
- Native code is not supported (yet)

General optimizations

- Do not allocate at drawing time
- Avoid method calls
  - Especially interface calls
- Avoid invalidate()
- Invalidate only what you need
  - invalidate(left, top, right, bottom)
- Flatten the view hierarchy

DEMO

HierarchyViewer

DDMS

Bitmaps

- Drawable stretch bitmaps
  - Size your bitmap accordingly
  - Bitmap.createScaledBitmap()
  - BitmapFactory.Options.inSampleSize
- Dithering at drawing time is costly
  - Pre-dither bitmaps (Photoshop plugin)
  - BitmapFactory.Options.inDither

Backgrounds

- Remove unnecessary backgrounds
  - No “opaque view” optimization
  - getWindow().setBackgroundDrawable(null)
  - For instance: Home, Google Maps, Shelves
- Prefer ColorDrawable

DEMO

Home

Maps

Shelves

Drawing cache

- Intermediate bitmap
- Special API
  - View.setDrawingCacheEnabled()
  - View.buildDrawingCache()
  - View.getDrawingCache()
- Sometimes managed automatically
  - ViewGroup (animations)
  - ListView (scrolling)

DEMO

Home

ListView

Concluding statement

Filthy Rich Clients are possible on today's mobile devices. Powerful APIs and hardware open new possibilities that have barely been explored.

Q&A

romainguy@android.com

Thanks for your attention!

http://www.android.com
http://source.android.com
http://code.google.com/android