

facebook

Fresco

loading images fast

Jie Wang
Software Engineer
Sep 2016

facebook

Apps using Fresco

Facebook For Android

Messenger

Groups

Pages Manager

Moments

React Native

Twitter

Wikipedia

facebook

Fresco is an Open source library to fetch and show images in an **efficient** way

- Fetch images from different sources
- Decoding
- Transcoding
- Rotate and resize
- Progressive JPEG
- Animated GIF and WebP support
- Caching
- Prefetching to memory and disk cache
- Image post processing
- Instrumentation
- Cancellation and Prioritization

facebook

Integrate Fresco using Android Studio or Gradle

```
1 dependencies {
2     // your app's other dependencies
3     compile 'com.facebook.fresco:fresco:0.13.0'
4 }
```

*With ABI splits & ProGuard: 250 ~ 300 KB

```
1 dependencies {
2     // If your app supports Android versions before Ice Cream Sandwich (API level 14)
3     compile 'com.facebook.fresco:animated-base-support:0.13.0'
4
5     // For animated GIF support
6     compile 'com.facebook.fresco:animated-gif:0.13.0'
7
8     // For WebP support, including animated WebP
9     compile 'com.facebook.fresco:animated-webp:0.13.0'
10    compile 'com.facebook.fresco:webpsupport:0.13.0'
11
12    // For WebP support, without animations
13    compile 'com.facebook.fresco:webpsupport:0.13.0'
14 }
```

facebook

Simple way to use it: Drawee

```
<com.facebook.drawee.view.SimpleDraweeView
    android:id="@+id/my_image_view"
    android:layout_width="20dp"
    android:layout_height="wrap_content"
    fresco:viewAspectRatio="1.33"
    <!-- other attributes -->
```

```
<com.facebook.drawee.view.SimpleDraweeView
    android:id="@+id/my_image_view"
    android:layout_width="20dp"
    android:layout_height="20dp"
    fresco:fadeDuration="300"
    fresco:actualImageScaleType="focusCrop"
    fresco:placeholderImage="@color/wait_color"
    fresco:placeholderImageScaleType="fitCenter"
    fresco:failureImage="@drawable/error"
    fresco:failureImageScaleType="centerInside"
    fresco:retryImage="@drawable/retrying"
    fresco:retryImageScaleType="centerCrop"
    fresco:progressBarImage="@drawable/progress_bar"
    fresco:progressBarImageScaleType="centerInside"
    fresco:progressBarAutoRotateInterval="1000"
    fresco:backgroundImage="@color/blue"
    fresco:overlayImage="@drawable/watermark"
    fresco:pressedStateOverlayImage="@color/red"
    fresco:roundAsCircle="false"
    fresco:roundedCornerRadius="1dp"
    fresco:roundTopLeft="true"
    fresco:roundTopRight="false"
    fresco:roundBottomLeft="false"
    fresco:roundBottomRight="true"
    fresco:roundWithOverlayColor="@color/corner_color"
    fresco:roundingBorderWidth="2dp"
    fresco:roundingBorderColor="@color/border_color"
/>
```


facebook

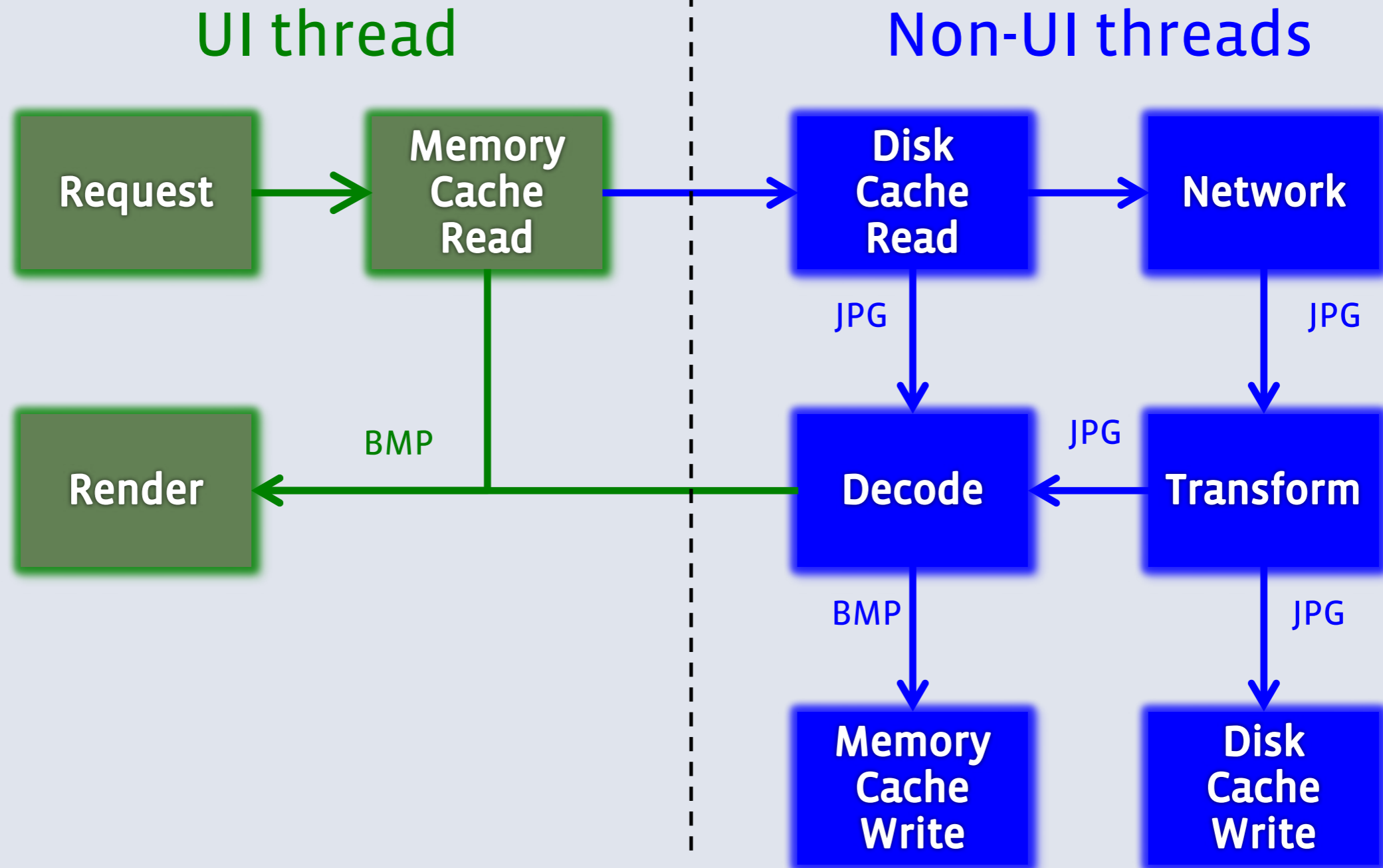
How to use it:

```
// Build image request
ImageRequest imageRequest = ImageRequest.fromUri(myUri);

// To prefetch to disk cache
ImagePipeline imagePipeline = null;
imagePipeline.prefetchToDiskCache(imageRequest, callerContext);

// To prefetch to bitmap cache
DataSource<CloseableReference<CloseableImage>> dataSource =
    imagePipeline.fetchDecodedImage(imageRequest, callerContext);
try {
    CloseableReference<CloseableImage> imageReference = dataSource.getResult();
    if (imageReference != null) {
        try {
            CloseableImage image = imageReference.get();
            // Do something with the image
        } finally {
            imageReference.close();
        }
    }
} finally {
    dataSource.close();
}
```

The Image Pipeline



facebook

Memory is important factor

- $1/10^{\text{th}}$ for each application
- Images use lots of memory
 - 400x800 pixels = 1536bytes = 1.5MB

facebook

Java has the Garbage Collector

- It removes all the unused object
- But there are drawbacks....

facebook

Unfortunately, Facebook has
stopped.

Report

OK

`java.lang.OutOfMemoryError`

facebook

Possible solutions to avoid OOM

- Use smaller images
- Use lower resolution

Less than ideal!!!!

facebook

Different types of Heap

Dalvik Heap

- JDK
- Limited
- Slow
- Safe

Native Heap

- NDK
- Unlimited
- Fast
- Unsafe

Other Heaps

- System calls
- Same advantages of the Native Heap

facebook

Use the Native Heap

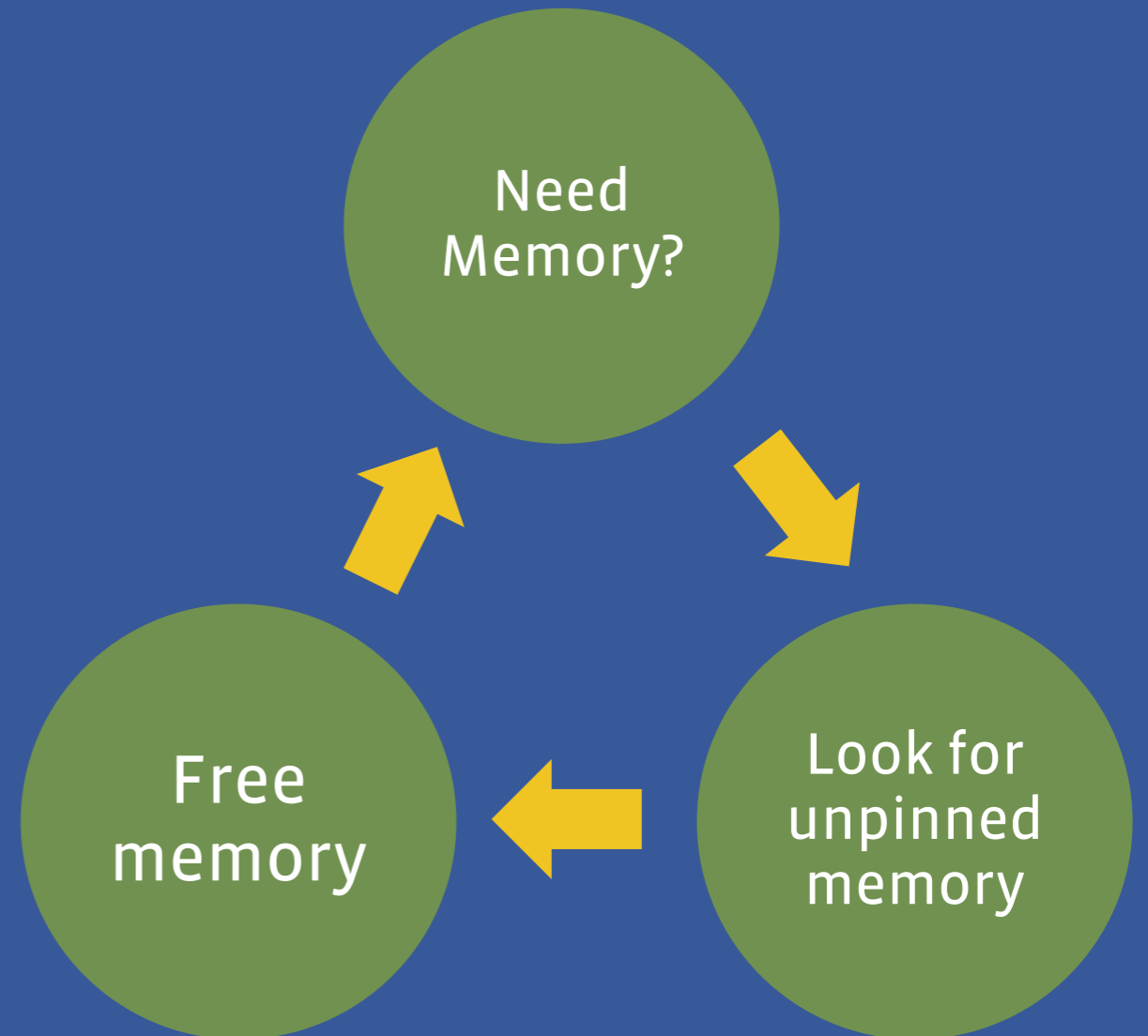
```
public final class Bitmap implements Parcelable {  
    public final long mNativeBitmap;  
    - - -  
}
```

```
jobject GraphicsJNI::createBitmap(JNIEnv* env, SkBitmap* bitmap,  
    int bitmapCreateFlags, jbyteArray ninePatchChunk, jobject ninePatchInsets,  
    int density) {  
    bool isMutable = bitmapCreateFlags & kBitmapCreateFlag_Mutable;  
    bool isPremultiplied = bitmapCreateFlags & kBitmapCreateFlag_Premultiplied;  
    // The caller needs to have already set the alpha type properly, so the  
    // native SkBitmap stays in sync with the Java Bitmap.  
    assert_premultiplied(bitmap->info(), isPremultiplied);  
    jobject obj = env->NewObject(gBitmap_class, gBitmap_constructorMethodID,  
        reinterpret_cast<jlong>(bitmap), bitmap->javaByteArray(),  
        bitmap->width(), bitmap->height(), density, isMutable, isPremultiplied,  
        ninePatchChunk, ninePatchInsets);  
    hasException(env);  
    return obj;  
}
```

facebook

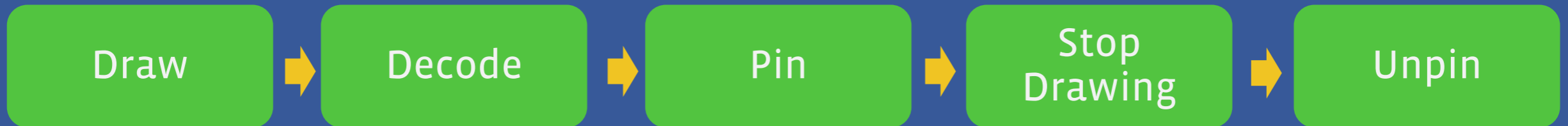
Use the Others Heap: **ashmem**

- Create region
- Pin region
- Unpin region
- The system frees the memory just if needed



facebook

How to use ashmem? **Purgeable** Bitmap



facebook

Purgeable Bitmap in code

```
BitmapFactory.Options options = new BitmapFactory.Options();  
options.inPurgeable = true;  
Bitmap bitmap = BitmapFactory.decodeByteArray(jpegData, 0, jpegData.length, options);
```

What is wrong with this?

Decoding is done on the UI Thread!!!

facebook

What's wrong with **Purgeable** Bitmap

While `inPurgeable` can help avoid big Dalvik heap allocations (**from API level 11 onward**), it sacrifices performance predictability since any image that the view system tries to draw may incur a decode delay which can lead to dropped frames. Therefore, most apps should avoid using **`inPurgeable`** to allow for a **fast and fluid UI**. To minimize Dalvik heap allocations use the **`inBitmap`** flag instead.

facebook

So what's the **inBitmap** flag?

```
Bitmap bitmap = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888);
BitmapFactory.Options options = new BitmapFactory.Options();
while (true) {
    // Some code
    options.inBitmap = bitmap;
    Bitmap bitmap = BitmapFactory.decodeByteArray(jpegData, 0, jpegData.length, options)
    // Use the bitmap - - -
}
```

- It looks perfect!! But....

facebook

In FB we cannot use **inBitmap**

- Introduced in API Level 11
- Prior to KitKat only JPEG and PNG were supported with the same size
- We need a different solution

facebook

The Facebook Solution

```
int AndroidBitmap_lockPixels(JNIEnv* env, jobject jbitmap, void** addrPtr) {
    if (NULL == env || NULL == jbitmap) {
        return ANDROID_BITMAP_RESULT_BAD_PARAMETER;
    }

    SkBitmap* bm = GraphicsJNI::getNativeBitmap(env, jbitmap);
    if (NULL == bm) {
        return ANDROID_BITMAP_RESULT_JNI_EXCEPTION;
    }

    bm->lockPixels();
    void* addr = bm->getPixels();
    if (NULL == addr) {
        bm->unlockPixels();
        return ANDROID_BITMAP_RESULT_ALLOCATION_FAILED;
    }

    if (addrPtr) {
        *addrPtr = addr;
    }
    return ANDROID_BITMAP_RESULT_SUCCESS;
}
```

facebook

The Facebook Solution: Pin but not Unpin

```
@DoNotStrip  
private static native void nativePinBitmap(Bitmap bitmap);
```

```
static void Bitmaps_pinBitmap(  
    JNIEnv* env,  
    jclass clazz,  
    jobject bitmap) {  
    UNUSED(clazz);  
    int rc = AndroidBitmap_lockPixels(env, bitmap, 0);  
    if (rc != ANDROID_BITMAP_RESULT_SUCCESS) {  
        safe_throw_exception(env, "Failed to pin Bitmap");  
    }  
}
```


facebook

It's important to **recycle()**

void **recycle()**

Free the native object associated with this bitmap, and clear the reference to the pixel data.

```
BitmapFactory.Options options = new BitmapFactory.Options();
options.inPurgeable = true;
Bitmap bitmap = BitmapFactory.decodeByteArray(jpegData, 0, jpegData.length, options);
lockBitmap(bitmap);
// When done
bitmap.recycle();
```


facebook

Great **Power** means Great **Responsibility**

SharedReferences

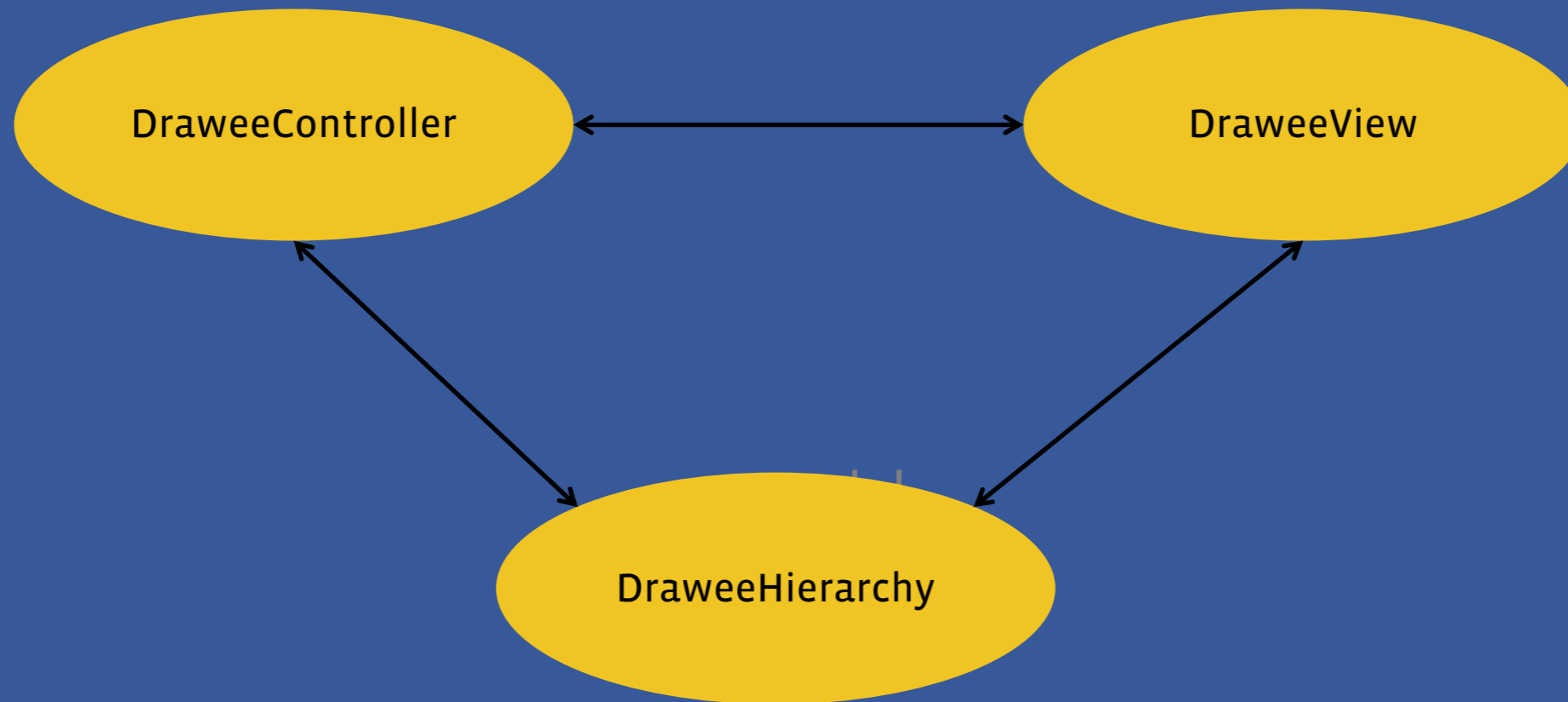
<<interface>>
CloseableReference

- Deterministic memory management
- Simple and clear rules

facebook

Drawee makes things simple

- MVC like framework to display images



facebook

The future of Fresco

- Smaller libraries and less .SO
 - WebP
 - Animations (GIF and WebP)
- Making Image Pipeline pluggable
- Many other optimizations and extensions

1. <http://frescolib.org>
Blog
<http://frescolib.org/blog/2016/09/02/fresco-101.html>
2. <http://fresco-cn.org>
3. Facebook: <https://www.facebook.com/frescolib/>
4. Twitter: [@frescolib](https://twitter.com/frescolib)

facebook

(c) 2009 Facebook, Inc. or its licensors. "Facebook" is a registered trademark of Facebook, Inc.. All rights reserved. 1.0