Androidにおける
リアルタイムカメラエフェクト実装の
手法について

sbc シャープビジネスコンピュータソフトウェア株式会社

2010年8月28日

NBS開発統轄部NB開発センター

鵜川　裕文

TwitterID awaku7

Agenda

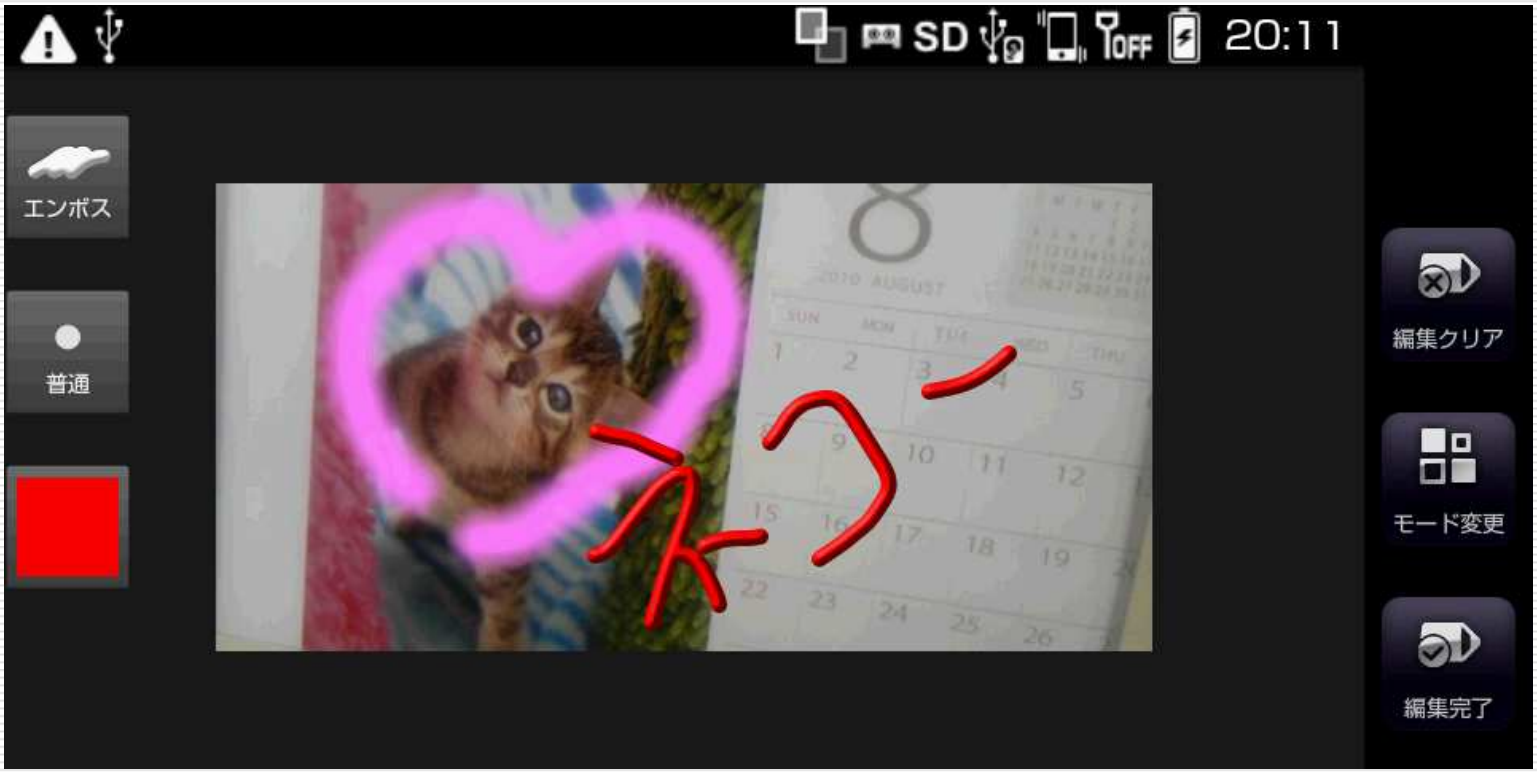
- 少し弊社のご紹介（口頭で）
- カメラアプリの仕組み
- リアルタイムエフェクト？
- リアルタイムエフェクトカメラの作り方

カメラは楽しい

- ほとんどの携帯電話にカメラが付いており
  多くの人が利用している
- 携帯電話で画像編集が出来るようになった
- 撮るだけでなく、撮った後の楽しみが増えた

カメラは楽しい！

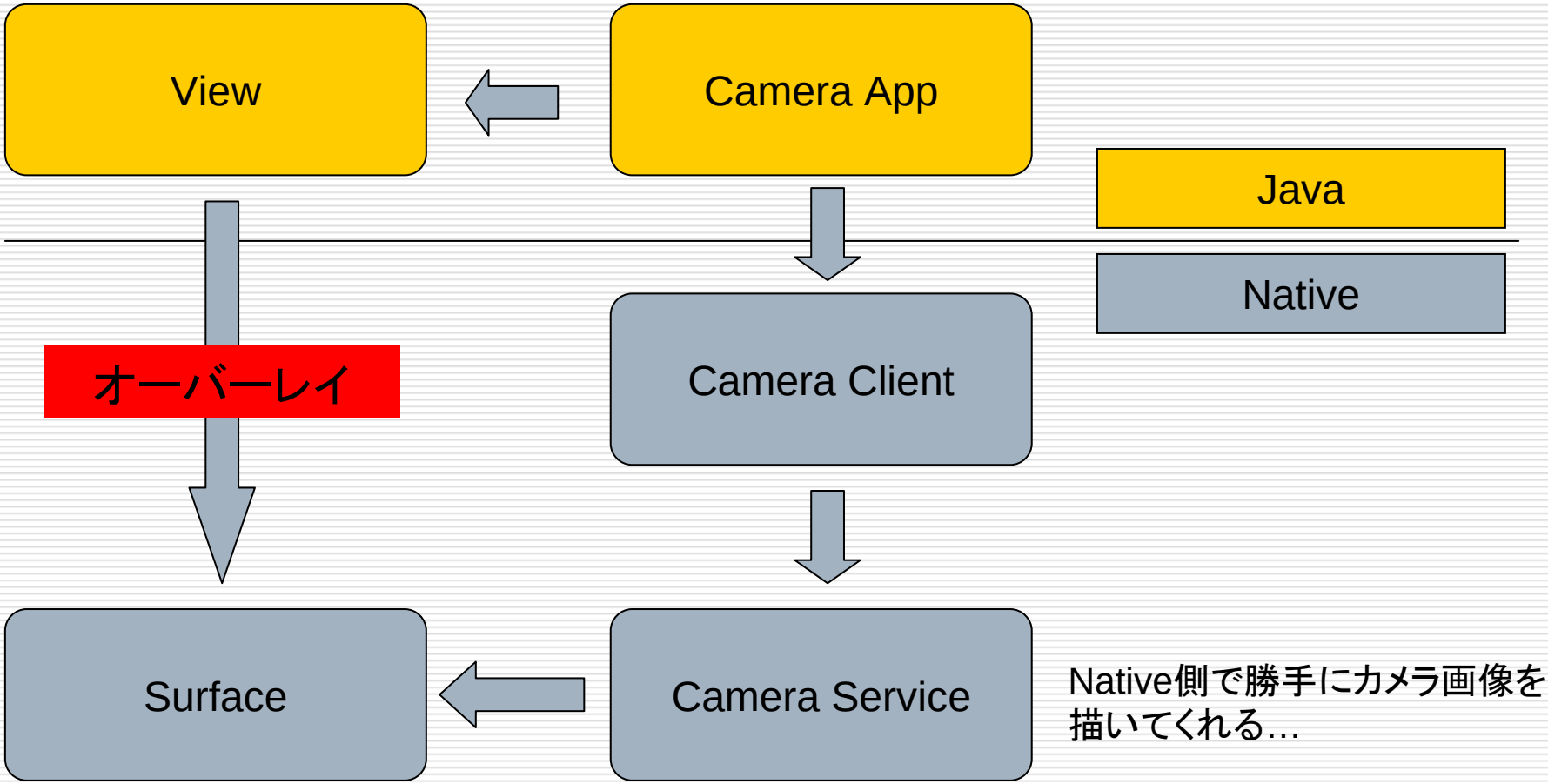
- IS01のカメラアプリの手書き加工画面



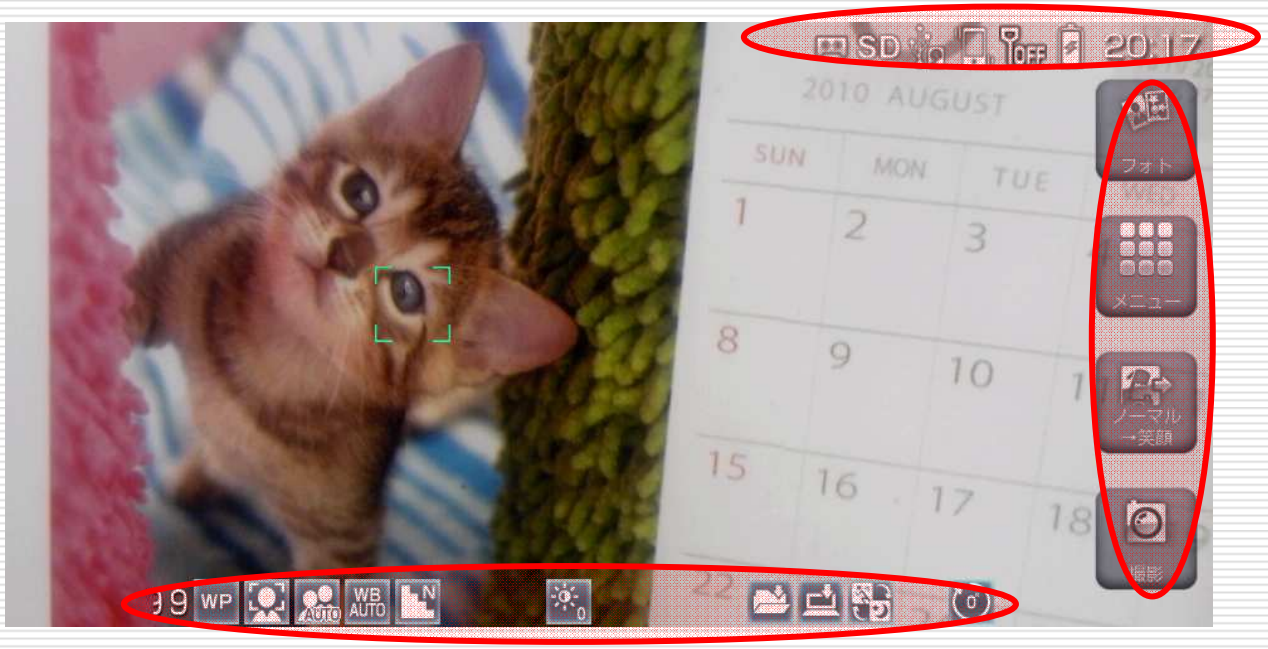
カメラは楽しい！

- IS01のエフェクト加工画面

一般的なカメラアプリの仕組み



カメラにおけるオーバーレイ(1)



例）IS01 カメラアプリ

ボタンや透過したステータスバーの下側にカメラプレビューがオーバレイされている。

カメラにおけるオーバーレイ(2)

例）UkiUkiView

ボタンやアイコンの下側にSurfaceViewのカメラプレビューがオーバーレイされている。

カメラにおけるオーバーレイ(3)

- レイアウトの例

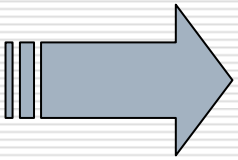
RelativeLayoutで構成（実際はマージンなどで座標指定,LinearLayoutの埋め込みなどを行う）

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android=http://schemas.android.com/apk/res/android
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"          >

    <SurfaceView />    <!-- 一番下にカメラのプレビュー -->
    <ImageButton />    <!-- プレビューの上にビューを乗せる -->
    <ImageButton />

</RelativeLayout>
```

ここまでのまとめ

- カメラで映像を撮った後に画像編集する
  →手書き、エフェクト

- カメラビューの上にアイコンなどを乗せる
  →カメラ、UkiUkiView



オーバーレイ以外の使い方はないか？

リアルタイムエフェクト？

リアルタイムエフェクトを実装する事で、撮影時にリアルタイムで
「撮影後の静止画イメージ」を参照する事が出来る様になります。

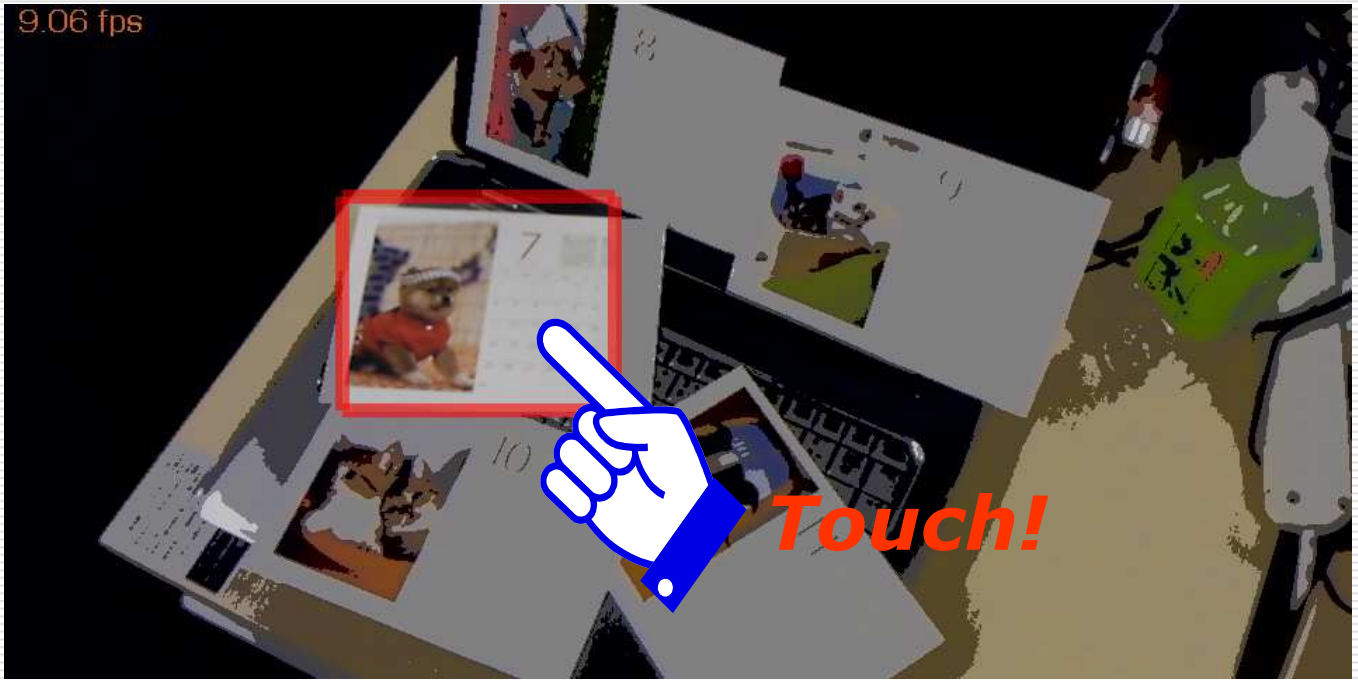
デモンストレーション

リアルタイムエフェクト？

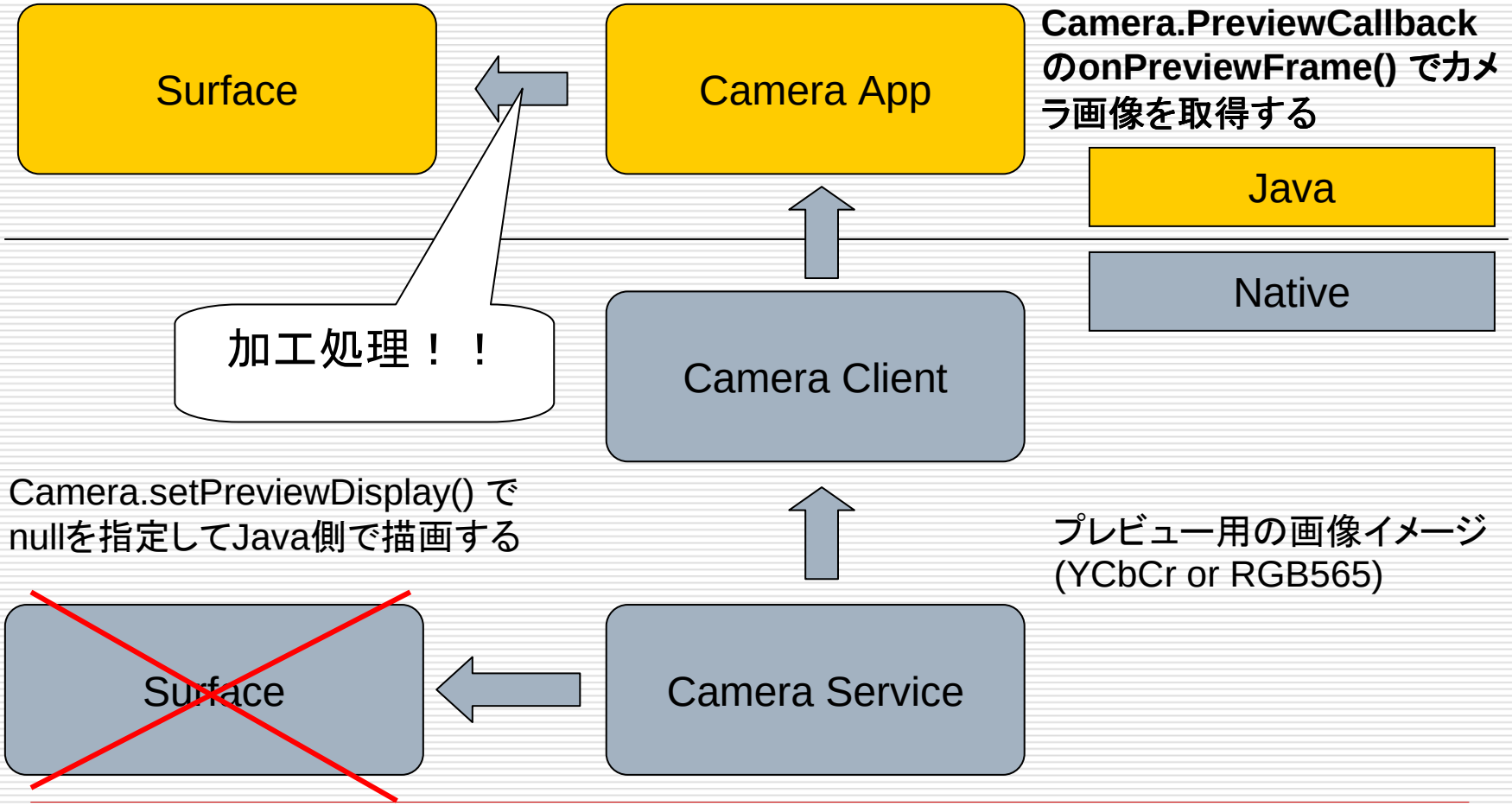
リアルタイムエフェクトを実装する事で、撮影時にリアルタイムで
「撮影後の静止画イメージ」を参照する事が出来る様になります。

リアルタイムエフェクト？

リアルタイムエフェクトを実装する事で、撮影時にリアルタイムで
「撮影後の静止画イメージ」を参照する事が出来る様になります。

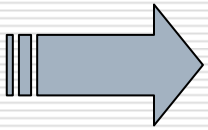


今回のリアルタイムエフェクトの仕組み



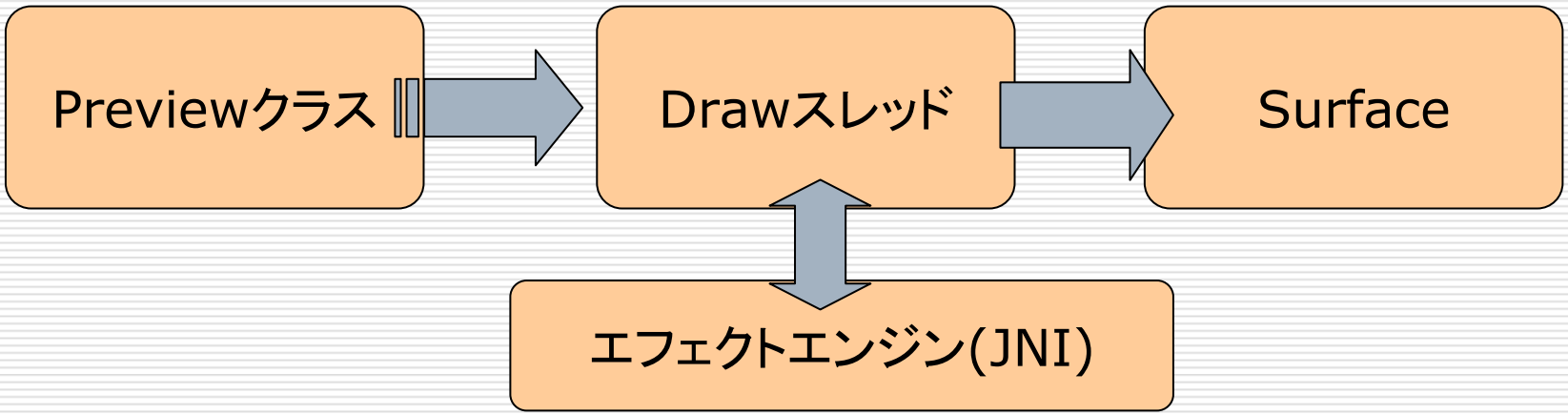
難しいパフォーマンスの問題

- Java実装のみで Surface に書き込むために YUV to RGB の変換を行うと 0.5 fps 程度の性能しか得られない。
- カメラからのプレビュー用のフレームはおよそ100msecの間隔で受け取ることが出来るが、処理にそれ以上の時間を要するとFPSが極端に落ち込んで見えてしまう。



JNIを使いエフェクト処理を行う必要がある

実装の仕方



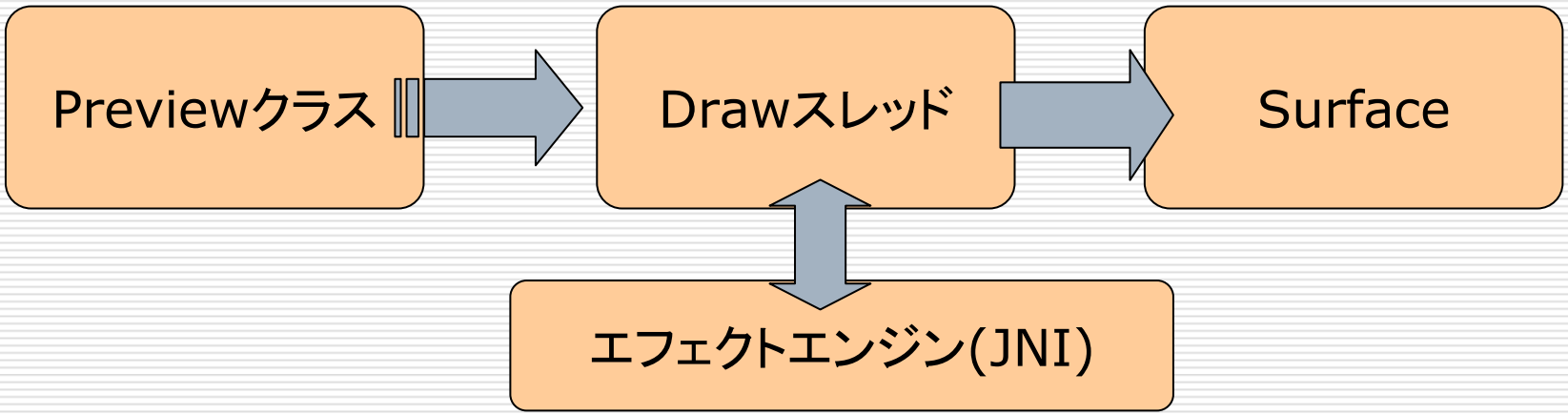
- Previewクラス

  →SurfaceHolder.Callback と Camera.PreviewCallback の実装

- Drawスレッド

  →画像データにエフェクト処理を加えてSurfaceに描きこむ、という作業を繰り返す

実装の仕方

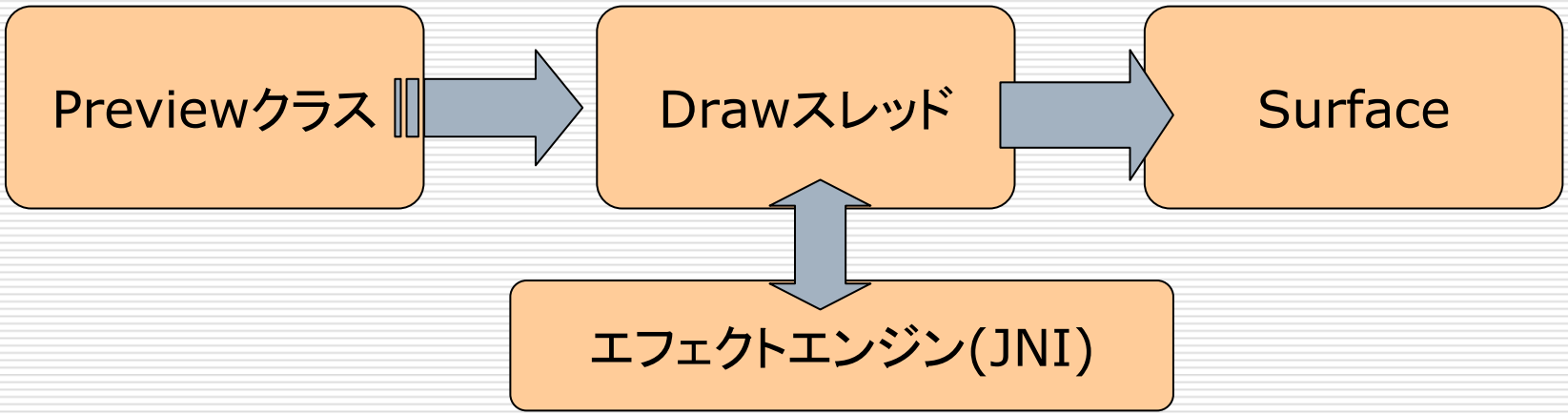


- フレームのフォーマット

  →YCbCr(NV21)、YCbCr(NV16)、RGB565のいずれか

  Camera.Parameters.getPreviewFormat() で確認すること

実装の仕方

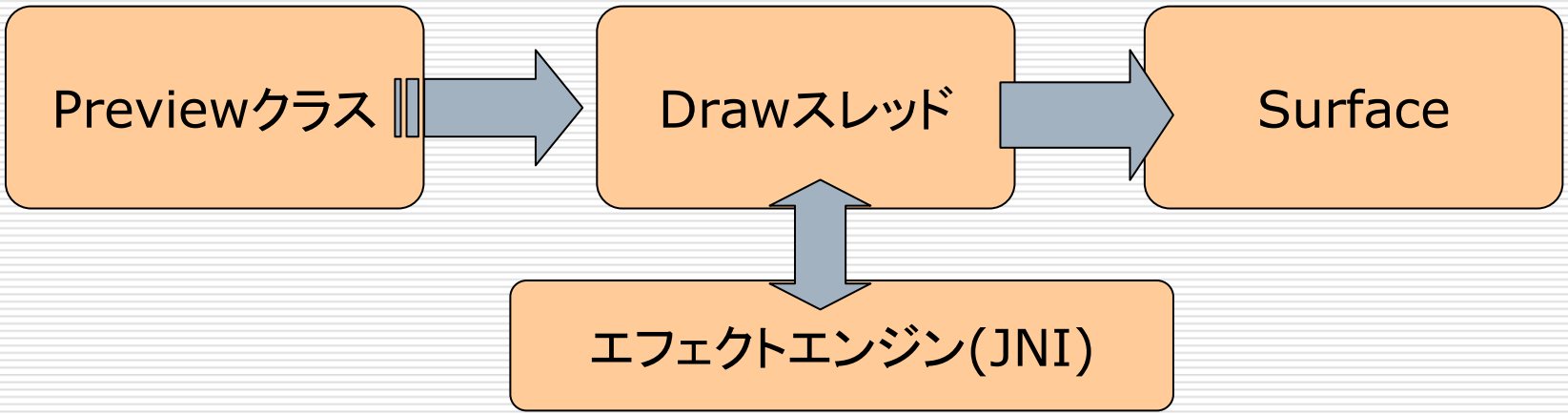


- Surfaceへの描画

  →***RGB565***のBitmapオブジェクトを作成し、Canvas.drawBitmap()で描画する

  つまり、フレームのフォーマットを自前でRGB565へ変換する必要がある

実装の仕方



★ポイント★

- onPreviewFrame(Previewクラス)で受け取ったフレームは受け取ったことをDrawスレッドに通知して直ちに処理を終えること

Tips

- setPreviewCallback()はよろしくない
  ＞エフェクト処理中にもコールバックが動く
- setOneShotPreviewCallback()を使う
  ＞自身のタイミングでコールバックが動く
- コールバックを取り逃がすと次のフレームが来るのは約100msec後
- 4fpsの時のCPU使用率は60%程度
  ＞抜けフレームの処理、工夫の余地あり
- JNIでJavaからGetしたフレームデータは必ずJNI側でRelease！

Tips more…

- NV21 = YUV420 Semi-Planer
- NV16 = YUV422 Semi-Planer
- デフォルトはNV21
- FroyoのエミュレータはNV16

勉強してください！！

ご清聴ありがとうございました