To enable libraries to do more, library should provide new low-level capabilities that expose the possibilities of the underlying platform as closely as possible.

2D Graphics in Android

www.kandroid.org

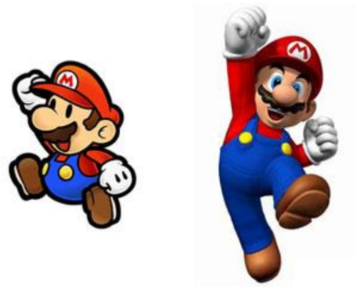
이경민 (snailee@gmail.com)

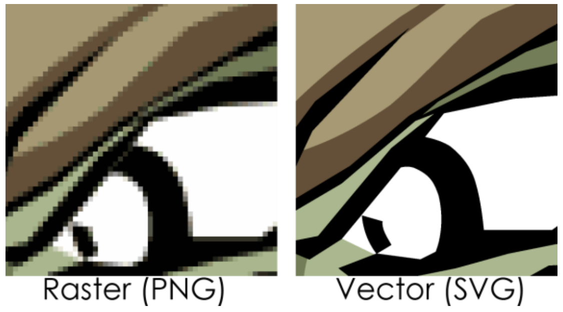
Contents

Basic Terminology

- **2D computer graphics** is the computer-based generation of digital images—mostly from **two-dimensional models** (such as 2D **geometric models**, **text**, and digital **images**) and by techniques specific to them.
- **3D computer graphics** (in contrast to 2D computer graphics) are graphics that use a **three-dimensional representation of geometric data** (often Cartesian) that is stored in the computer for the purposes of **performing calculations** and **rendering 2D images**.
- 3D computer graphics rely on many of the same algorithms as 2D computer **vector graphics** in the wire-frame model and 2D computer **raster graphics** in the final rendered display.
- **Vector graphics** is the use of geometrical primitives such as **points**, **lines**, **curves**, and **shapes** or **polygons**—all of which are based on mathematical expressions—to represent images in computer graphics. Vector graphics are based on **vectors** (also called **paths**), which lead through locations called control points or nodes.
- A **raster graphics** image is a dot matrix data structure representing a generally rectangular grid of **pixels**, or points of **color**, viewable via a monitor, paper, or other display medium.

From Wikipedia

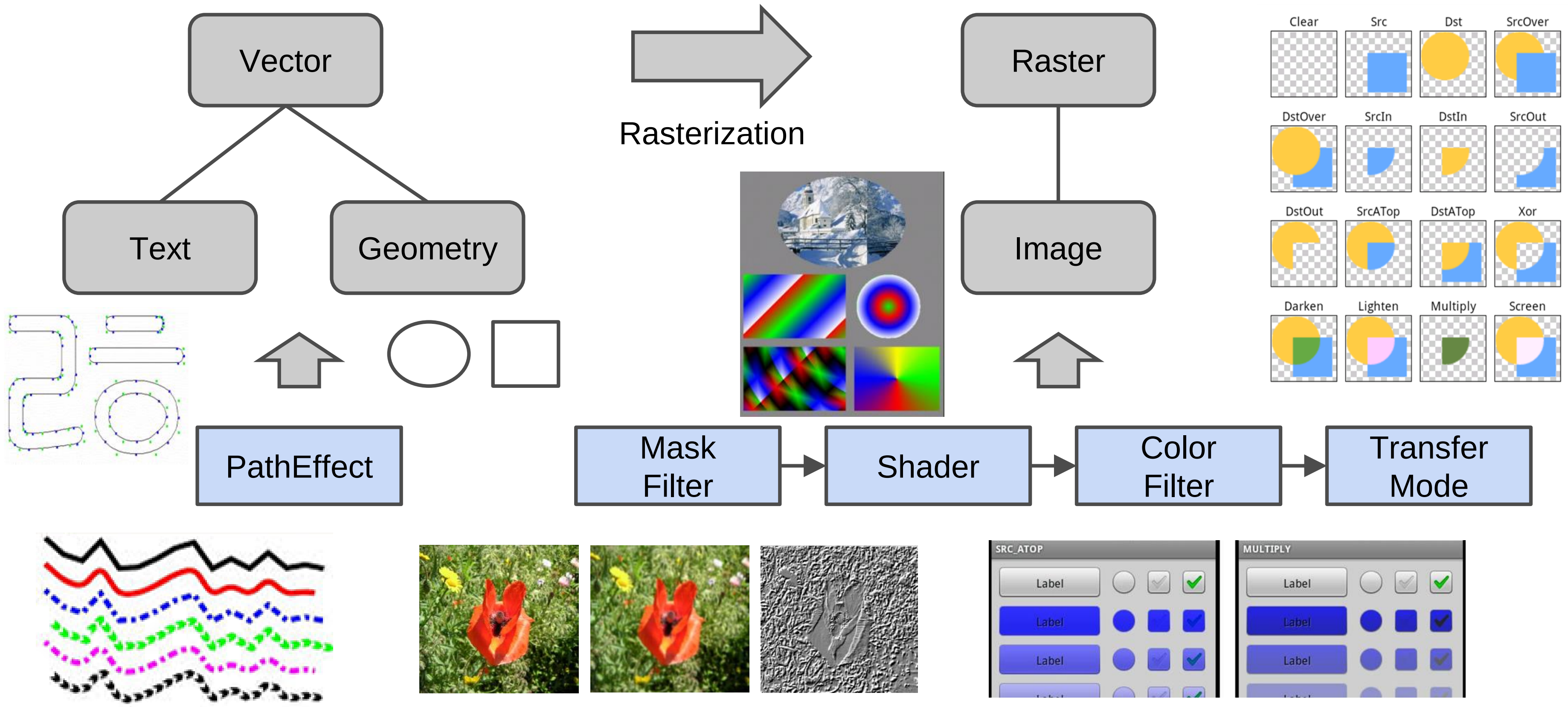

Source: http://austinvisuals.com/wp-content/uploads/2013/03/mario2d3d.jpg



Source: http://zyanger.blogspot.kr/2010/09/week-6-clear-raster-vs-vector-graphics.html

What 2D graphics must support?

- Types of 2D Drawable: Texts, Geometries, Images
- Types of 2D Effects: Path Effects, Mask Filters, Shaders, Color Filters, Transfer(or Blend) Modes

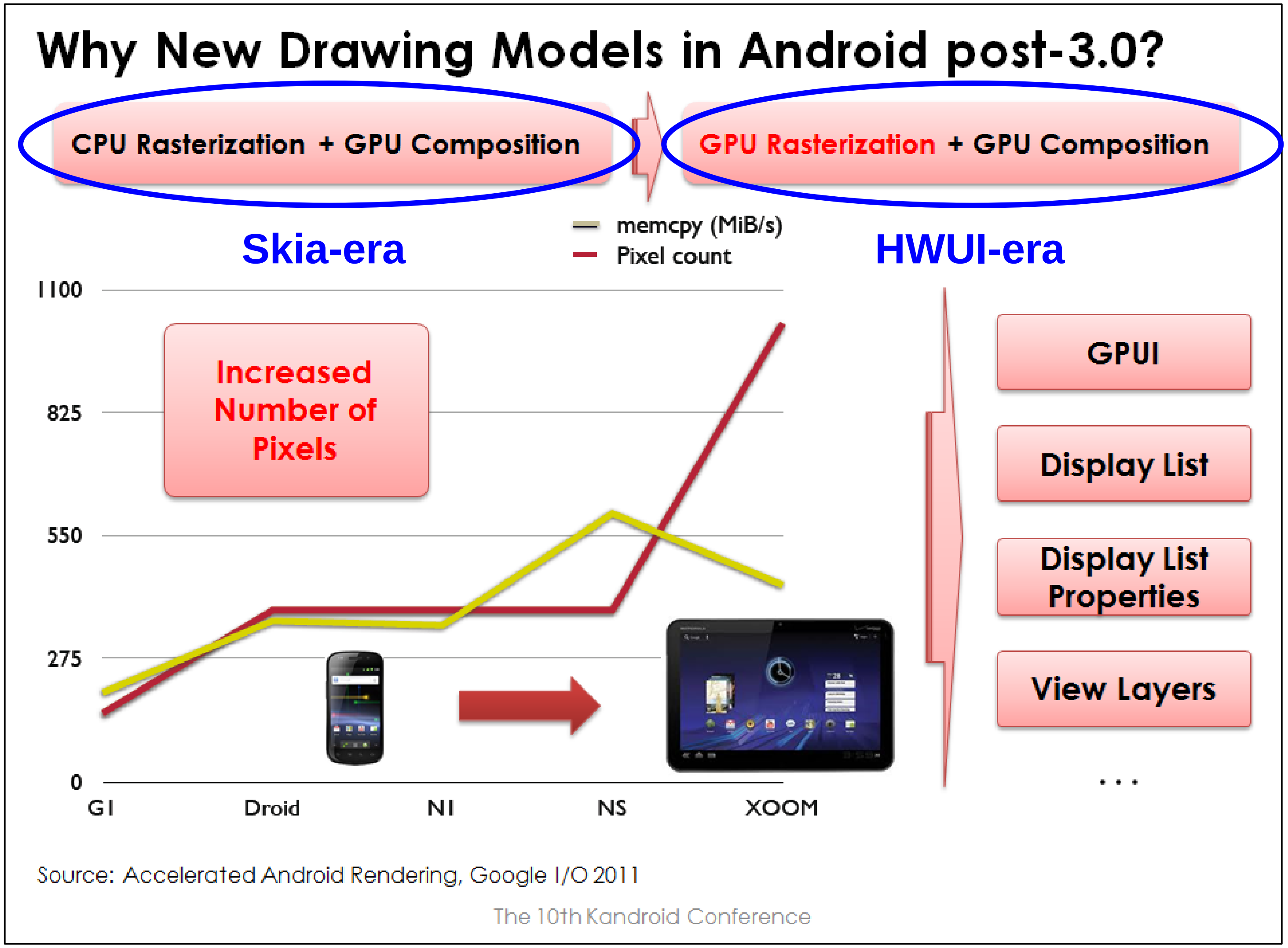


Source: http://lodev.org/cgtutor/filtering.html

"Image filtering modify the pixels in an image based on some function of a local neighborhood of the pixels."
From http://alumni.media.mit.edu/~maov/classes/vision09/lect/09_Image_Filtering_Edge_Detection_09.pdf

Contents

How Android supports 2D graphics?



Why New Drawing Models in Android post-3.0?

CPU Rasterization + GPU Composition → GPU Rasterization + GPU Composition

Skia-era

HWUI-era

Source: Accelerated Android Rendering, Google I/O 2011

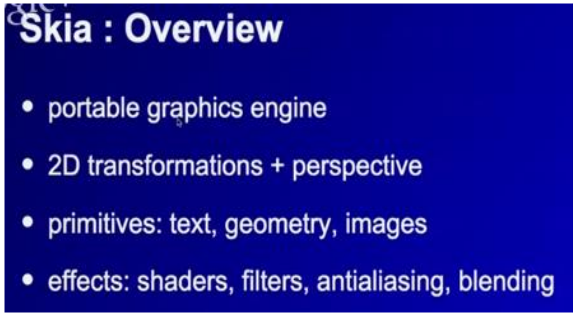
The 10th Kandroid Conference

Source: “Skia and Freetype - Android 2D Graphics Essentials” at 10th Kandroid Conference (2012)

Skia-era: Why Skia?

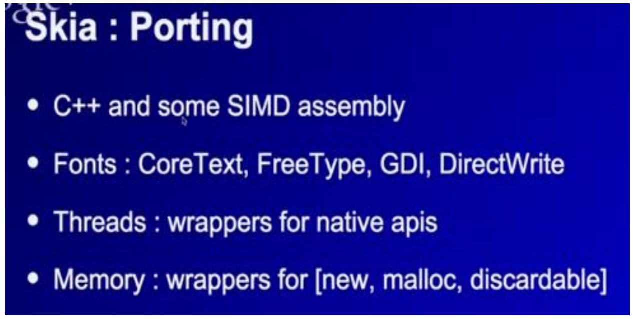
Skia is the open source 2D graphics library used by Chrome, Chrome OS, Firefox, Firefox OS, Android, and other products. We strive to provide a single set of APIs for **accurate**, **high performance** rendering across **a variety of hardware and software platforms**.

Source: https://docs.google.com/document/d/1Q4-YN8wDY9Q3L7gkqOJmmCLM73dj3tr9epUHL1vMZm4/



Skia : Overview

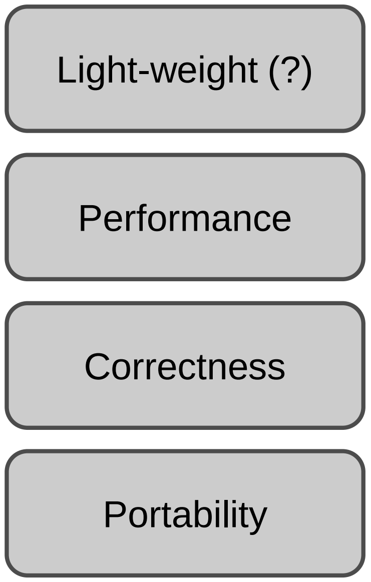
- portable graphics engine
- 2D transformations + perspective
- primitives: text, geometry, images
- effects: shaders, filters, antialiasing, blending



Skia : Porting

- C++ and some SIMD assembly
- Fonts : CoreText, FreeType, GDI, DirectWrite
- Threads : wrappers for native apis
- Memory : wrappers for [new, malloc, discardable]

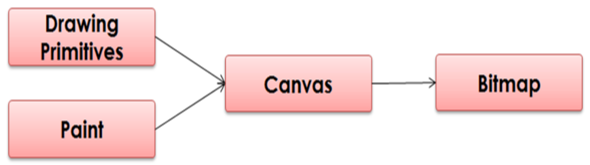
Source: "Skia Update" at BlinkOn 3 (2014)
https://www.youtube.com/watch?v=SU58JHK0-3o

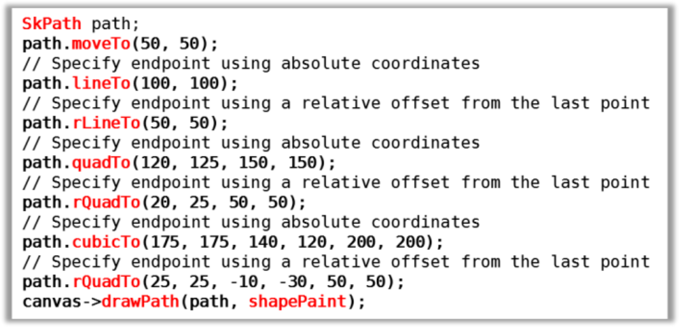


Skia

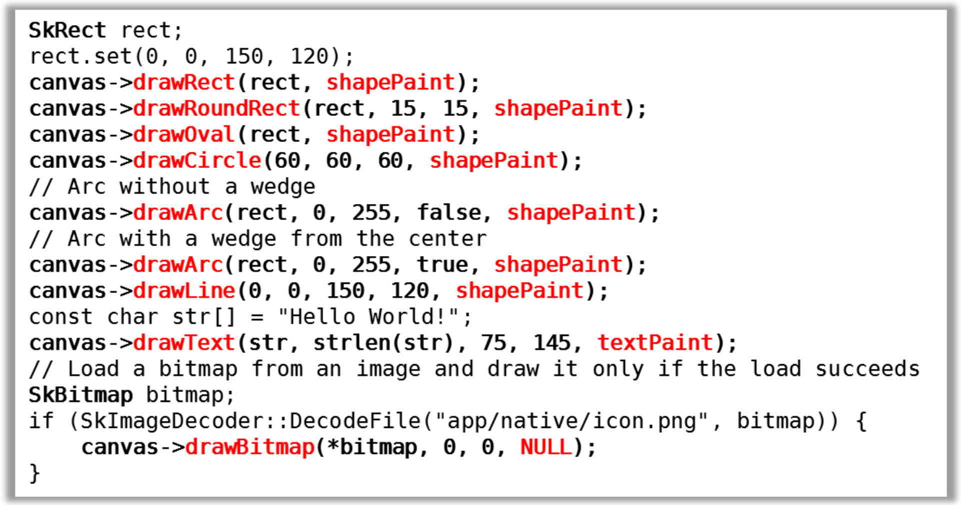
Next Skia

Skia-era: Skia API





```
SkPath path;
path.moveTo(50, 50);
// Specify endpoint using absolute coordinates
path.lineTo(100, 100);
// Specify endpoint using a relative offset from the last point
path.rLineTo(50, 50);
// Specify endpoint using absolute coordinates
path.quadTo(120, 125, 150, 150);
// Specify endpoint using a relative offset from the last point
path.rQuadTo(20, 25, 50, 50);
// Specify endpoint using absolute coordinates
path.cubicTo(175, 175, 140, 120, 200, 200);
// Specify endpoint using a relative offset from the last point
path.rQuadTo(25, 25, -10, -30, 50, 50);
canvas->drawPath(path, shapePaint);
```

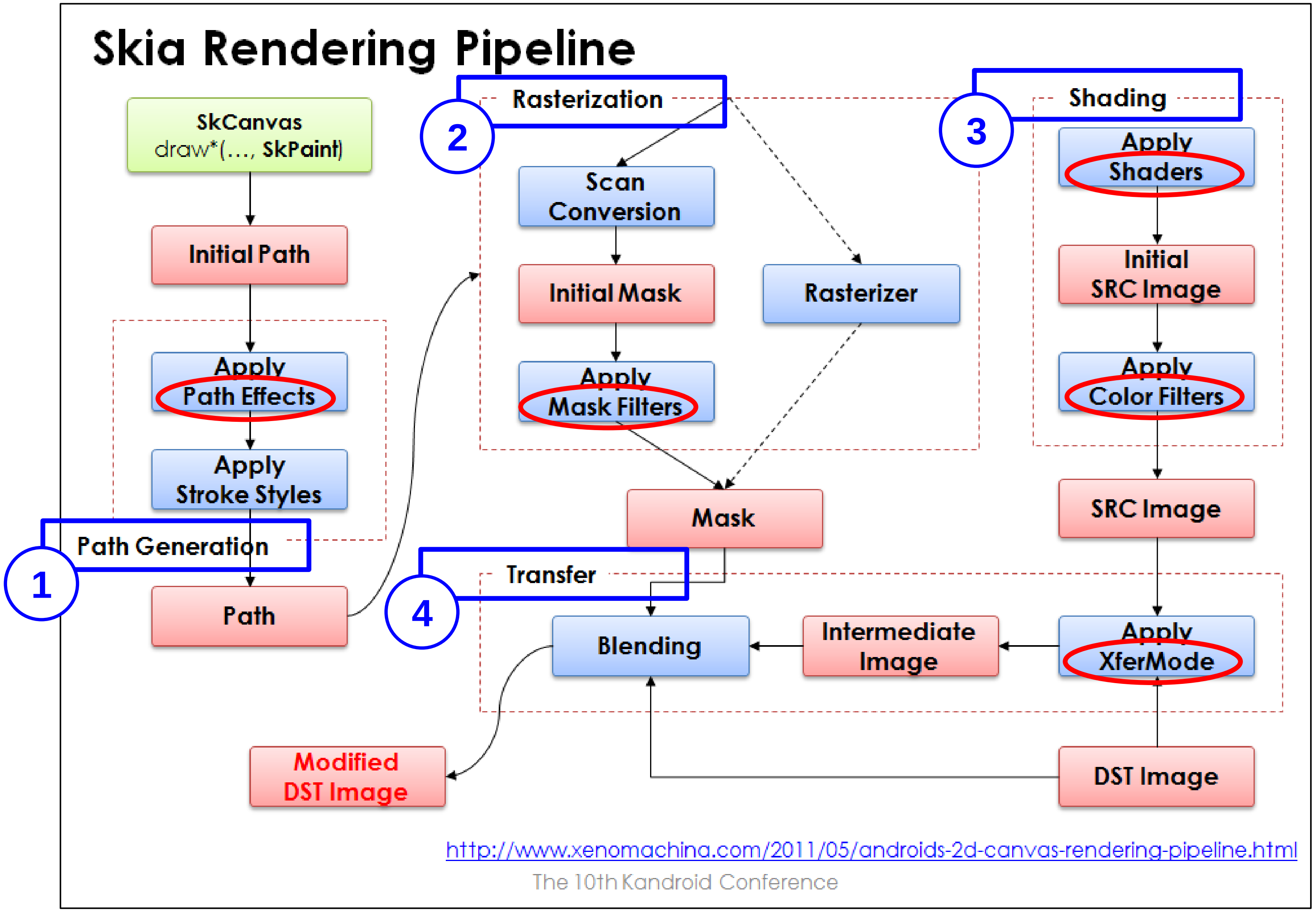


```
SkRect rect;
rect.set(0, 0, 150, 120);
canvas->drawRect(rect, shapePaint);
canvas->drawRoundRect(rect, 15, 15, shapePaint);
canvas->drawOval(rect, shapePaint);
canvas->drawCircle(60, 60, 60, shapePaint);
// Arc without a wedge
canvas->drawArc(rect, 0, 255, false, shapePaint);
// Arc with a wedge from the center
canvas->drawArc(rect, 0, 255, true, shapePaint);
canvas->drawLine(0, 0, 150, 120, shapePaint);
const char str[] = "Hello World!";
canvas->drawText(str, strlen(str), 75, 145, textPaint);
// Load a bitmap from an image and draw it only if the load succeeds
SkBitmap bitmap;
if (SkImageDecoder::DecodeFile("app/native/icon.png", bitmap)) {
    canvas->drawBitmap(*bitmap, 0, 0, NULL);
}
```

Source: “Skia and Freetype - Android 2D Graphics Essentials” at 10th Kandroid Conference (2012)

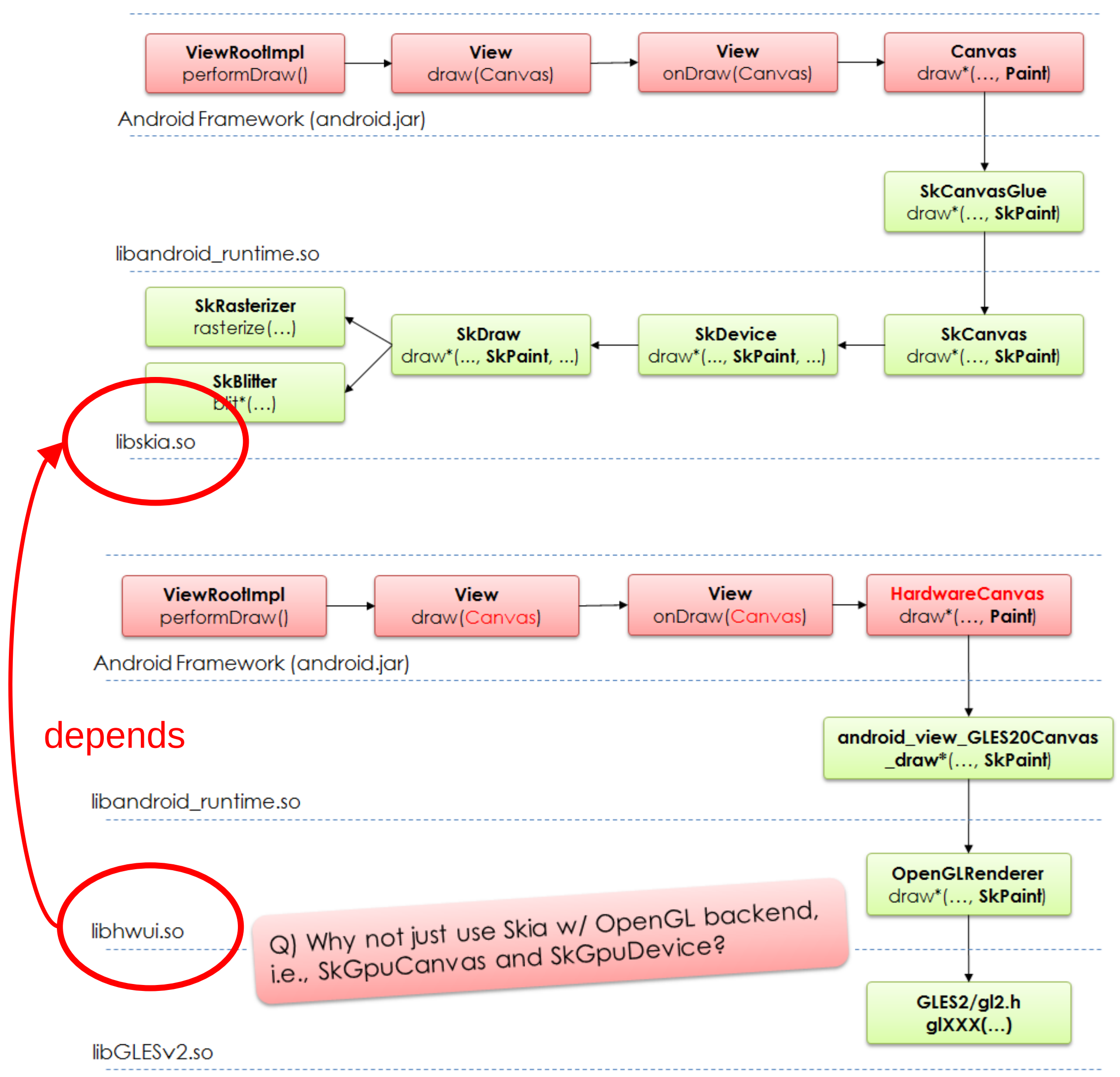
- **SkCanvas**: main drawing API (drawRect, drawText, drawLine, drawPath, etc)
- **SkPaint**: encapsulates styling of draw calls (color, path style, blending mode, font, etc)
- **SkDevice**: abstracts the backend (**SkBitmapDevice**, **SkGpuDevice**, **SkPDFDevice**, etc)
- **SkPicture**, **SkPicturePlayback**: records and replays draw operations

Skia-era: Skia Rendering Pipeline

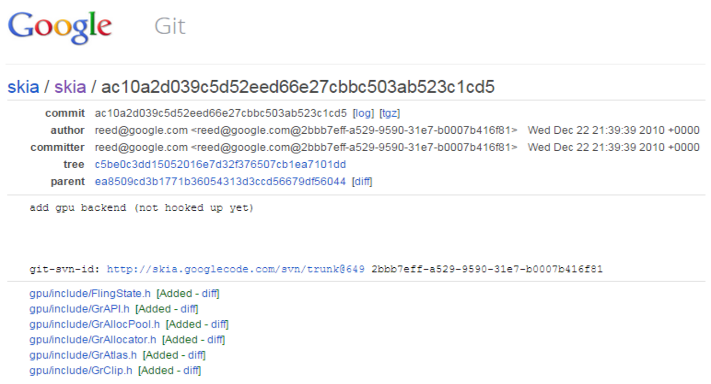


Source: "Skia and Freetype - Android 2D Graphics Essentials" at 10th Kandroid Conference (2012)

HWUI-era: Why not Skia's GPU backend?



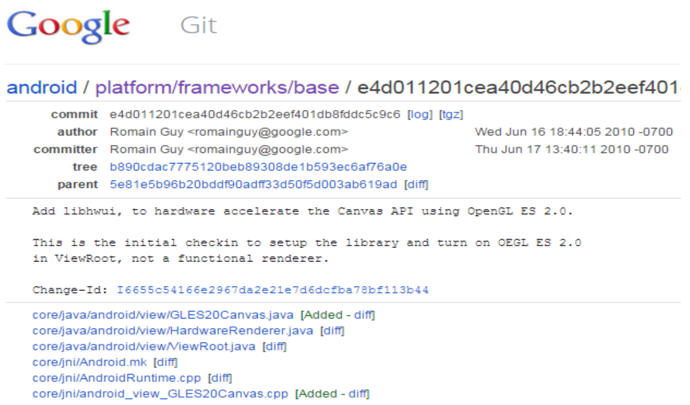
ViewRootImpl performDraw()
View draw(Canvas)
View onDraw(Canvas)
Canvas draw*(..., Paint)
Android Framework (android.jar)
SkCanvasGlue draw*(..., SkPaint)
libandroid_runtime.so
SkRasterizer rasterize(...)
SkDraw draw*(..., SkPaint, ...)
SkDevice draw*(..., SkPaint, ...)
SkCanvas draw*(..., SkPaint)
SkBlitter blit*(...)
libskia.so
ViewRootImpl performDraw()
View draw(Canvas)
View onDraw(Canvas)
HardwareCanvas draw*(..., Paint)
Android Framework (android.jar)
depends
android_view_GLES20Canvas _draw*(..., SkPaint)
libandroid_runtime.so
OpenGLRenderer draw*(..., SkPaint)
libhwui.so
Q) Why not just use Skia w/ OpenGL backend, i.e., SkGpuCanvas and SkGpuDevice?
GLES2/gl2.h glXXX(...)
libGLESv2.so



skia / skia / ac10a2d039c5d52eed66e27cbbc503ab523c1cd5

```
   commit  ac10a2d039c5d52eed66e27cbbc503ab523c1cd5 [log] [tgz]
   author  reed@google.com <reed@google.com@2bbb7eff-a529-9590-31e7-b0007b416f81>  Wed Dec 22 21:39:39 2010 +0000
committer  reed@google.com <reed@google.com@2bbb7eff-a529-9590-31e7-b0007b416f81>  Wed Dec 22 21:39:39 2010 +0000
     tree  c5be0c3dd15052016e7d32f376507cb1ea7101dd
   parent  ea8509cd3b1771b36054313d3ccd56679df56044 [diff]

add gpu backend (not hooked up yet)



git-svn-id: http://skia.googlecode.com/svn/trunk@649 2bbb7eff-a529-9590-31e7-b0007b416f81

gpu/include/FlingState.h [Added - diff]
gpu/include/GrAPI.h [Added - diff]
gpu/include/GrAllocPool.h [Added - diff]
gpu/include/GrAllocator.h [Added - diff]
gpu/include/GrAtlas.h [Added - diff]
gpu/include/GrClip.h [Added - diff]
```

Add gpu backend (2010/12)



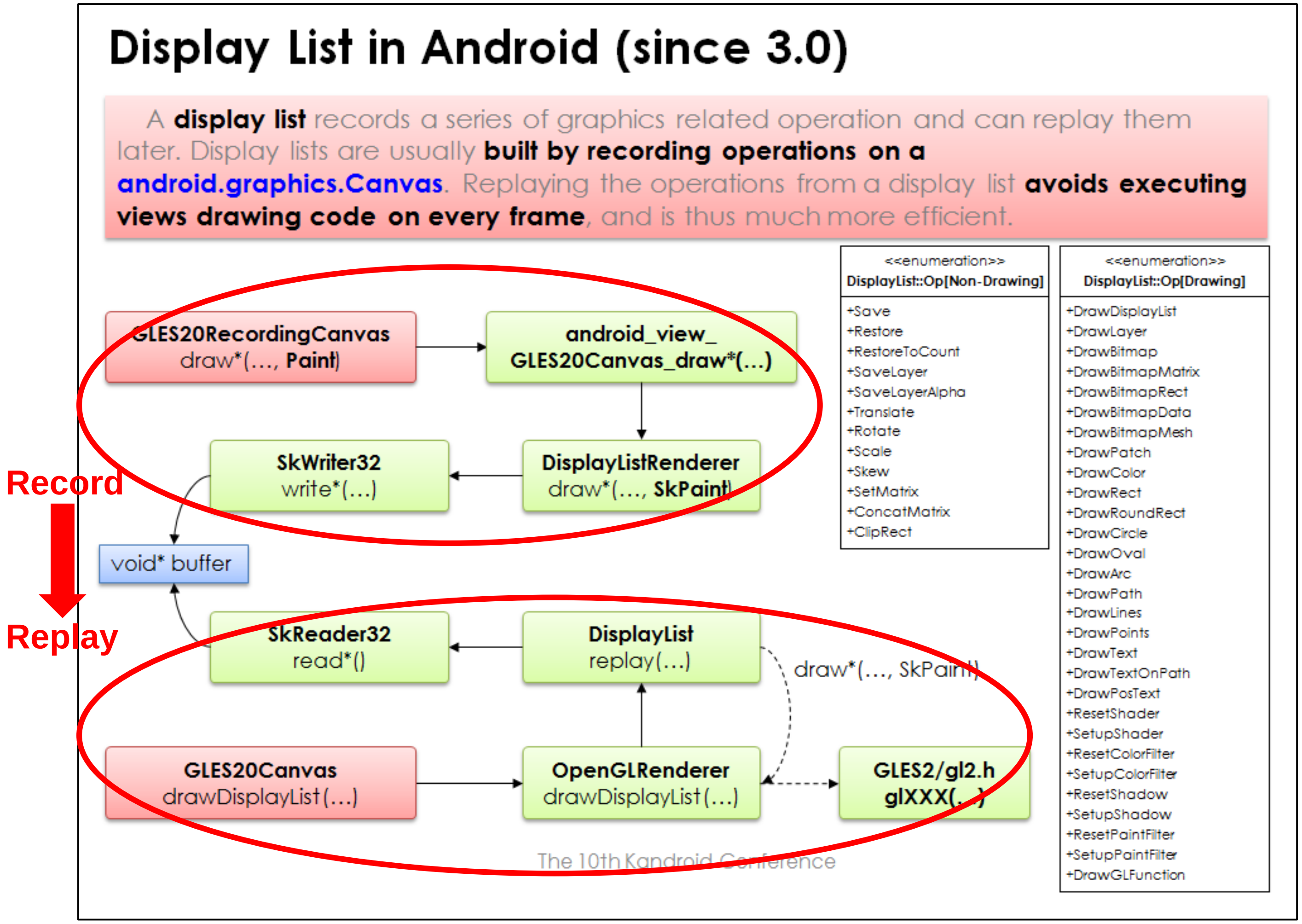
android / platform/frameworks/base / e4d011201cea40d46cb2b2eef401

```
   commit  e4d011201cea40d46cb2b2eef401db8fddc5c9c6 [log] [tgz]
   author  Romain Guy <romainguy@google.com>        Wed Jun 16 18:44:05 2010 -0700
committer  Romain Guy <romainguy@google.com>        Thu Jun 17 13:40:11 2010 -0700
     tree  b890cdac7775120beb89308de1b593ec6af76a0e
   parent  5e81e5b96b20bddf90adff33d50f5d003ab619ad [diff]

Add libhwui, to hardware accelerate the Canvas API using OpenGL ES 2.0.

This is the initial checkin to setup the library and turn on OEGL ES 2.0
in ViewRoot, not a functional renderer.

Change-Id: I6655c54166e2967da2e21e7d6dcfba78bf113b44

core/java/android/view/GLES20Canvas.java [Added - diff]
core/java/android/view/HardwareRenderer.java [diff]
core/java/android/view/ViewRoot.java [diff]
core/jni/Android.mk [diff]
core/jni/AndroidRuntime.cpp [diff]
core/jni/android_view_GLES20Canvas.cpp [Added - diff]
```

Add libhwui (2010/06)

Source: "Skia and Freetype - Android 2D Graphics Essentials" at 10th Kandroid Conference (2012)

HWUI-era: Display List



Display List in Android (since 3.0)

A **display list** records a series of graphics related operation and can replay them later. Display lists are usually **built by recording operations on a android.graphics.Canvas**. Replaying the operations from a display list **avoids executing views drawing code on every frame**, and is thus much more efficient.

Record
Replay
GLES20RecordingCanvas draw*(..., Paint)
android_view_ GLES20Canvas_draw*(...)
SkWriter32 write*(...)
DisplayListRenderer draw*(..., SkPaint)
void* buffer
SkReader32 read*()
DisplayList replay(...)
draw*(..., SkPaint)
GLES20Canvas drawDisplayList(...)
OpenGLRenderer drawDisplayList(...)
GLES2/gl2.h glXXX(...)
The 10th Kandroid Conference

<<enumeration>>
DisplayList::Op[Non-Drawing]

+Save
+Restore
+RestoreToCount
+SaveLayer
+SaveLayerAlpha
+Translate
+Rotate
+Scale
+Skew
+SetMatrix
+ConcatMatrix
+ClipRect

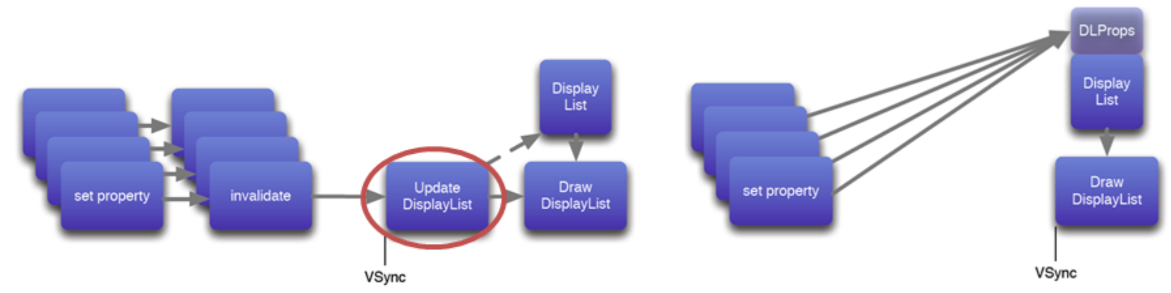
<<enumeration>>
DisplayList::Op[Drawing]

+DrawDisplayList
+DrawLayer
+DrawBitmap
+DrawBitmapMatrix
+DrawBitmapRect
+DrawBitmapData
+DrawBitmapMesh
+DrawPatch
+DrawColor
+DrawRect
+DrawRoundRect
+DrawCircle
+DrawOval
+DrawArc
+DrawPath
+DrawLines
+DrawPoints
+DrawText
+DrawTextOnPath
+DrawPosText
+ResetShader
+SetupShader
+ResetColorFilter
+SetupColorFilter
+ResetShadow
+SetupShadow
+ResetPaintFilter
+SetupPaintFilter
+DrawGLFunction

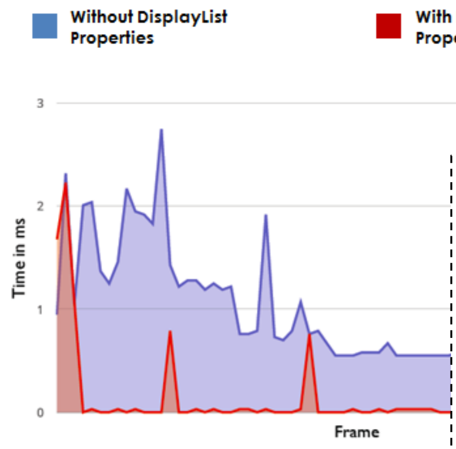
Source: "Skia and Freetype - Android 2D Graphics Essentials" at 10th Kandroid Conference (2012)

HWUI-era: Display List Properties => Render Properties

Display List Properties (since 4.1)



+ bool mClipChildren;
+ float mAlpha;
+ int mMultipliedAlpha;
+ bool mHasOverlappingRendering;
+ float mTranslationX, mTranslationY;
+ float mRotation, mRotationX, mRotationY;
+ float mScaleX, mScaleY;
+ float mPivotX, mPivotY;
+ float mCameraDistance;
+ int mLeft, mTop, mRight, mBottom;
+ int mWidth, mHeight;
+ int mPrevWidth, mPrevHeight;
+ bool mPivotExplicitlySet;
+ bool mMatrixDirty;
+ bool mMatrixIsIdentity;
+ uint32_t mMatrixFlags;
+ SkMatrix* mTransformMatrix;
+ Sk3DView* mTransformCamera;
+ SkMatrix* mTransformMatrix3D;
+ SkMatrix* mStaticMatrix;
+ SkMatrix* mAnimationMatrix;
+ bool mCaching;



Source: For Butter or Worse, Google I/O

The 10th Kandroid Conference

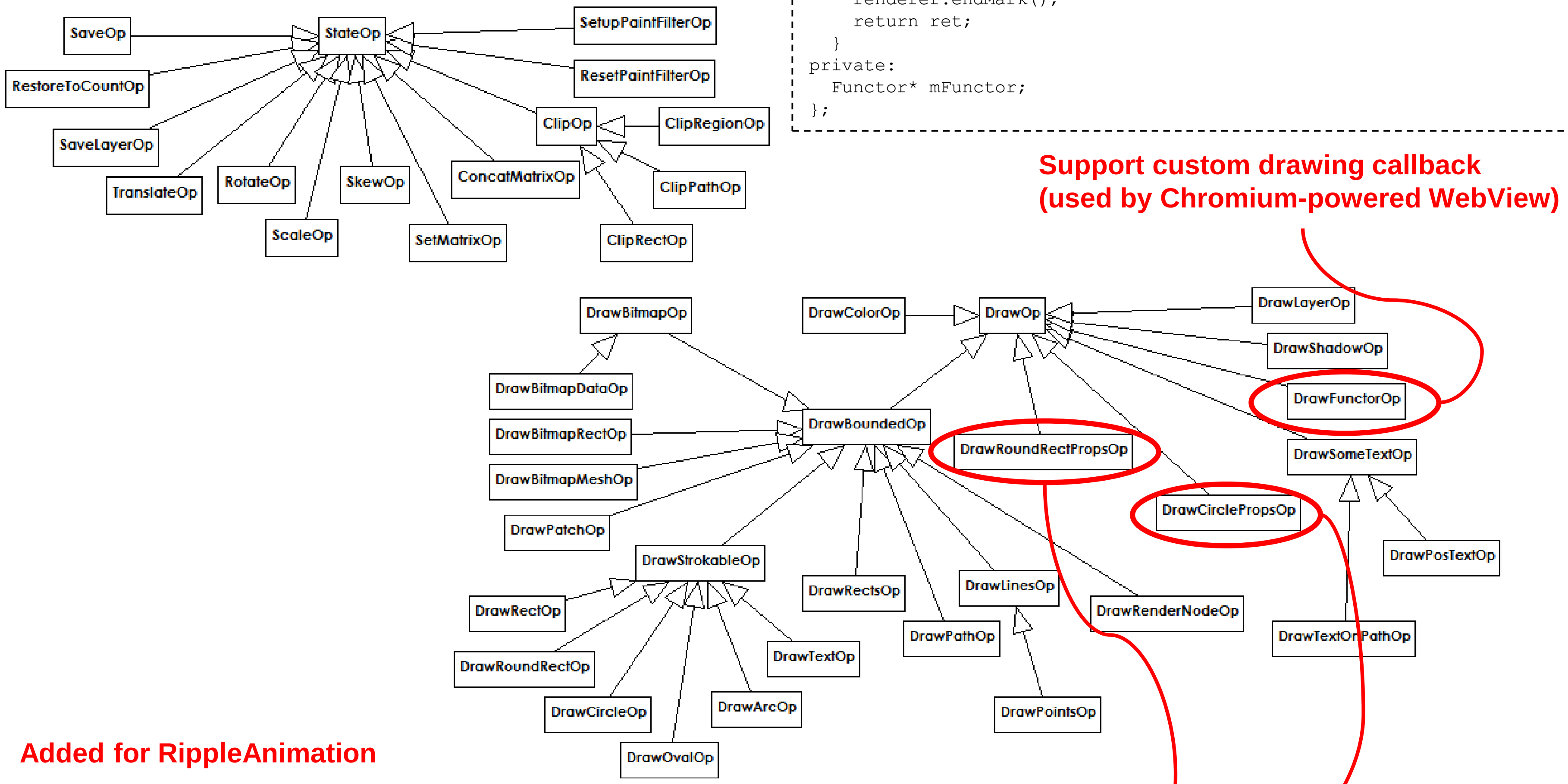
Source: “Skia and Freetype - Android 2D Graphics Essentials” at 10th Kandroid Conference (2012)

```
class LayerProperties
    LayerType mType;
    bool mOpaque;
    uint8_t mAlpha;
    SkXfermode::Mode mMode;
    SkColorFilter* mColorFilter;
```

```
class RenderProperties
    struct PrimitiveFields {
        Outline mOutline;
        RevealClip mRevealClip;
        int mClippingFlags;
        bool mProjectBackwards;
        bool mProjectionReceiver;
        float mAlpha;
        bool mHasOverlappingRendering;
        float mElevation;
        float mTranslationX, mTranslationY, mTranslationZ;
        float mRotation, mRotationX, mRotationY;
        float mScaleX, mScaleY;
        float mPivotX, mPivotY;
        int mLeft, mTop, mRight, mBottom;
        int mWidth, mHeight;
        bool mPivotExplicitlySet;
        bool mMatrixOrPivotDirty;
        Rect mClipBounds;
    } mPrimitiveFields;
    SkMatrix* mStaticMatrix;
    SkMatrix* mAnimationMatrix;
    LayerProperties mLayerProperties;
```

HWUI-era: DisplayListOp

```
class DrawFunctorOp : public DrawOp {
public:
  virtual status_t applyDraw(OpenGLRenderer& renderer,
                             Rect& dirty) {
    renderer.startMark("GL functor");
    status_t ret = renderer.callDrawGLFunction(mFunctor, dirty);
    renderer.endMark();
    return ret;
  }
private:
  Functor* mFunctor;
};
```



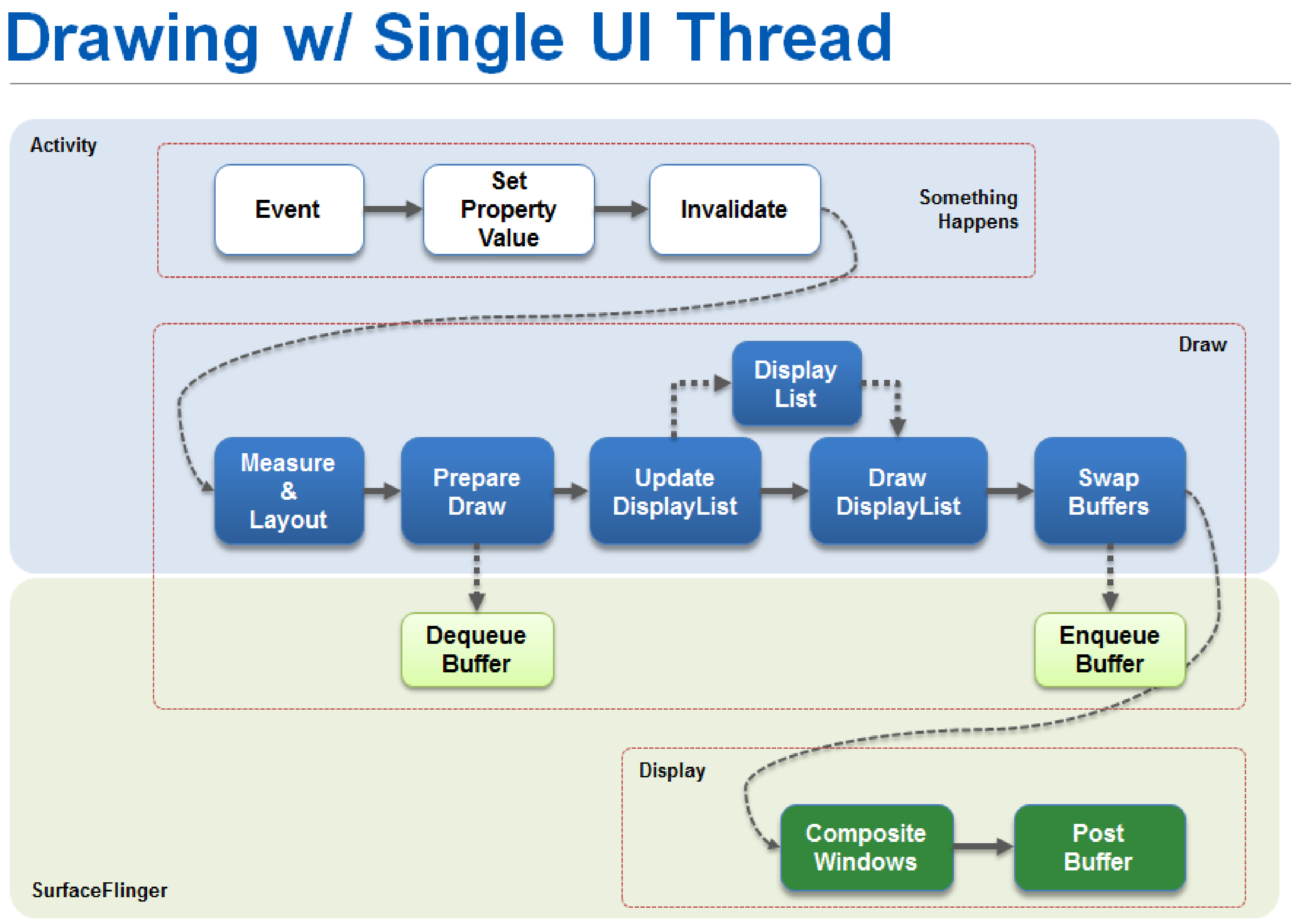
Support custom drawing callback
(used by Chromium-powered WebView)

Added for RippleAnimation

frameworks\base\libs\hwui\DisplayListRenderer.h

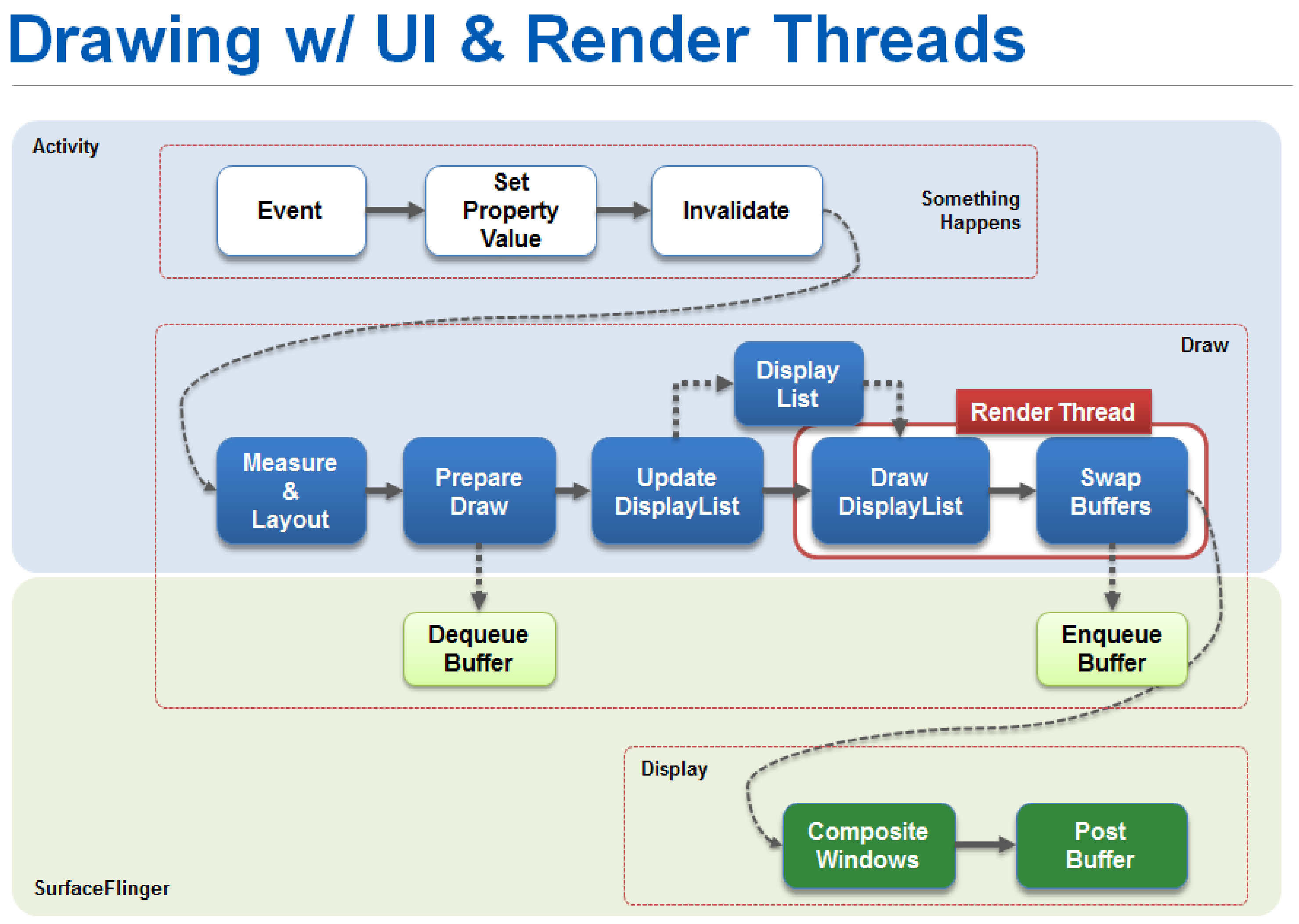
```
virtual status_t drawRoundRect(float left, float top, float right, float bottom,
        float rx, float ry, const SkPaint* paint);
virtual status_t drawRoundRect(CanvasPropertyPrimitive* left, CanvasPropertyPrimitive* top,
        CanvasPropertyPrimitive* right, CanvasPropertyPrimitive* bottom,
        CanvasPropertyPrimitive* rx, CanvasPropertyPrimitive* ry,
        CanvasPropertyPaint* paint);
virtual status_t drawCircle(float x, float y, float radius, const SkPaint* paint);
virtual status_t drawCircle(CanvasPropertyPrimitive* x, CanvasPropertyPrimitive* y,
        CanvasPropertyPrimitive* radius, CanvasPropertyPaint* paint);
```

HWUI-era: UI and Render Thread



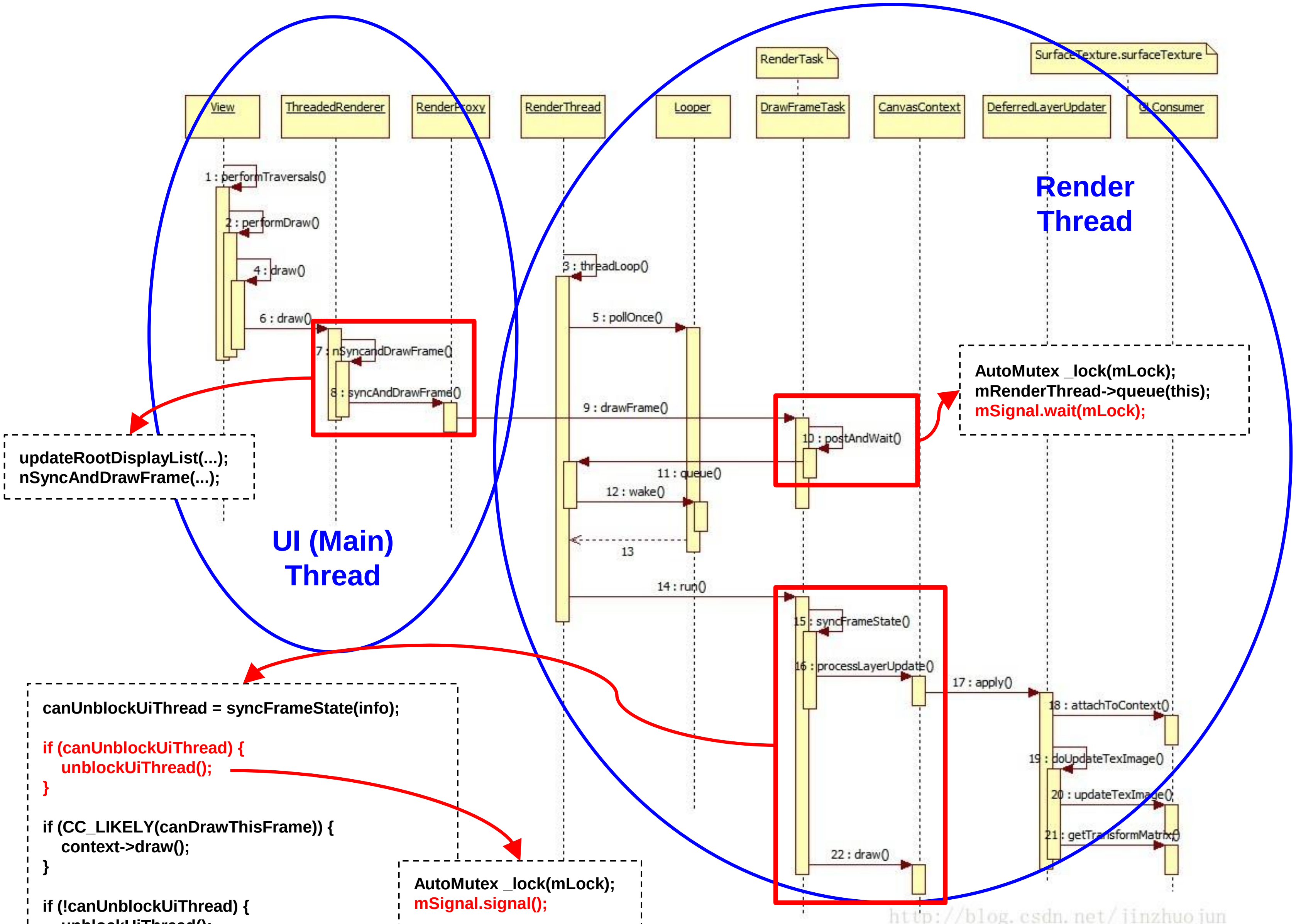
Source: “RE-view of Android L Developer PRE-view”, DEVIEW2014

HWUI-era: UI and Render Thread



Source: “RE-view of Android L Developer PRE-view” at DEVIEW2014

HWUI-era: Sync Between UI and Render Thread



```
updateRootDisplayList(...);
nSyncAndDrawFrame(...);
```

```
AutoMutex _lock(mLock);
mRenderThread->queue(this);
mSignal.wait(mLock);
```

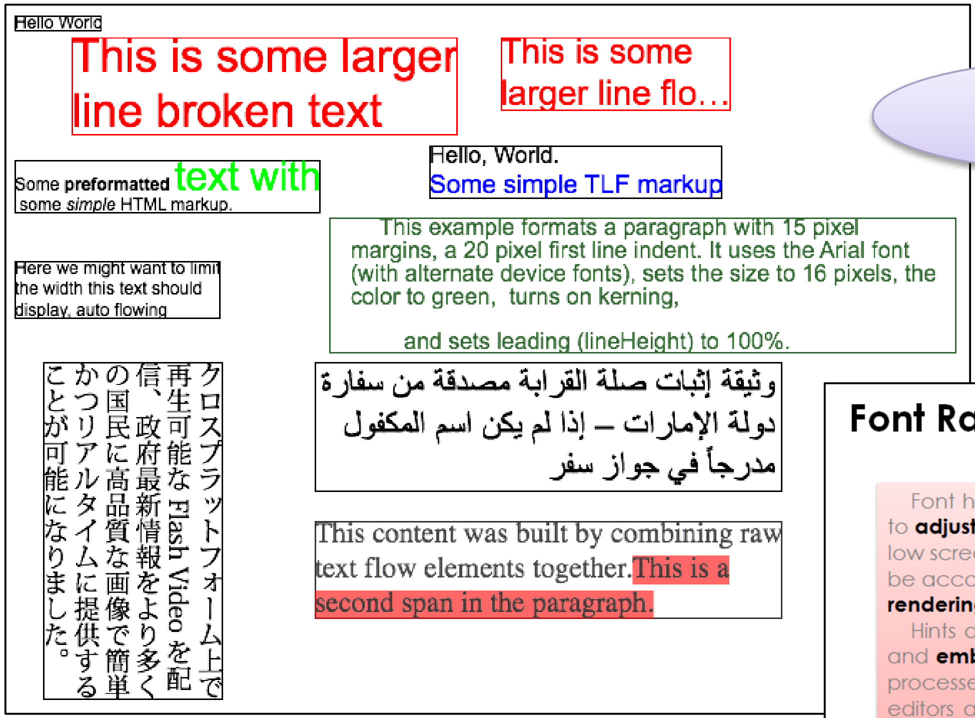
```
canUnblockUiThread = syncFrameState(info);

if (canUnblockUiThread) {
    unblockUiThread();
}

if (CC_LIKELY(canDrawThisFrame)) {
    context->draw();
}

if (!canUnblockUiThread) {
    unblockUiThread();
}
```

```
AutoMutex _lock(mLock);
mSignal.signal();
```

Reference: http://blog.csdn.net/jinzhuojun/article/details/44062175

Contents

Challenges: Layout and Rasterization



Hello World

This is some larger line broken text

This is some larger line flo…

Some preformatted text with some *simple* HTML markup.

Hello, World.
Some simple TLF markup

Here we might want to limit the width this text should display, auto flowing

This example formats a paragraph with 15 pixel margins, a 20 pixel first line indent. It uses the Arial font (with alternate device fonts), sets the size to 16 pixels, the color to green, turns on kerning,

and sets leading (lineHeight) to 100%.

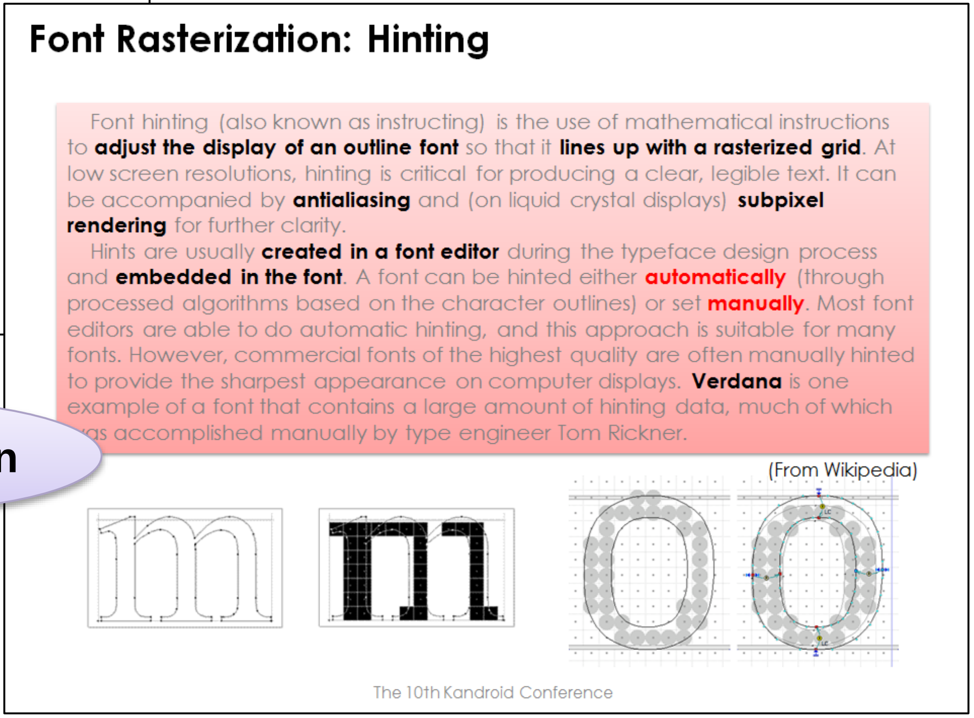
クロスプラットフォーム上で再生可能なFlash Videoを配信、政府最新情報をより多くの国民に高品質な画像で簡単かつリアルタイムに提供することが可能になりました。

وثيقة إثبات صلة القرابة مصدقة من سفارة دولة الإمارات – إذا لم يكن اسم المكفول مدرجاً في جواز سفر

This content was built by combining raw text flow elements together. This is a second span in the paragraph.

Layout

Source: http://wiki.starling-framework.org/extensions/tlfsprite

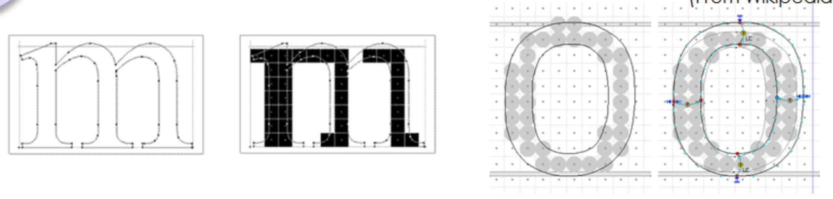


Font Rasterization: Hinting

Font hinting (also known as instructing) is the use of mathematical instructions to **adjust the display of an outline font** so that it **lines up with a rasterized grid**. At low screen resolutions, hinting is critical for producing a clear, legible text. It can be accompanied by **antialiasing** and (on liquid crystal displays) **subpixel rendering** for further clarity.

Hints are usually **created in a font editor** during the typeface design process and **embedded in the font**. A font can be hinted either **automatically** (through processed algorithms based on the character outlines) or set **manually**. Most font editors are able to do automatic hinting, and this approach is suitable for many fonts. However, commercial fonts of the highest quality are often manually hinted to provide the sharpest appearance on computer displays. **Verdana** is one example of a font that contains a large amount of hinting data, much of which was accomplished manually by type engineer Tom Rickner.

(From Wikipedia)



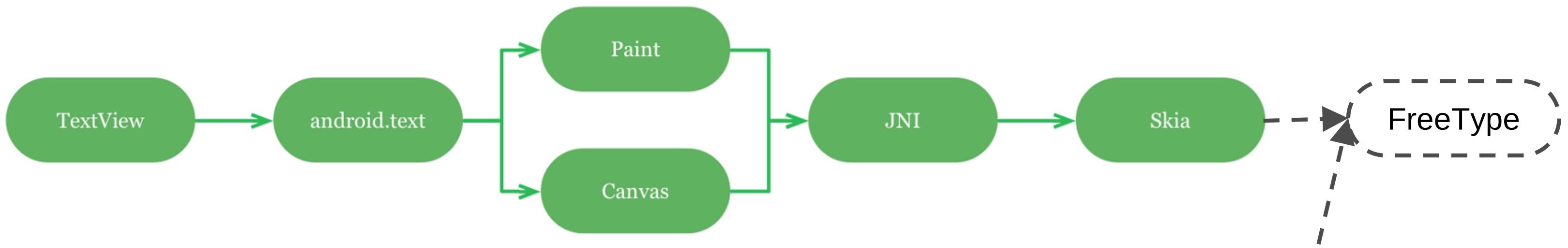
The 10th Kandroid Conference

Rasterization

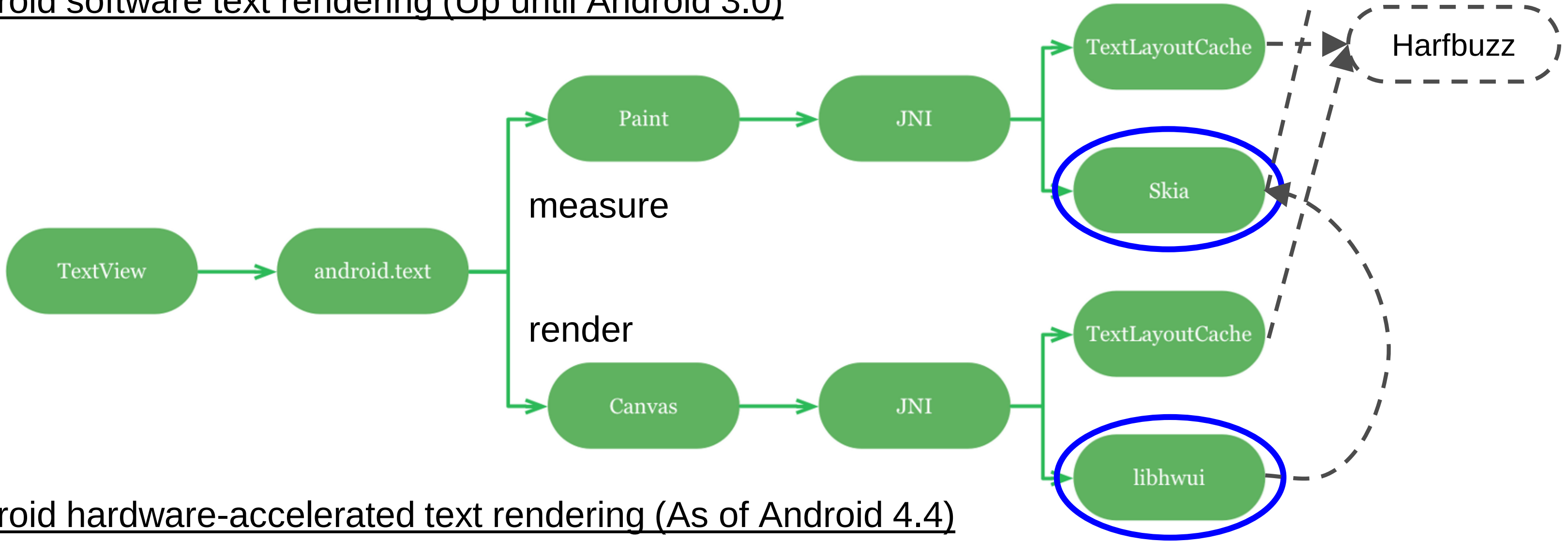
Source: "Skia and Freetype - Android 2D Graphics Essentials" at 10th Kandroid Conference (2012)

Android Text Rendering

- **android.widget.TextView**, a View that handles layout and rendering
- **android.text.***, a collection of classes to create stylized text and **layouts**
- **android.graphics.Paint**, to **measure** text
- **android.graphics.Canvas**, to **render** text



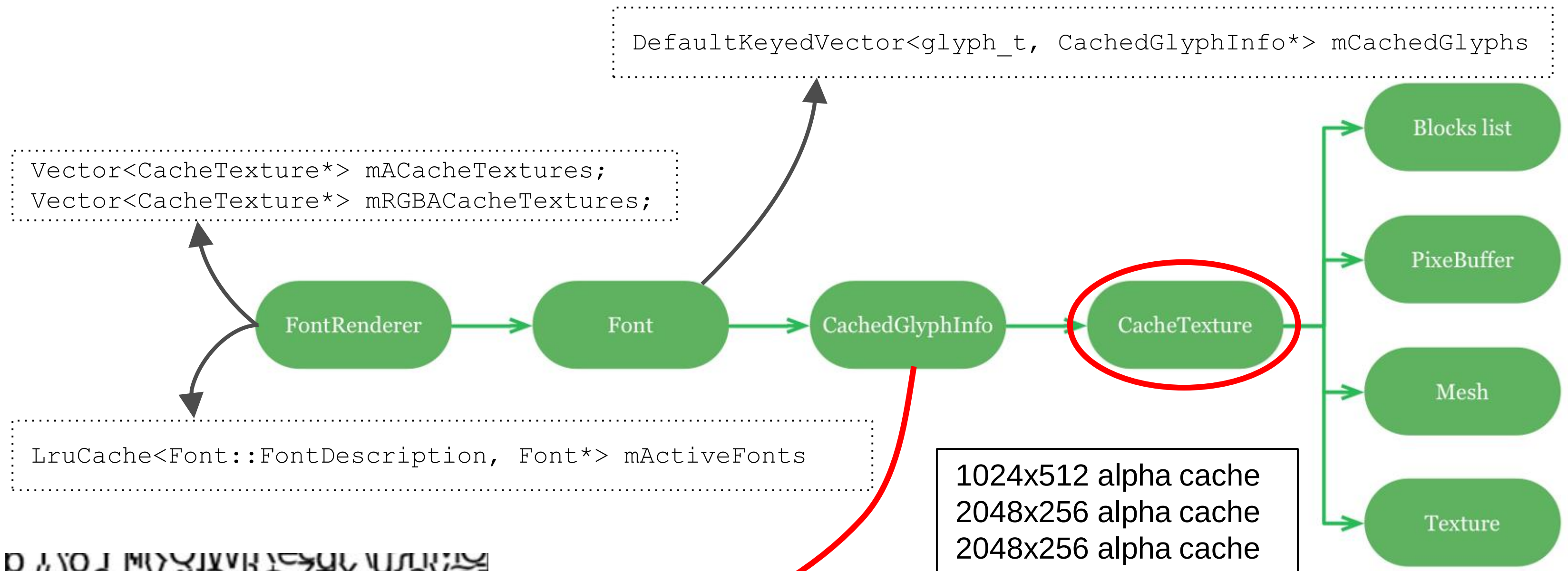
Android software text rendering (Up until Android 3.0)



Android hardware-accelerated text rendering (As of Android 4.4)

Reference: Android’s Font Renderer - Efficient text rendering with OpenGL ES
https://medium.com/@romainguy/androids-font-renderer-c368bbde87d9

Performance Optimization: Caching Architecture

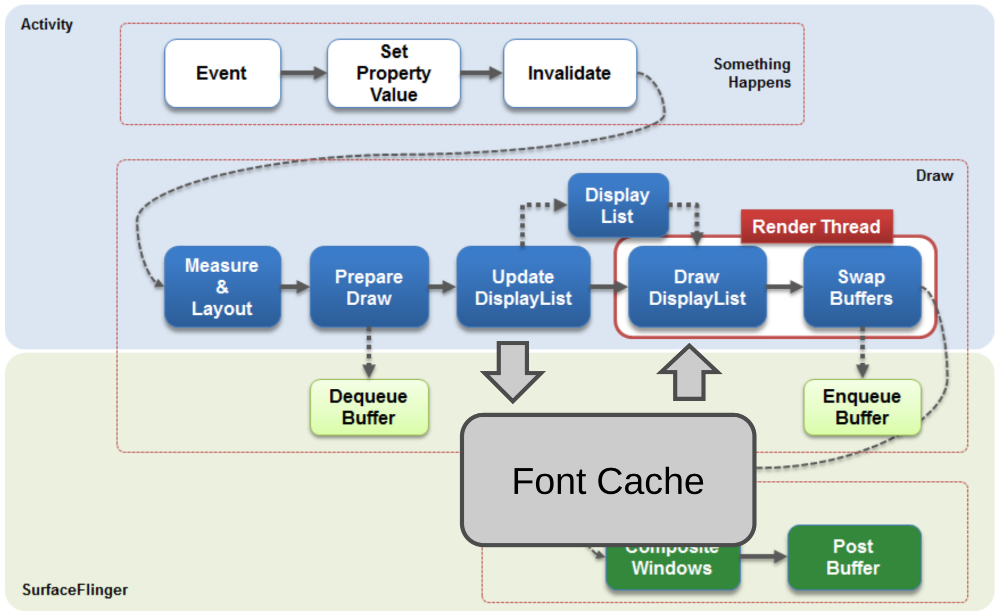

DefaultKeyedVector<glyph_t, CachedGlyphInfo*> mCachedGlyphs
Vector<CacheTexture*> mACacheTextures;
Vector<CacheTexture*> mRGBACacheTextures;
FontRenderer
Font
CachedGlyphInfo
CacheTexture
Blocks list
PixeBuffer
Mesh
Texture
LruCache<Font::FontDescription, Font*> mActiveFonts

1024x512 alpha cache
2048x256 alpha cache
2048x256 alpha cache
2048x512 alpha cache
1024x512 RGBA cache
2048x256 RGBA cache

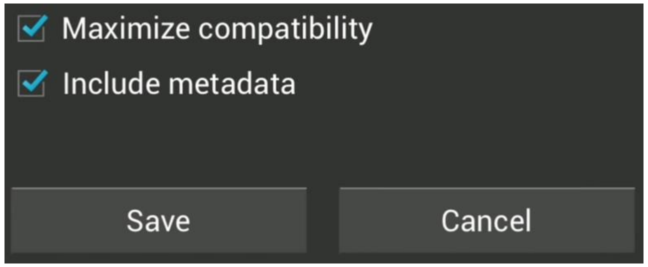
In realtime computer graphics, a **texture atlas** is a large image containing a collection, or "atlas", of sub-images, each of which is a texture for some part of a 2D or 3D object.
From http://en.wikipedia.org/wiki/Texture_atlas

Performance Optimization: Pre-Caching

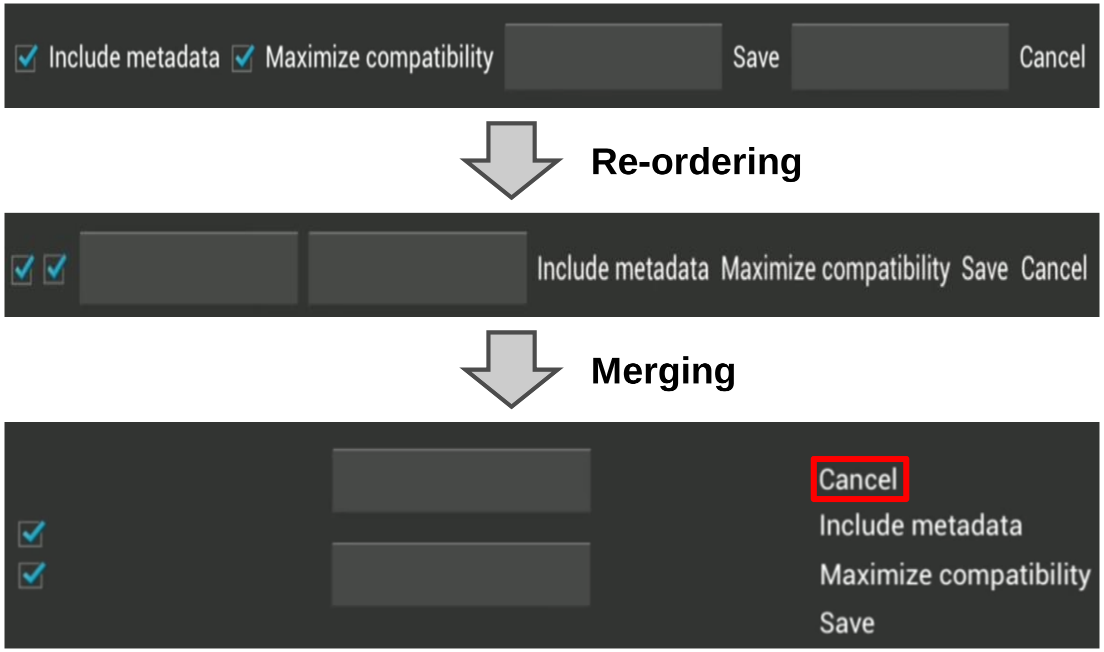
- To completely avoid, or at least minimize, the number of **texture uploads** mid-frame
  - Texture uploads are expensive operations that can stall the CPU and/or the GPU.
  - Even worse, modifying a texture during a frame can create severe memory pressure on some GPU architectures.

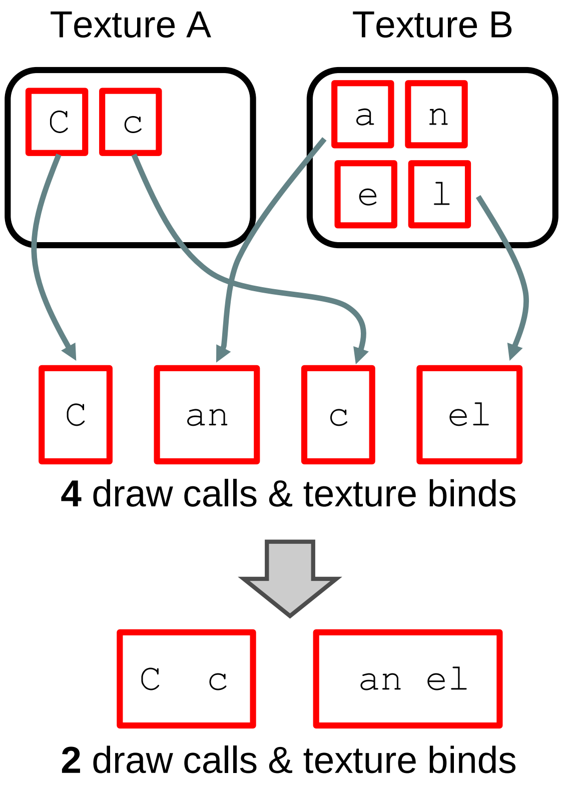


Performance Optimization: Batching & Merging



- Buffers text geometry across multiple draw calls.
- Reduces the number of commands issued to the OpenGL driver.





Source: “Android Graphics Performance” at Google I/O 2013
https://www.youtube.com/watch?v=vQZFaec9NpA

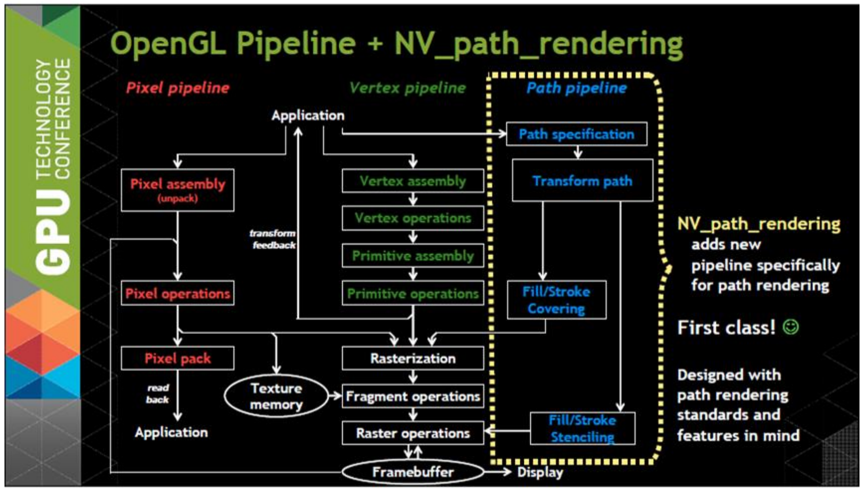
Performance Optimization: Font Rasterization on GPU

glyphy

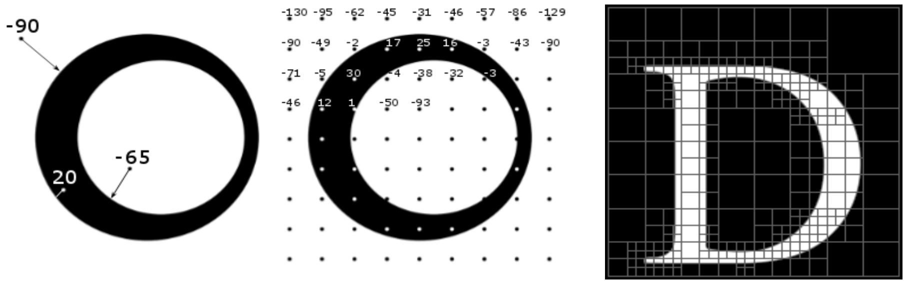
High-quality glyph rendering using OpenGL ES2 shaders

Source: https://code.google.com/p/glyphy/

Source: http://www.ronaldperry.org/SaffronWebPage



Source: https://developer.nvidia.com/nv-path-rendering



SDF: Signed Distance Field

ADF: Adaptively Sampled Distance Field

Q & A