第３部　Tips

Tips - #01 適切な画像リソースの作り方(Unity編)
Tips - #02 適切な画像リソースの作り方(Cocos2d-x編)
Tips - #03 2DゲームでPVRTC,ETCを使う場合の注意点
Tips - #04　.pvrファイルをSpriteとして使う方法(Unity)
Tips - #05　どの範囲の機種をサポートするべきか
Tips - #06　Androidの画面解像度
Tips - #07 カタログスペックと実測値の乖離
Tips - #08 プロファイラ

Tips - #01 適切な画像リソースの作り方(Unity編)

Unity で、描画性能や画質を向上させるためのノウハウ

Tips - #01 適切な画像リソースの作り方(Unity編)

Compressed(デフォルト)

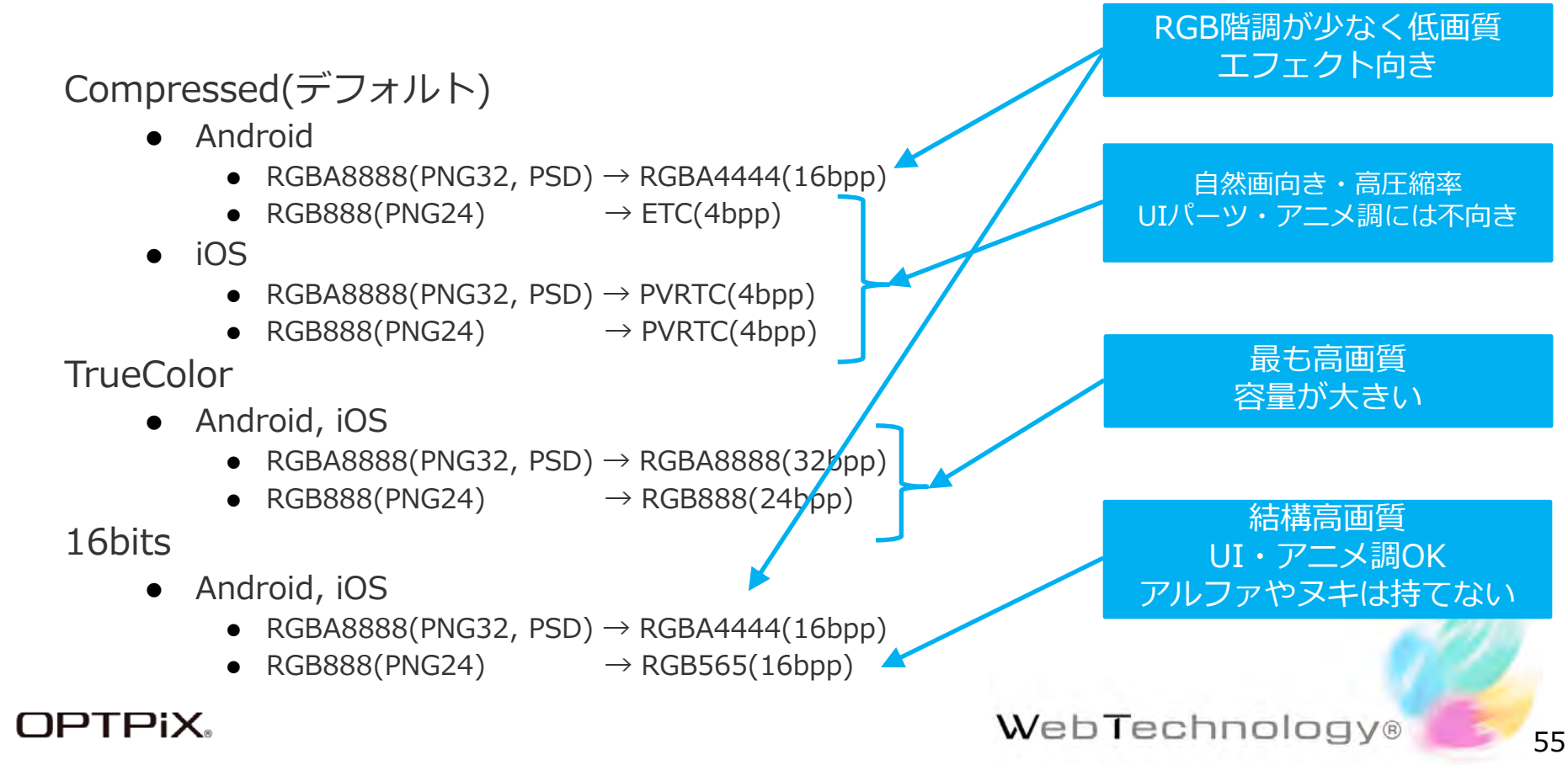
- Android
  - RGBA8888(PNG32, PSD) → RGBA4444(16bpp)
  - RGB888(PNG24) → ETC(4bpp)
- iOS
  - RGBA8888(PNG32, PSD) → PVRTC(4bpp)
  - RGB888(PNG24) → PVRTC(4bpp)

TrueColor

- Android, iOS
  - RGBA8888(PNG32, PSD) → RGBA8888(32bpp)
  - RGB888(PNG24) → RGB888(24bpp)

16bits

- Android, iOS
  - RGBA8888(PNG32, PSD) → RGBA4444(16bpp)
  - RGB888(PNG24) → RGB565(16bpp)



.pvrコンテナにダイレクトカラーを格納して使う

ダイレクトカラーが格納可能

- RGB565
- RGBA5551
- RGBA4444

結構高画質
UI・アニメ調OK
アルファやヌキは持てない

そこそこ高画質
UI・アニメ調OK
ヌキが持てる

RGB階調が少なく低画質
エフェクト向き

- PVRTCやETCにはかなわないが、そこそこ高速
- 減色ツールを選べば、画質劣化も最小限
- Android, iOSで共通に使える
- .pvrコンテナ自体は非圧縮だが、.apk や .ipa ファイルでパッケージ化されるときには圧縮されるので、ダウンロードのサイズはそれほど増えない

.pvrコンテナにダイレクトカラーを格納して使う

.pvrコンテナを使うもう一つのメリット

- PSDやPNGでは、ビルドターゲット(iOS/Android)を切り替えるたびに画像ファイルのリソースをすべて変換する処理が入り、ファイル数が多いと数10分に及ぶこともある。
- .pvrコンテナの画像では、この自動変換は行われないため、**ビルドが高速化する**。

その他の参考情報

- 16bppはPNGで保存できないため、.pvrコンテナを使う必要がある
- Unityは、.pvrコンテナの画像は変換しない。TrueColor, 16bitsを指定しても無視される
- .pvr コンテナへのダイレクトカラー出力は、OPTPiX imésta などが対応している
- .pvr ファイルはVer.2、垂直Flip(垂直反転)を指定しておく
- .pvr コンテナのダイレクトカラーテクスチャは、縦横ドット数は任意に指定できる
  (PVRTCテクスチャは、2の累乗平方ドットのみ格納可能)
- どうしても16bppの画質では許容できない場合だけ、PNG24のファイルを使い、TrueColorを指定する

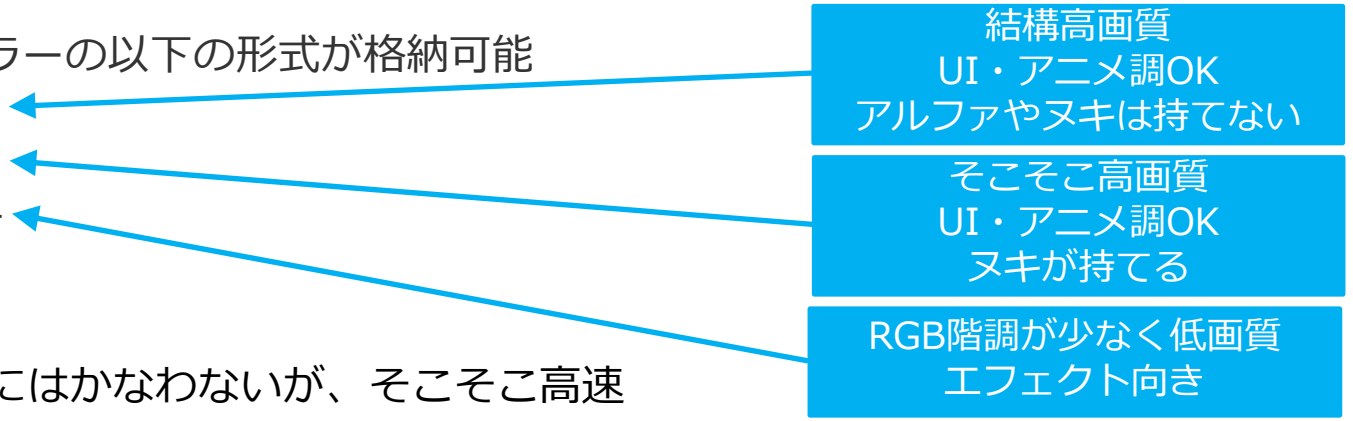
Cocos2d-x で、
描画性能や画質を向上させるためのノウハウ

- PNG32 をロード → 内部では RGBA8888
- 32bppのため、データサイズ大、パフォーマンス低

最も高画質
容量が大きい

.pvrコンテナにダイレクトカラーを格納して使う

- ダイレクトカラーの以下の形式が格納可能
  - RGB565
  - RGBA5551
  - RGBA4444



- PVRTCやETCにはかなわないが、そこそこ高速
- 減色ツールを選べば、画質劣化も最小限
- Android, iOSで共通に使える
- .pvrコンテナ自体は非圧縮だが、.apk や .ipa ファイルでパッケージ化されるときには圧縮されるので、ダウンロードのサイズはそれほど増えない

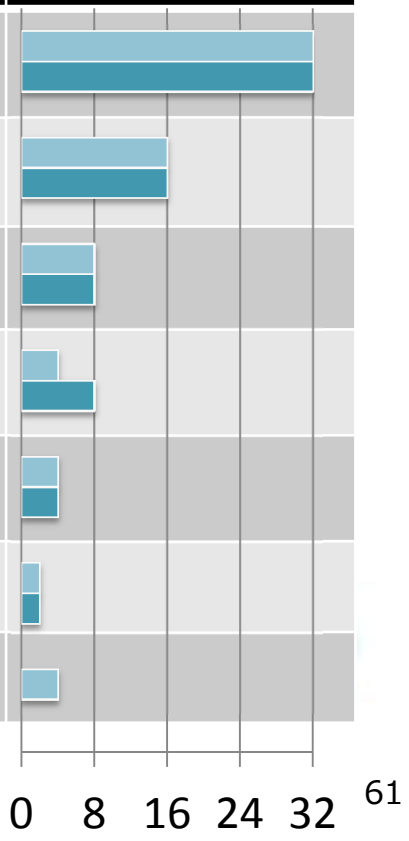
Tips - #03 2DゲームでPVRTC,ETCを使う場合の注意点

- PVRTC, ETCは、3Dテクスチャ用に開発された高圧縮フォーマットで、GPUの性能を最大に引き出すことができる
- 写真のような画像や、質感を表すようなテクスチャは、PVRTC,ETCともかなり高品質
- 動きが激しくてエッジやボヤけが気にならない場合は、PVRTC/ETC でも問題ない
- UIや、縁取りをはっきりとったアニメ調の絵では、輪郭がぼやけたり、ブロックノイズが発生する
- ETC はαが使えないので、ゲームでは使える場面が限られる
- iOSでは、PVRTCは必ず使用できるが、ETCは使用できない
- Androidでは、ETCは必ず使用できるが、PVRTCは一部の機種しか使用できない

PVRTC,ETCについて、おさらい
「圧縮テクスチャ(PVRTC・DXTC・ETC)における傾向と対策」からダイジェスト

テクスチャ形式ごとのbppと圧縮比

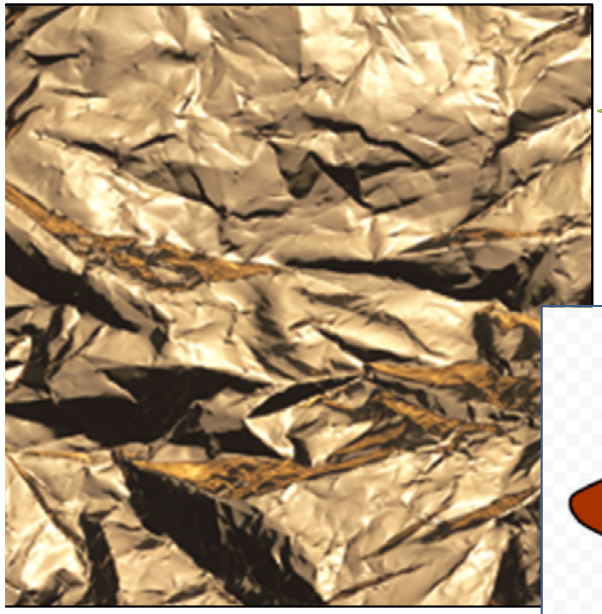
テクスチャ形式	アルファなし	アルファあり	サイズ
24bit / 32bitダイレクトカラー RGB888, RGBA8888	32bpp †	32bpp	
16bitダイレクトカラー RGB565,RGBA4444,RGB5551	16bpp	16bpp	
8bitインデックスカラー	8bpp	8bpp	
DXTC(S3TC)	4bpp	8bpp	
PVRTC 4bpp	4bpp	4bpp	
PVRTC 2bpp	2bpp	2bpp	
ETC	4bpp	使用不可	



OPTPiX

bpp：bit per pixel
† VRAM上では32bit必要

PVRTC, ETCの画質劣化の注意点



無問題

要注意

OPTPiX

PVRTC

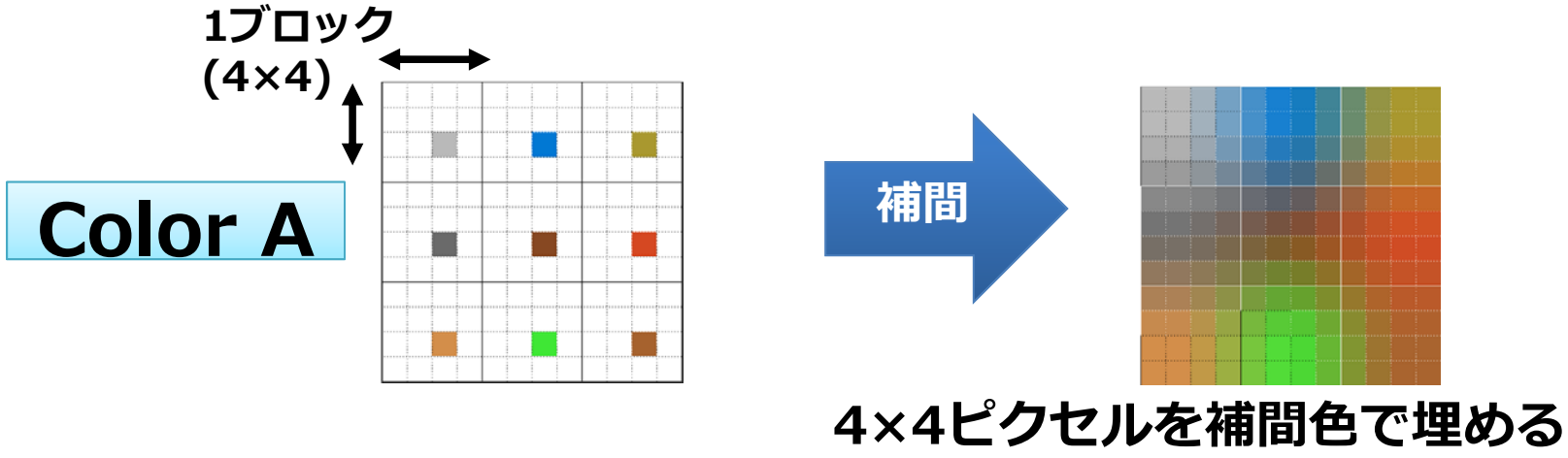
オリジナルの画像

PVRTC

PVRTC 4bpp

滲みを克服することは難しい

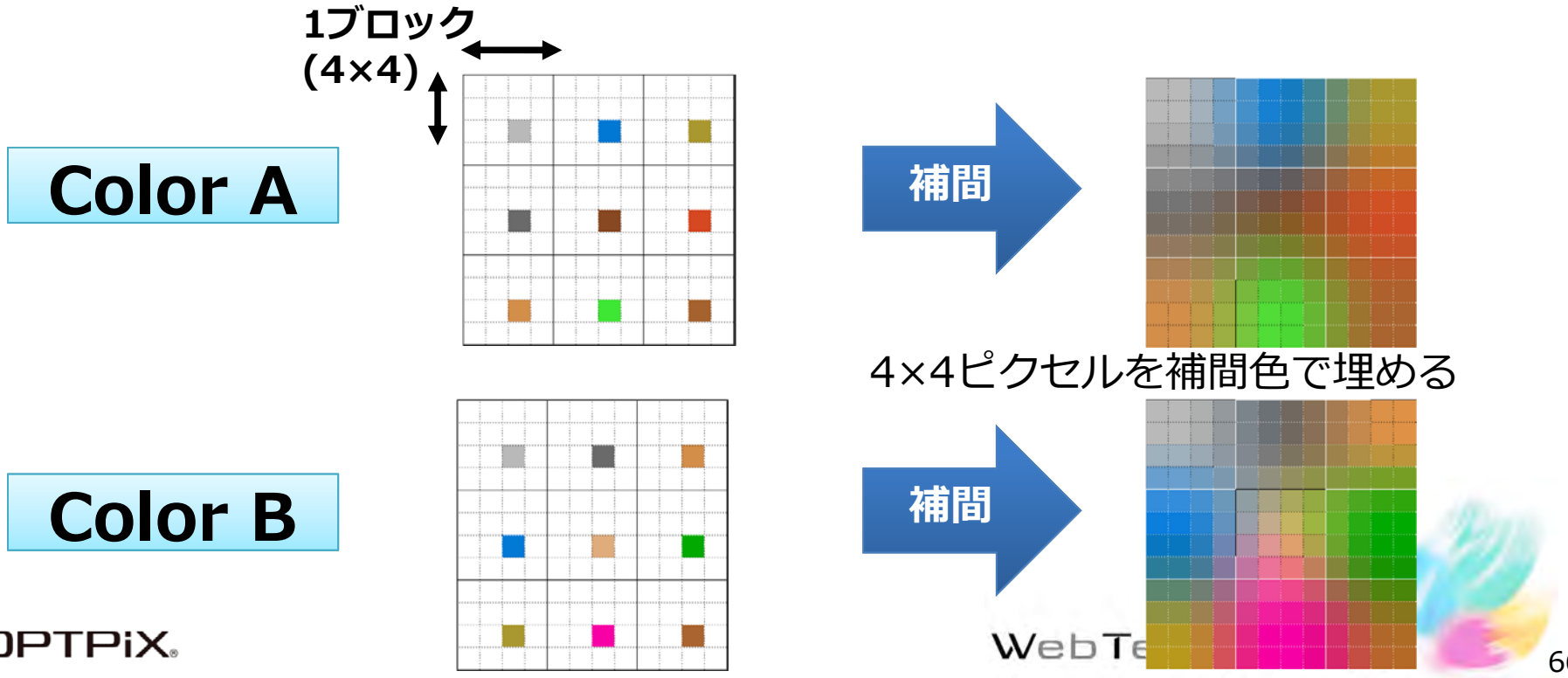
PVRTCの圧縮・伸長方法



（バイリニア）補間を行うので、
隣接ブロックの影響を受ける

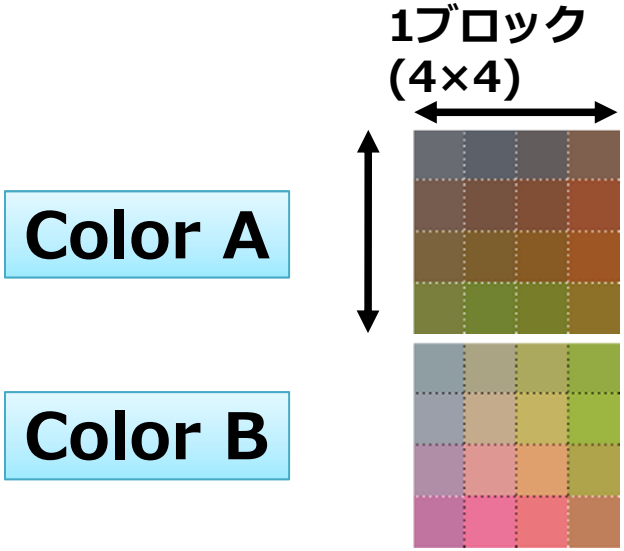
PVRTCの圧縮・伸長方法

Color A, B をそれぞれ、周囲のブロックと補間する



PVRTCの圧縮・伸長方法

補間結果のピクセルごとにブレンドする



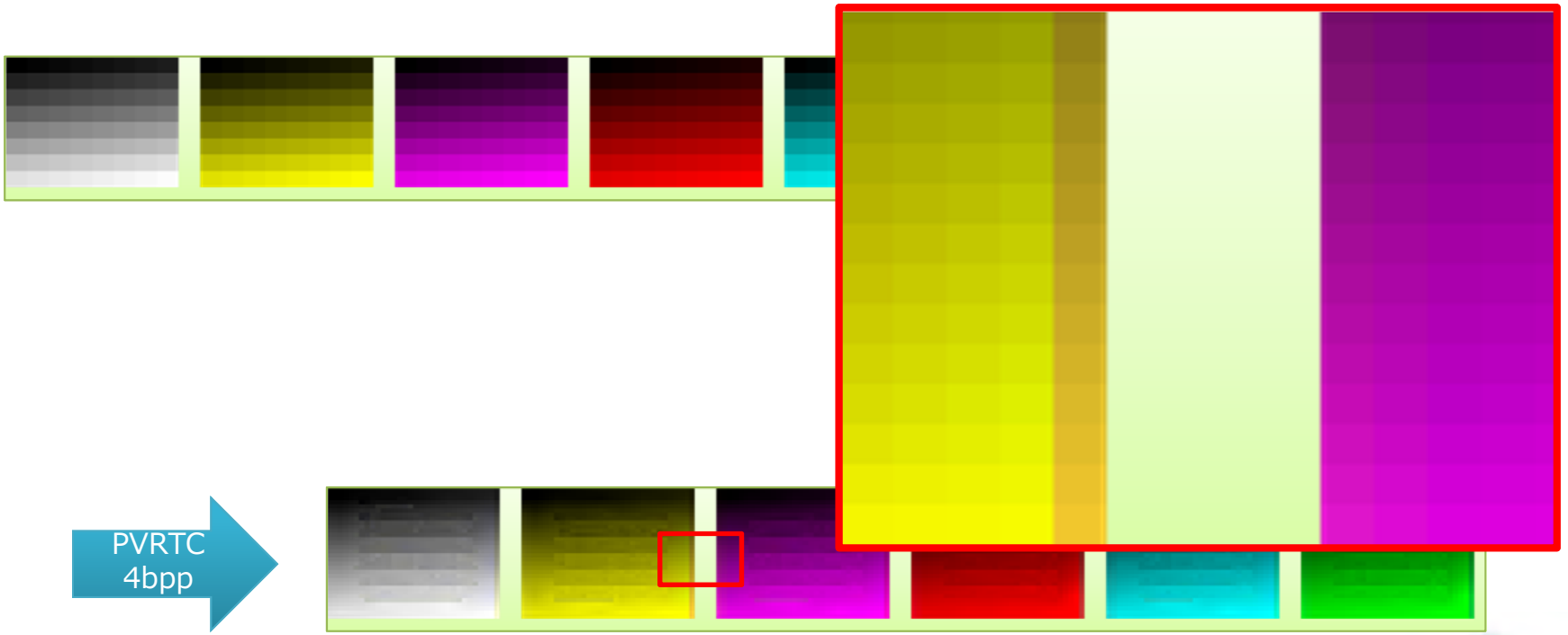
ブレンド方法

1	0	1	1
2	1	0	0
3	2	2	1
2	3	3	2

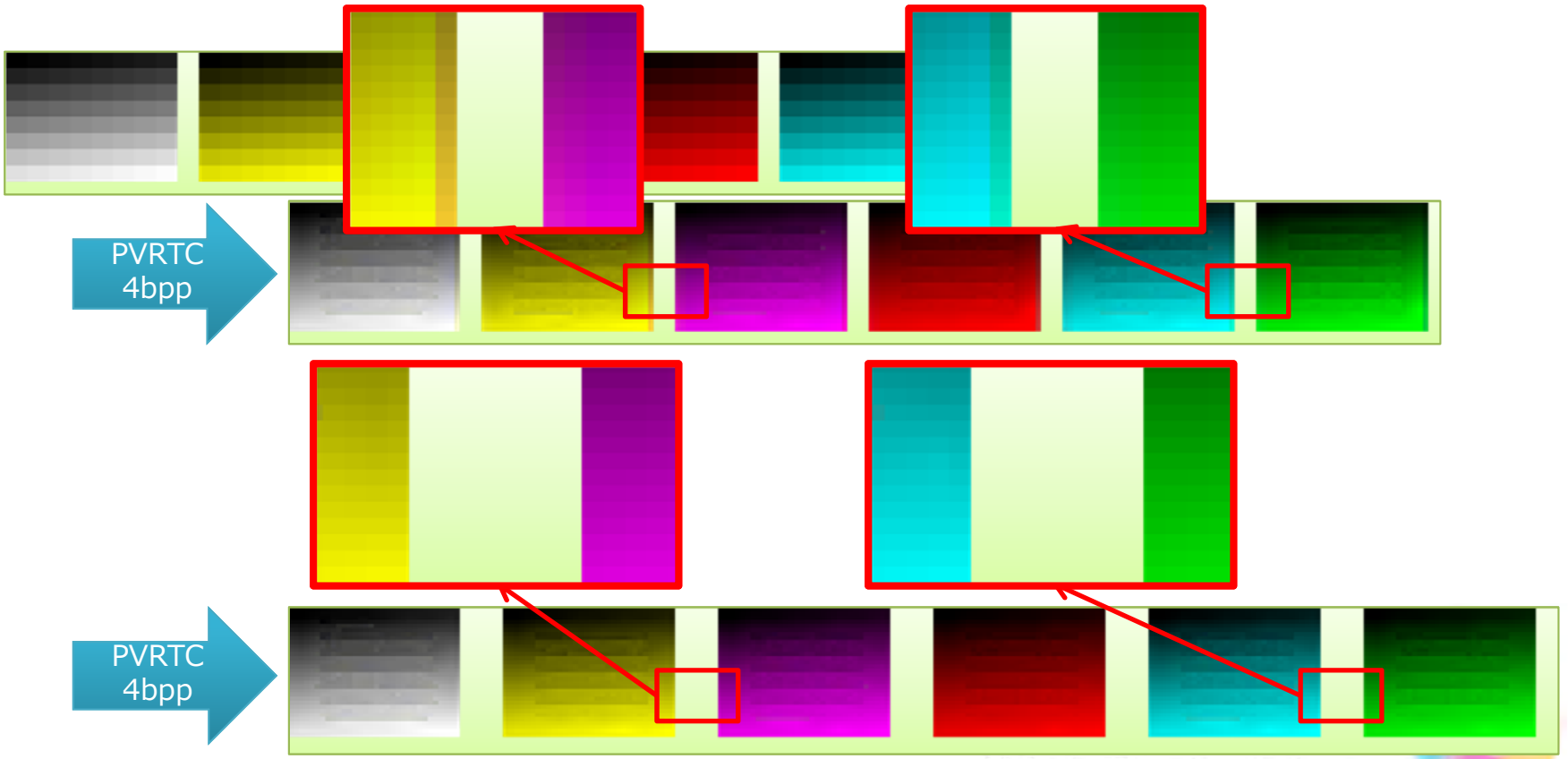
0 = A
1 = 5/8 A + 3/8 B
2 = 3/8 A + 5/8 B
3 = B

ブレンド結果を想定した Color A, B を、
圧縮時に適切に選んでおく必要がある

PVRTC 離れていても滲む

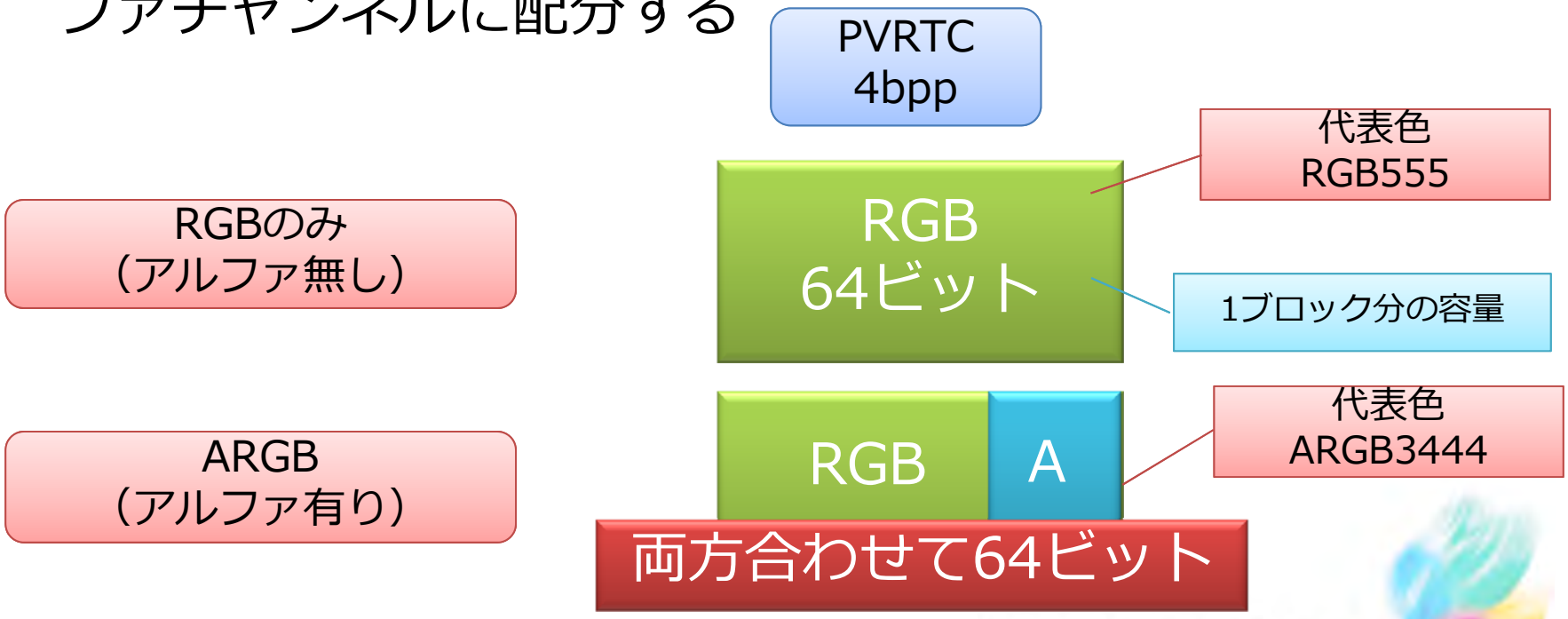


PVRTC 離れていても滲む



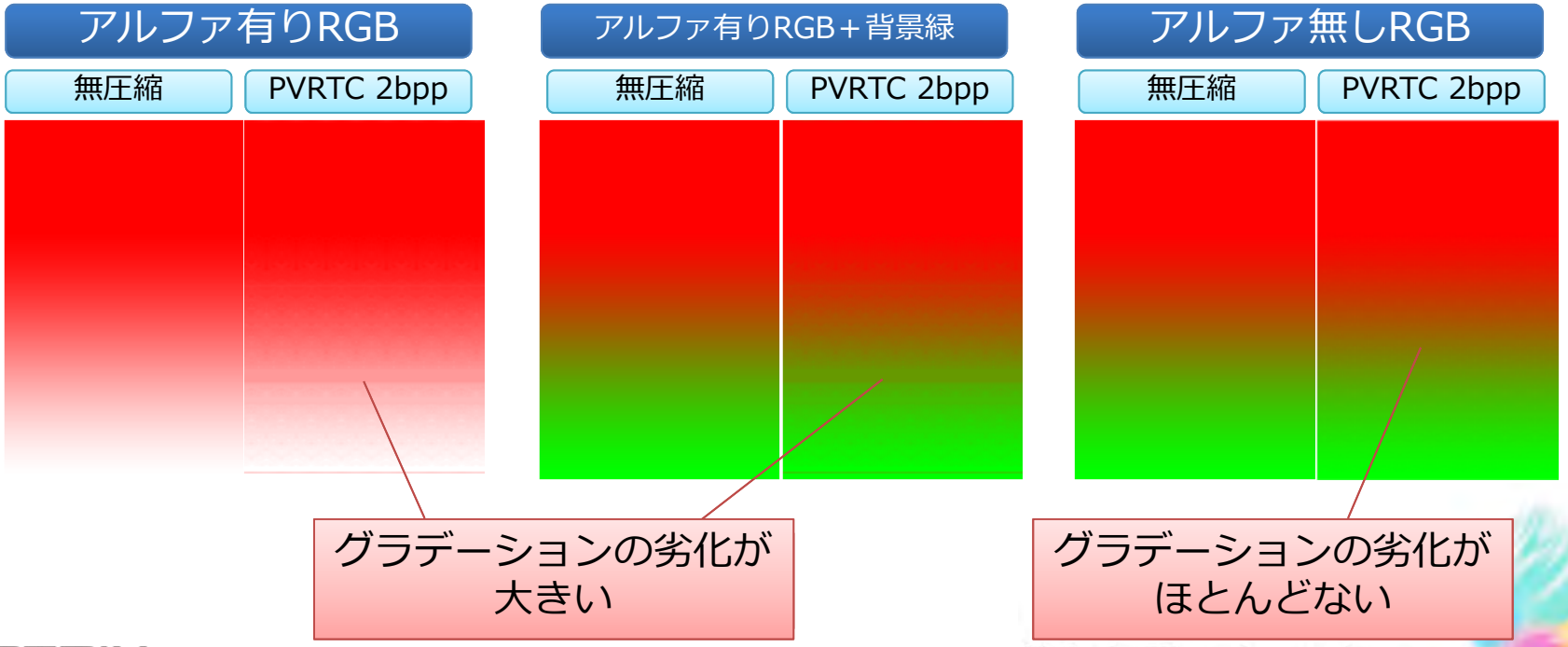
PVRTCのアルファチャンネル

- PVRTCはRGBチャンネルに使うビット数を削ってアルファチャンネルに配分する

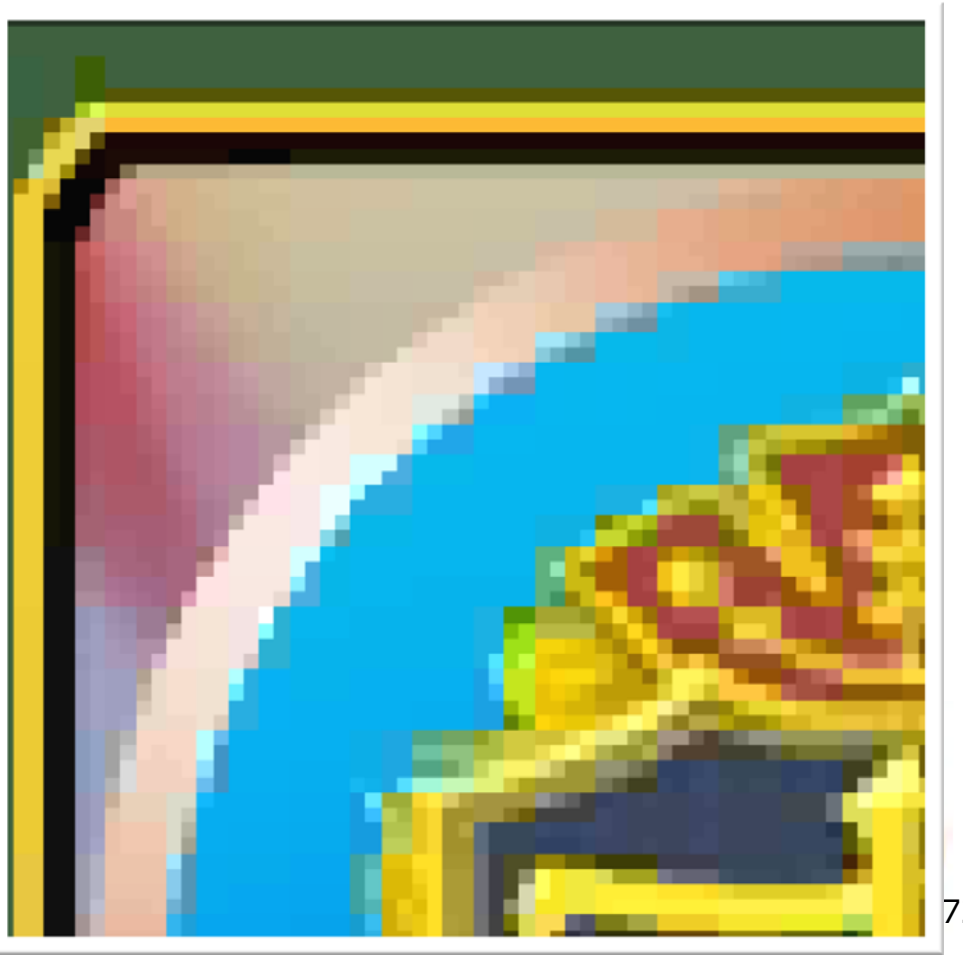


PVRTCのアルファチャンネル

- PVRTCではグラデーションをアルファで表現するよりも、RGBで表現する方が綺麗に表示できる



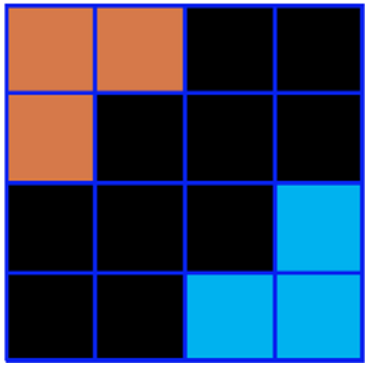
ETCのブロックノイズ



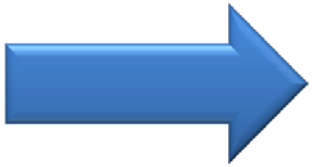
OPTPiX

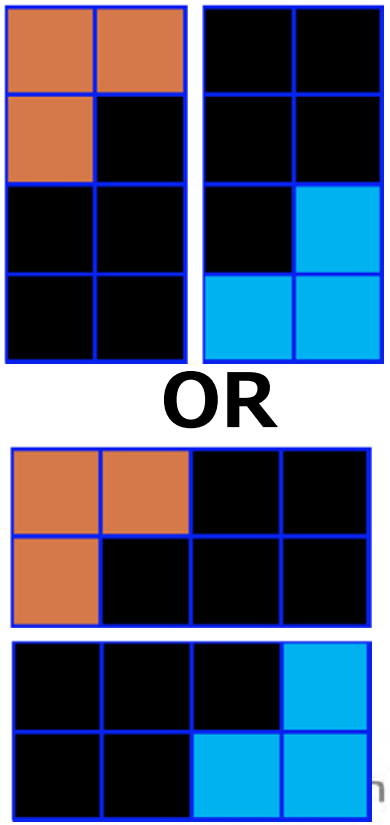
ETCの圧縮方法

- ４×４のブロックを２×４のサブブロックに分割する



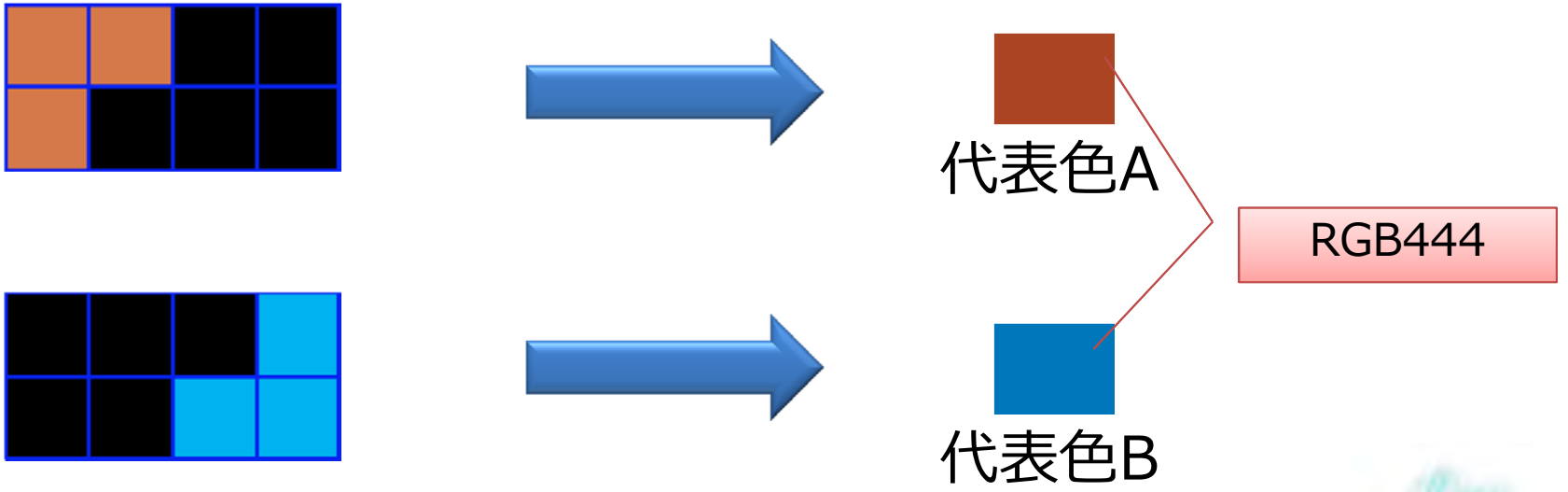
元画像





ETCの圧縮方法

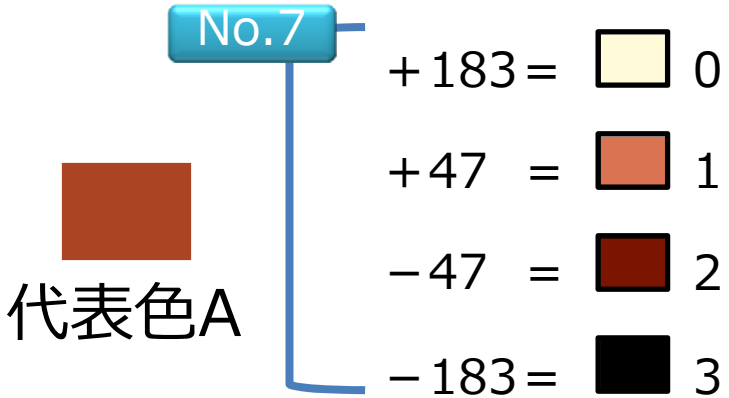
- サブブロック毎に代表色を決める



ETCの理論 (3) 代表色から4色を作る

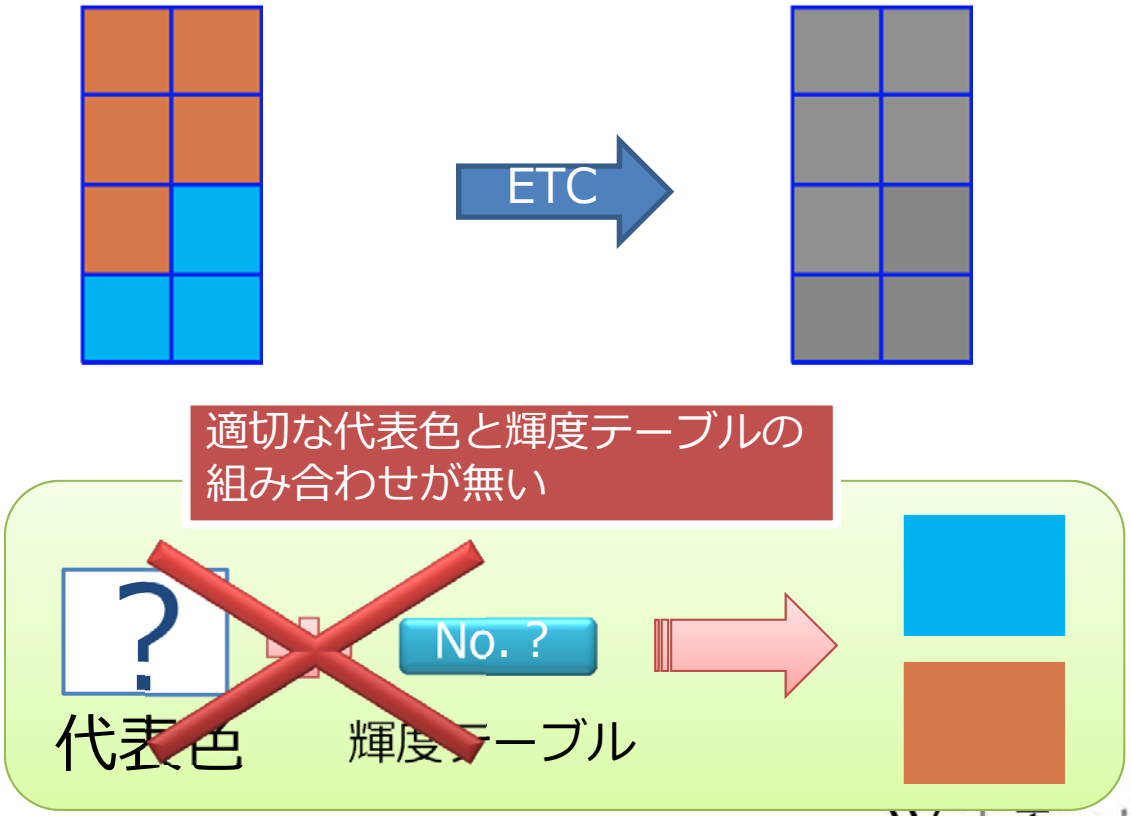
- 代表色に対する輝度の変換テーブルを選ぶ

ETCの圧縮方法



代表色から4色作る時に、輝度の変化しかできない

ETCの圧縮方法



輪郭を入れた例

黒 RGB(0,0,0)や白 RGB(255,255,255)はサブブロック中で
1色としてカウントされないのでジョーカー的に使うことができる。
そのため絵柄がOKであればノイズ低減に利用できる。

Technology®

圧縮テクスチャにおける傾向と対策 ダイジェスト

詳しくは、こちらをご覧ください
http://cedec.cesa.or.jp/2013/program/VA/6801.html

Tips - #04　.pvrファイルをSpriteとして使う方法(Unity)

- Unityエディタ上で .pvrファイルを追加すると、Spriteの作成ができない(Inspectorの設定項目が無効化されている)
- スクリプトでロードすると、Spriteのテクスチャに.pvrファイルを使用できる

```
var tex = Resources.Load("<FilePath>") as Texture2D;
var texRect = new Rect(0, 0, tex.width, tex.height);
var sprite = Sprite.Create(tex, texRect, new Vector2(0.5f, 0.5f),
100, 1, SpriteMeshType.FullRect)
```

Tips - #05　どの範囲の機種をサポートするべきか

- Cocos2d-x は、iOS 5.0以降、Android 2.3以降をサポート
- Unity は、iOS 4.0以降、Android 2.3.1以降をサポート
- サポート範囲を広げると
  - 低性能機対応のための開発工数増大
  - 低性能機にあわせたゲーム設計

- ゲームアプリでは、iOS 7以降、Android 4.0以降という割り切りも

Tips - #05　どの範囲の機種をサポートするべきか

「Unity Hardware Statistics」が役に立つ

http://stats.unity3d.com/mobile/

Tips - #05　どの範囲の機種をサポートするべきか

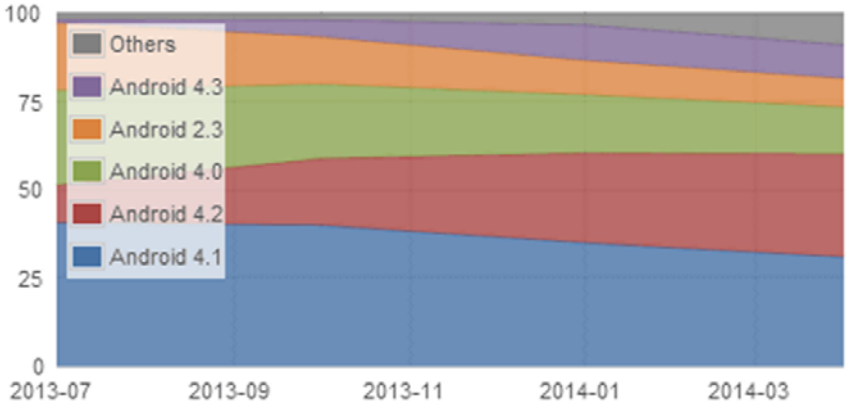
Unity　Gallery　Asset Store　Learn　Community　Company　Buy　Download

Hardware Stats　Web Player　Mobile　Standalone　Editor

Mobile (Android) Hardware Stats　2014-05

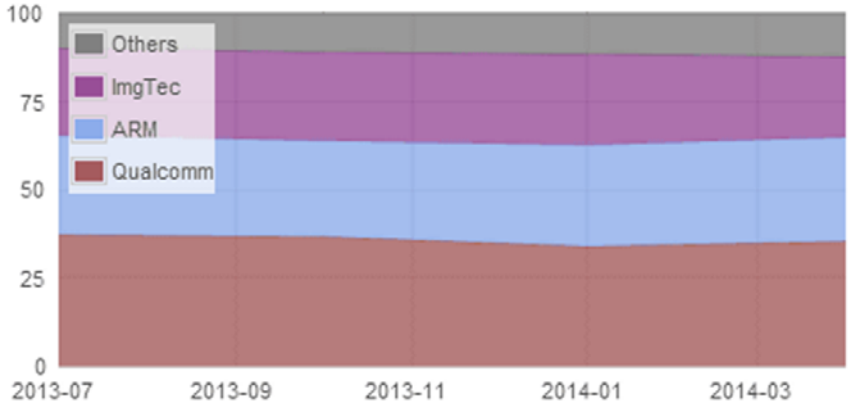
All　Android　iOS　WP8　WinRT

Operating System



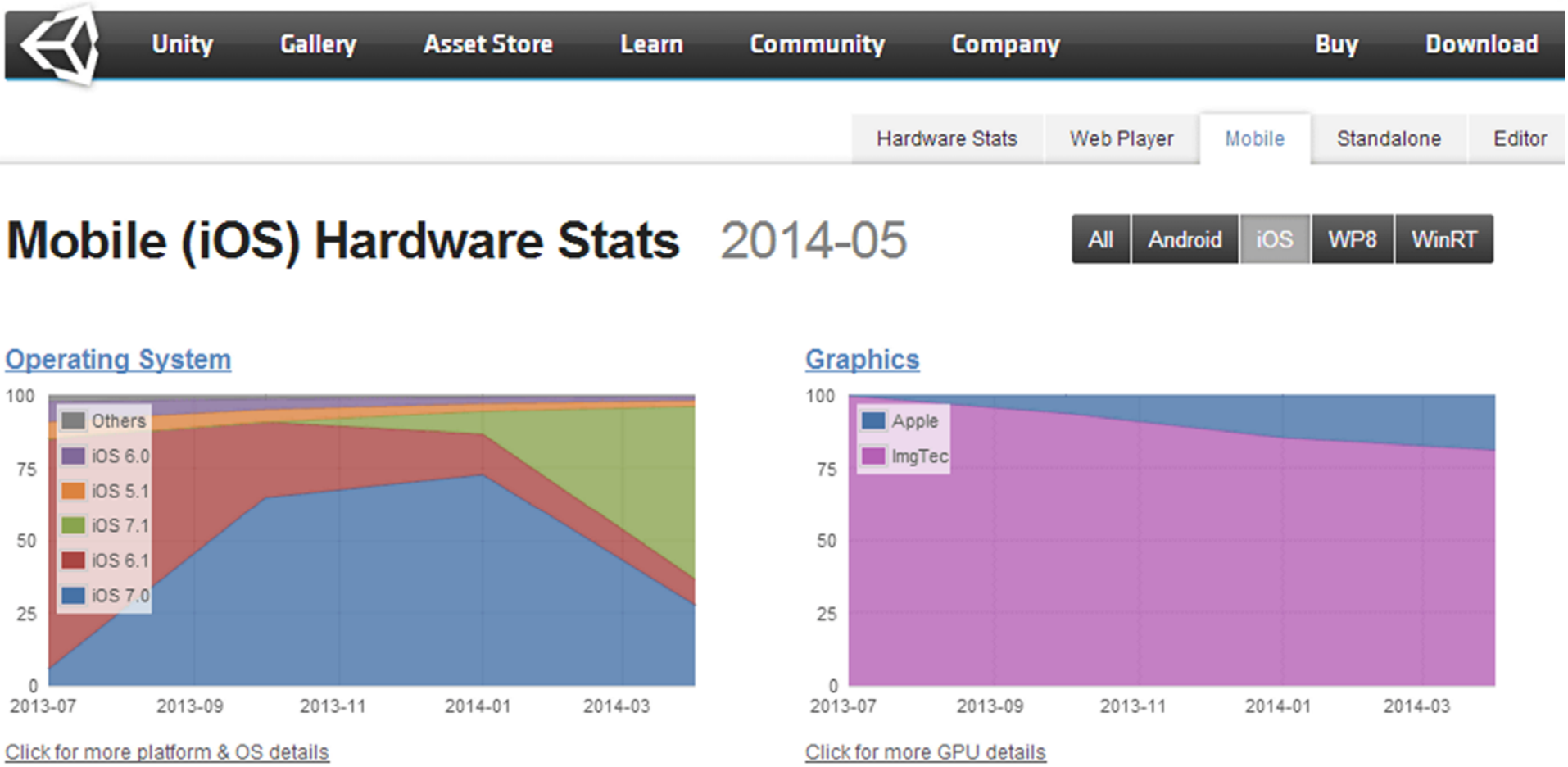
Click for more platform & OS details

Graphics



Click for more GPU details

Tips - #05　どの範囲の機種をサポートするべきか



Tips - #05　どの範囲の機種をサポートするべきか

- AppStore のiOSシェアhttps://developer.apple.com/support/appstore/ の結果とほぼ同じ
- Android (Google Play)のOSバージョンシェア
  http://developer.android.com/intl/ja/about/dashboards/ の結果でも、85%がAndroid 4以上
- 上記は全世界の情報なので、日本のユーザーの動向そのものではないことに注意
- Google Playにアプリを登録すると、そのカテゴリ(ゲーム、ツールなど)毎のAndroidバージョンのシェア情報が得られるようになる。日本をターゲット市場にしたアプリをリリースしていれば、そこから情報が得られる
- Android 3.xの情報が出ていないが、1.0%以下のため。3.xはもともと少なかった上に、バージョンアップで4.xになった機種があるため。3.xは無視して良い。

Tips - #06 Androidの画面解像度

- Android の画面解像度は多種多様
- Androidの断片化問題のひとつ
- 画面解像度にどのように対応するか

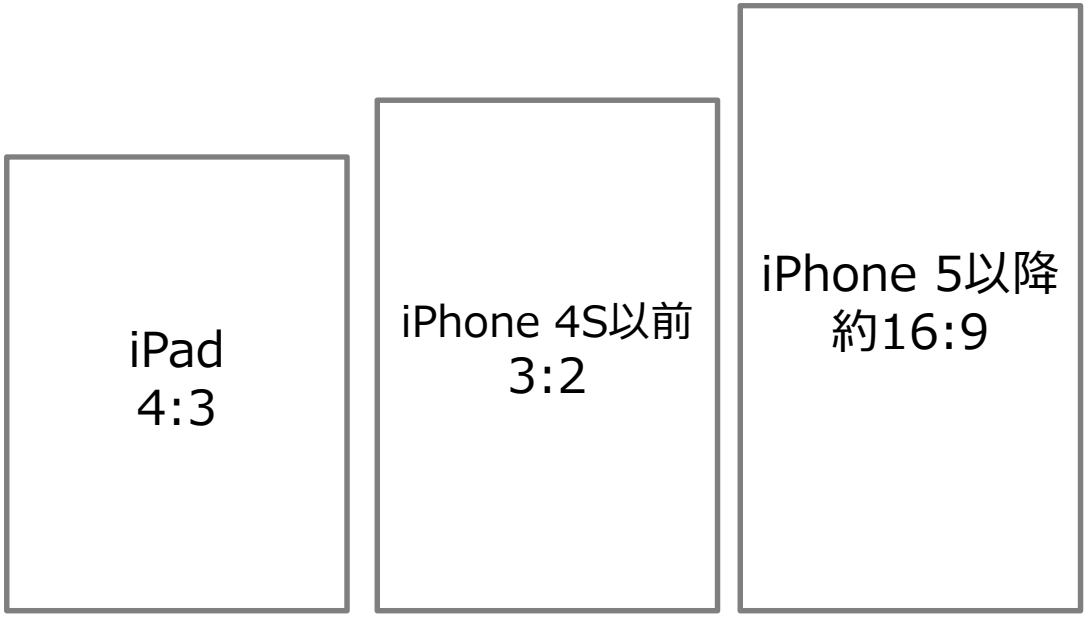
Tips - #06 Androidの画面解像度

解像度	アスペクト比	Androidの主な解像度一覧(国内発売製品)
320x480	2:3	採用機種少ない
480x800	3:5	Nexus One, Nexus Sで採用。2010～2011には非常に多かった。9:16と10:16の中間
480x854	約9:16	2010～2011には非常に多かった。
540x960	9:16	2011～2012には非常に多かった。
640x960	2:3	採用機種少ない
600x1024		採用機種少ない(一部タブレット)
768x1024	3:4	意外なことに採用機種少ない(iPadの解像度)
720x1280	9:16	Galaxy Nexusで採用。2011末頃～採用多い
768x1280	3:5	Nexus 4で採用。他の搭載機は非常に少ない。9:16と10:16の中間
800x1280	10:16	Nexus 7(2012)で採用。2011末頃～タブレットそれなりに多い
1080x1920	9:16	Nexus 5で採用。2013～現在まで、非常に多い
1200x1920	10:16	Nexus 7(2013)で採用。2012～タブレットで多い
1440x2560	9:16	採用機種少ない
1600x2560	10:16	Nexus 10で採用。2013～タブレットに採用されはじめた

16:10	1.6
5:3	1.666…
1024x600	1.706…
16:9	1.777…

Tips - #06　Androidの画面解像度

- 実はiOSのほうがアスペクト比問題は深刻



4:3	1.333…
3:2	1.5
16:10	1.6
5:3	1.666…
1024x 600	1.706…
16:9	1.777…

iOS
Android

Tips - #06　Androidの画面解像度

Unity　Gallery　Asset Store　Learn　Community　Company　Buy　Download

Hardware Stats　Web Player　Mobile　Standalone　Editor

Mobile (Android) Hardware Stats　2014-05

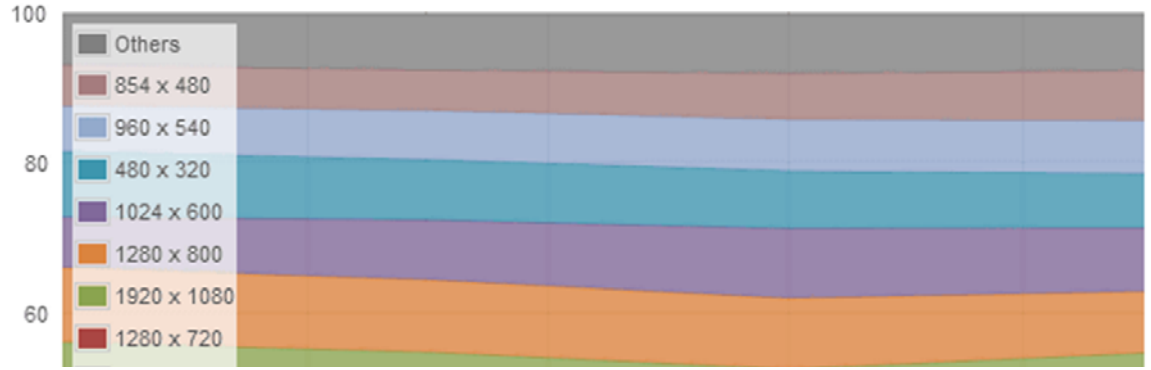
All　Android　iOS　WP8　WinRT

« Back to Overview

Display resolution

Top on 2014-05:

- 800 x 480: **23.8%**
- 1280 x 720: **18.2%**
- 1920 x 1080: **12.7%**
- 1024 x 600: **8.5%**
- 1280 x 800: **8.2%**
- 480 x 320: **7.2%**
- 960 x 540: **7.0%**
- 854 x 480: **6.7%**



Tips - #06 Androidの画面解像度

- Unity Hardware Statistics では、800x480のシェアが23.8%でトップだが、国内ではここまで多くないのではないか
- おそらく、2～4位の1280x720(HD), 1920x1080(FullHD), 1136x640(iPhone5)あたりが主なターゲットのはず
- Android端末一覧 http://ja.wikipedia.org/wiki/Android端末一覧 と併用すると良い。正確性の保証はないが、概ね正しいはず

Tips - #07 カタログスペックと実測値の乖離

Xperia arc (SO-01C)

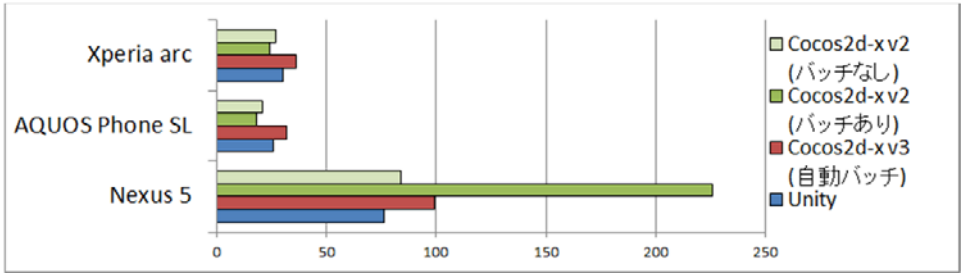
Snapdragon MSM8255 1GHz(1 Core) 480×854 (2011春発売)

AQUOS PHONE SL (IS15SH)

Snapdragon S2 MSM8655T 1.4GHz(1 Core) 540×960 (2012夏発売)

原因は不明

- 省電力モードを完全にOFFにできないため？
- プリインストールアプリの問題？
- メーカーのチューニングの差？



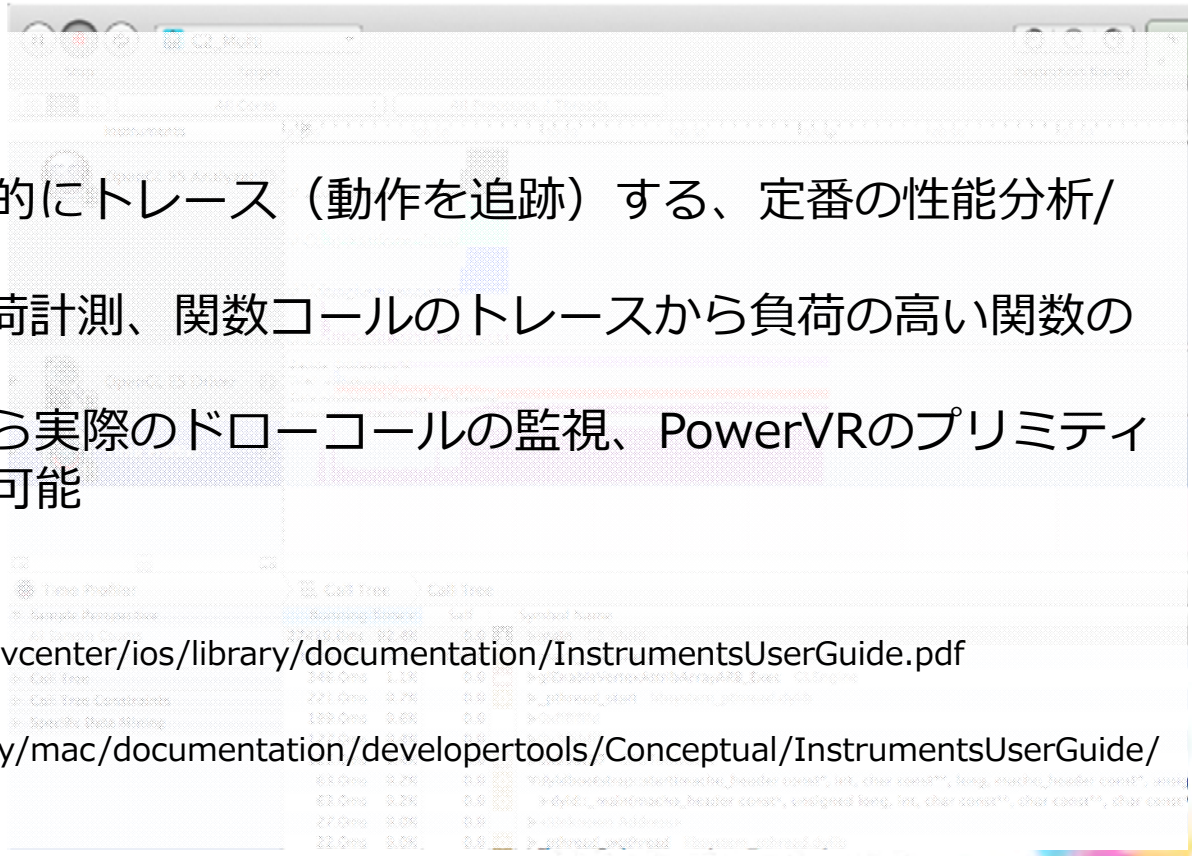
	Cocos2d-x v2 (バッチなし)	Cocos2d-x v2 (バッチあり)	Cocos2d-x v3 (自動バッチ)	Unity
Xperia arc	27個	24個	36個	30個
AQUOS PHONE SL	21個	18個	32個	26個
Nexus 5	84個	226個	99個	76個

Tips - #08 プロファイラ

- Instruments
- Android Debug Monitor
- OpenGL ES Trace
- Trepn Profiler

Tips - #08 プロファイラ

Instruments



- OS XやiOSのコードを動的にトレース（動作を追跡）する、定番の性能分析/テストツール
- TimeProfilerからCPU負荷計測、関数コールのトレースから負荷の高い関数の特定が短時間で可能
- OpenGL ES Analyzerから実際のドローコールの監視、PowerVRのプリミティブのバッチングの監視も可能

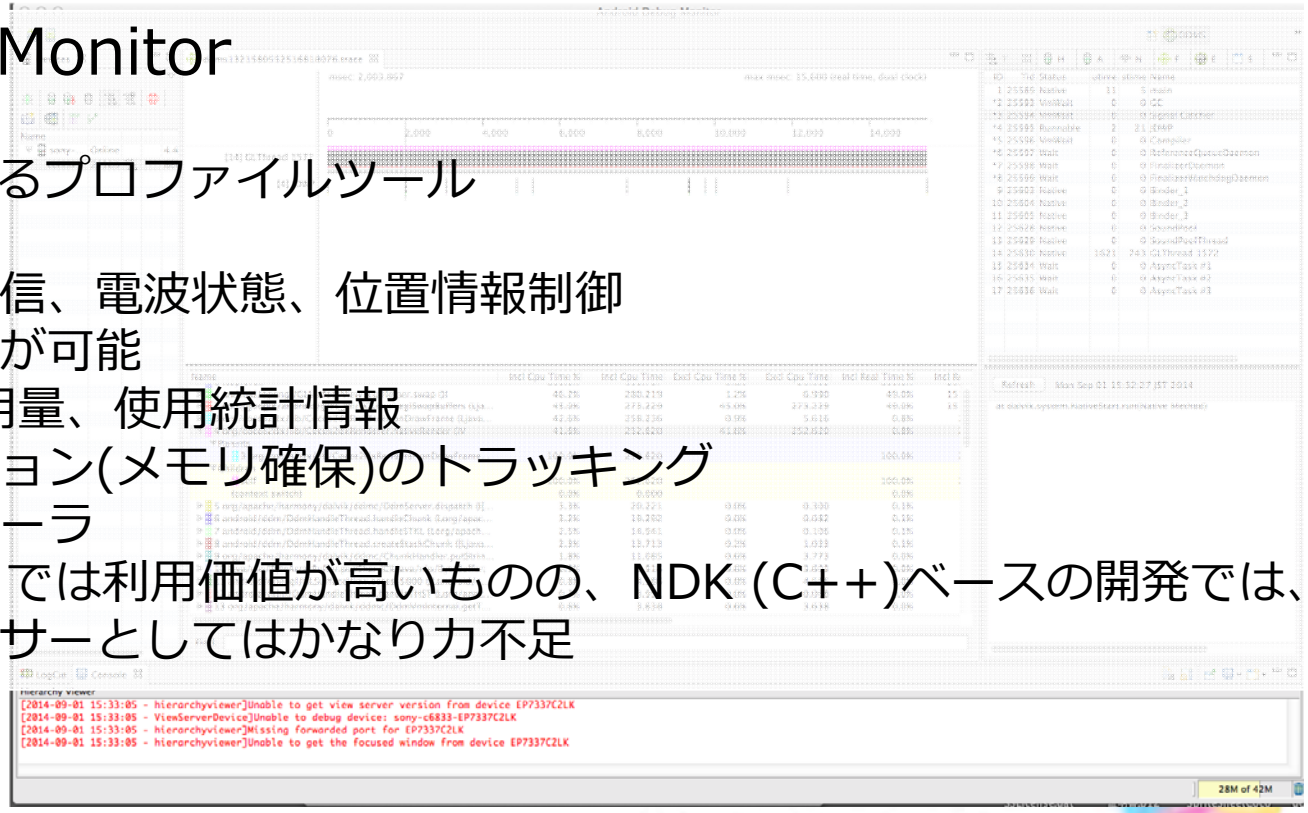
- 日本語情報(少し古い)
  https://developer.apple.com/jp/devcenter/ios/library/documentation/InstrumentsUserGuide.pdf
- 英語(最新)
  https://developer.apple.com/library/mac/documentation/developertools/Conceptual/InstrumentsUserGuide/Introduction/Introduction.html

Tips - #08 プロファイラ

Android Debug Monitor

Android SDKに含まれるプロファイルツール

- エミュレータへの着信、電波状態、位置情報制御
- スレッドの状態確認が可能
- Heap(メモリ)の使用量、使用統計情報
- メモリ・アロケーション(メモリ確保)のトラッキング
- ファイルエクスプローラ
- Javaベースのアプリでは利用価値が高いものの、NDK (C++)ベースの開発では、関数コールのトレーサーとしてはかなり力不足



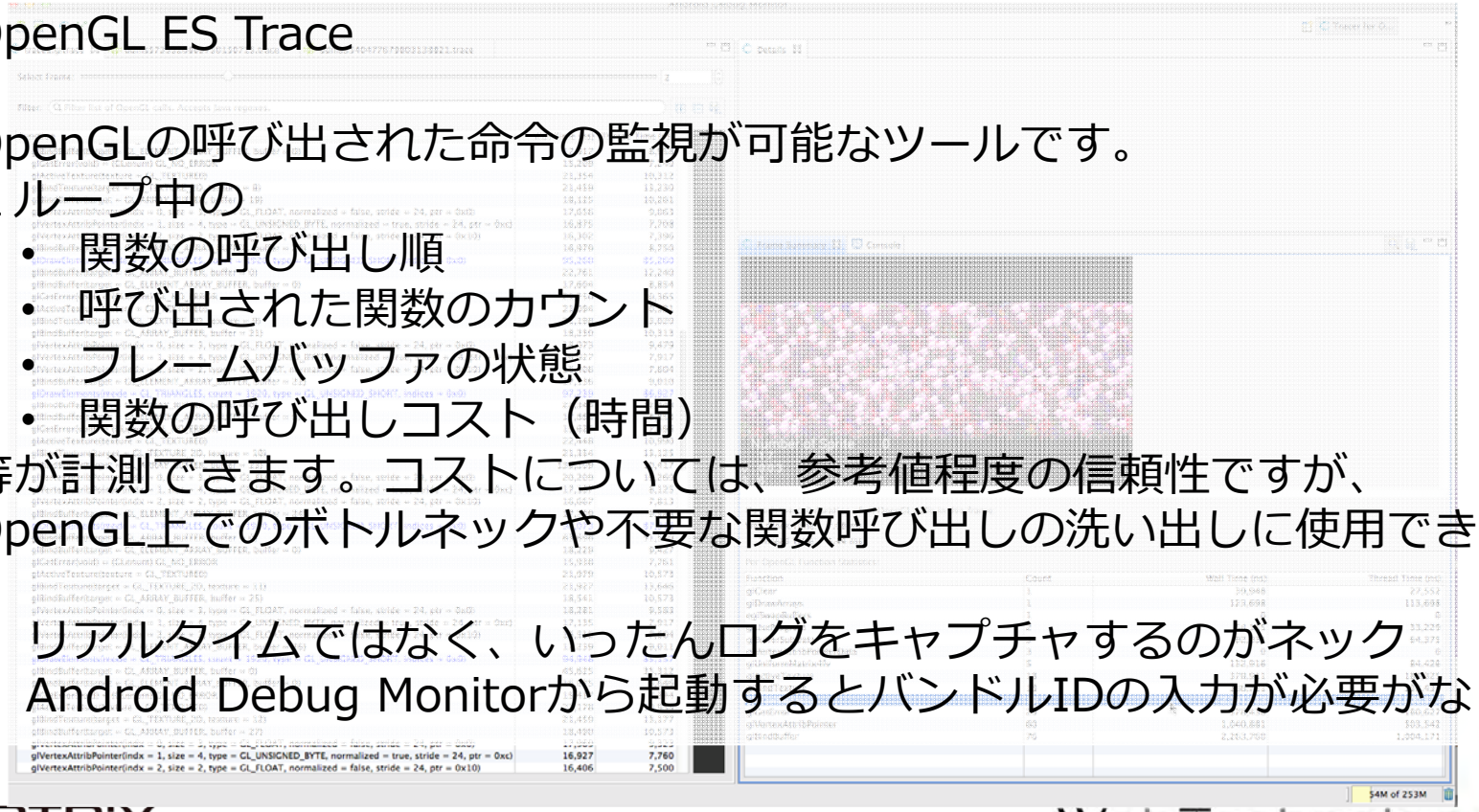
Tips - #08 プロファイラ

Android用のさまざまなプロファイラ

- OpenGL ES Trace
- チップメーカー毎に提供されているツール
  - Qualcomm Snapdragon Performance Visualizer
  - Qualcomm Trepn Profiler
  - PerfHUD
  - Mali Graphics Debugger
  - Power VR Performance Analysis Utilities

Tips - #08 プロファイラ

OpenGL ES Trace



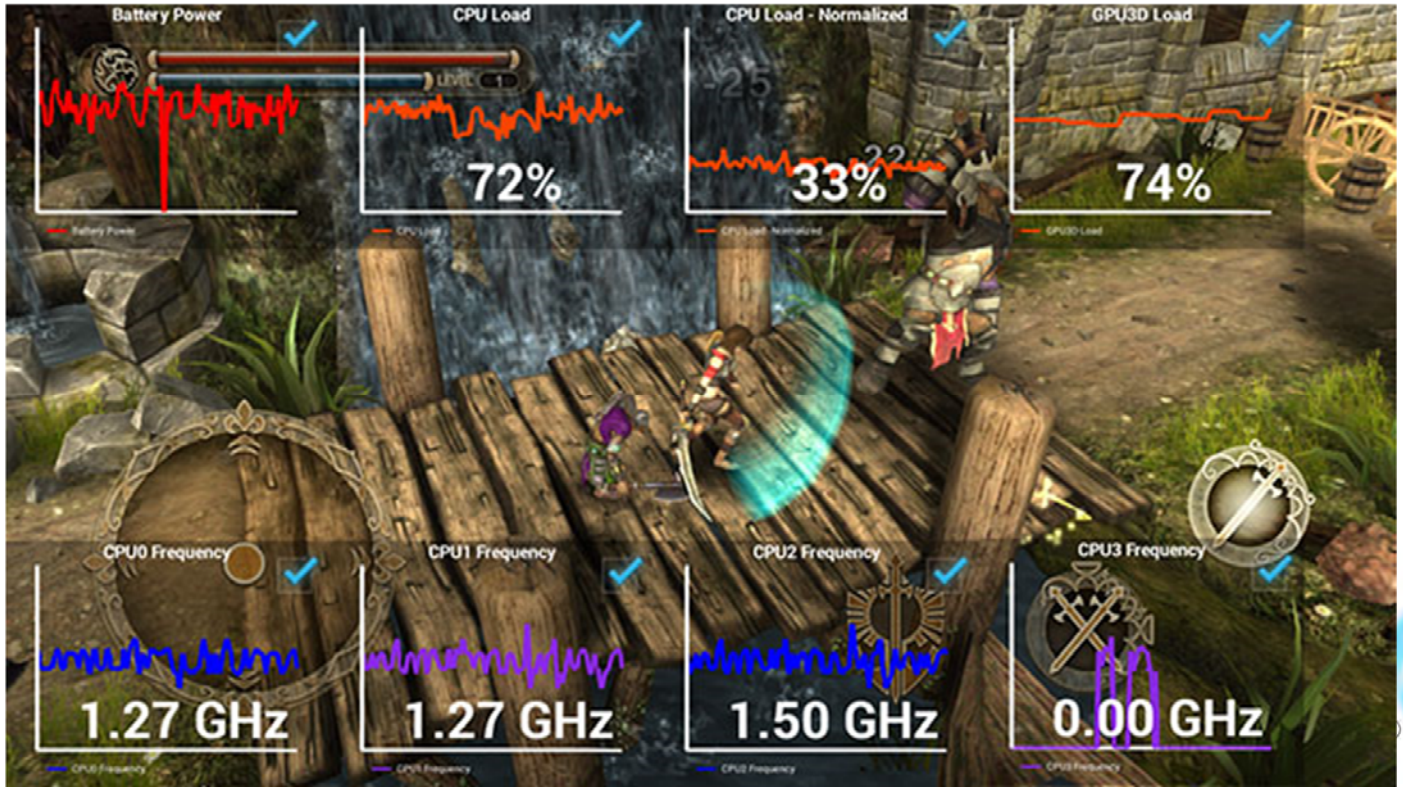
OpenGLの呼び出された命令の監視が可能なツールです。
１ループ中の
- 関数の呼び出し順
- 呼び出された関数のカウント
- フレームバッファの状態
- 関数の呼び出しコスト（時間）

等が計測できます。コストについては、参考値程度の信頼性ですが、
OpenGL上でのボトルネックや不要な関数呼び出しの洗い出しに使用できます。

- リアルタイムではなく、いったんログをキャプチャするのがネック
- Android Debug Monitorから起動するとバンドルIDの入力が必要がなく便利

Tips - #08 プロファイラ

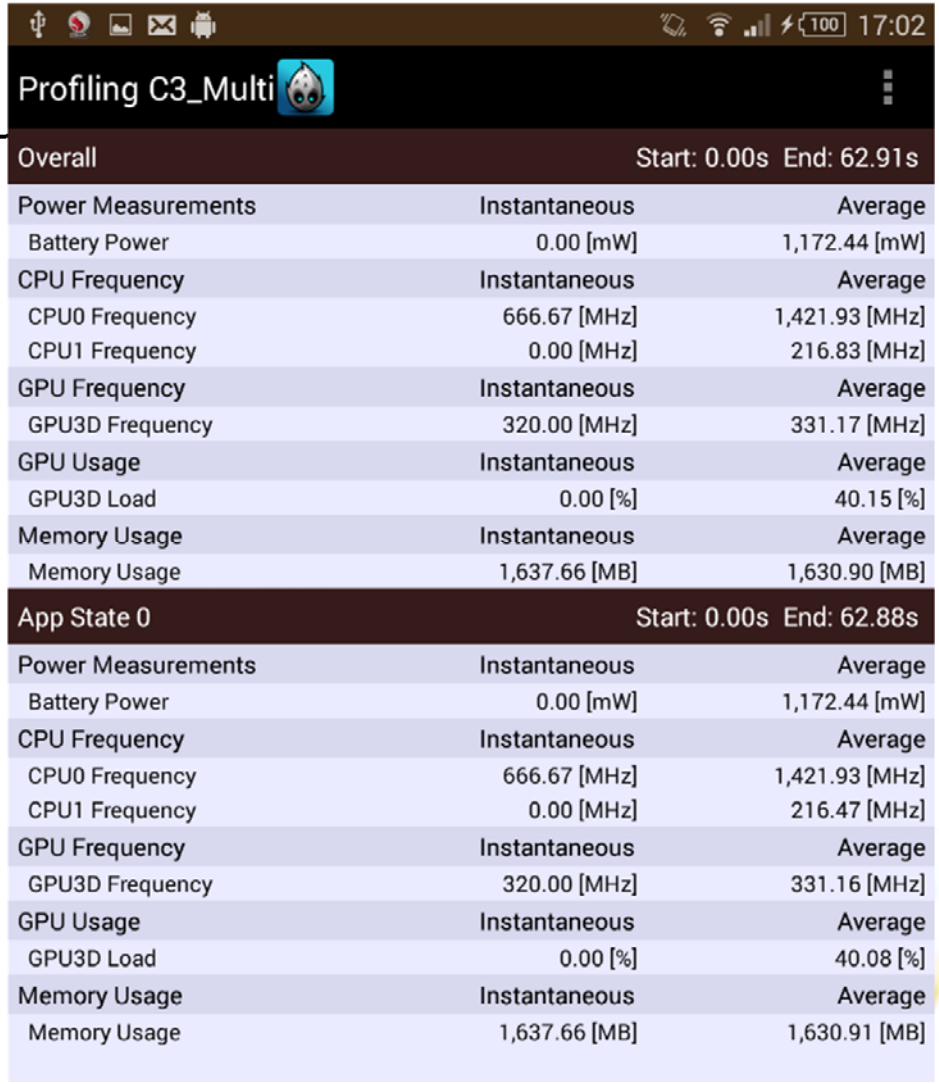
Qualcomm Trepn Profiler



OPTPiX

Tips - #08 プロファイラ

Qualcomm Trepn Profiler



OPTPiX

最後に

手戻りを起こさないためのポイント

- スマートフォンは、機種による性能差が非常に大きい
- エンジンによる性能差は、それに比較すれば小さいほうなので、目的にあったものを選択すべし
- ベンチマークは重要。ただし、ピーク値に注意
- 適切な画像リソースを使おう
- プロファイラを使おう

ご清聴ありがとうございました。