PVRTCの傾向と対策

PVRTCとは？（基本情報）

PowerVR Texture Comression

フォーマット: 4bpp, 2bpp

圧縮単位: 4×4ピクセル

画像の縦横サイズは2の累乗の正方形(256x256等)のみ

PVRTCとは？（補足情報）

- Imagination Technologies社のPowerVR用
- **iPhone, iPadはPVRTCのみサポート**
  * Appleのドキュメントには、将来にわたって必ずサポートされるとは限らないとの注記あり
- Androidでも、一部の機種(PowerVR搭載機)で使用可能

PVRTCの対策

トラブル　その1
色が滲む・・・？

PVRTC 画像をコンバートする

WebTechnology®
OPTPiX imésta

オリジナルの画像

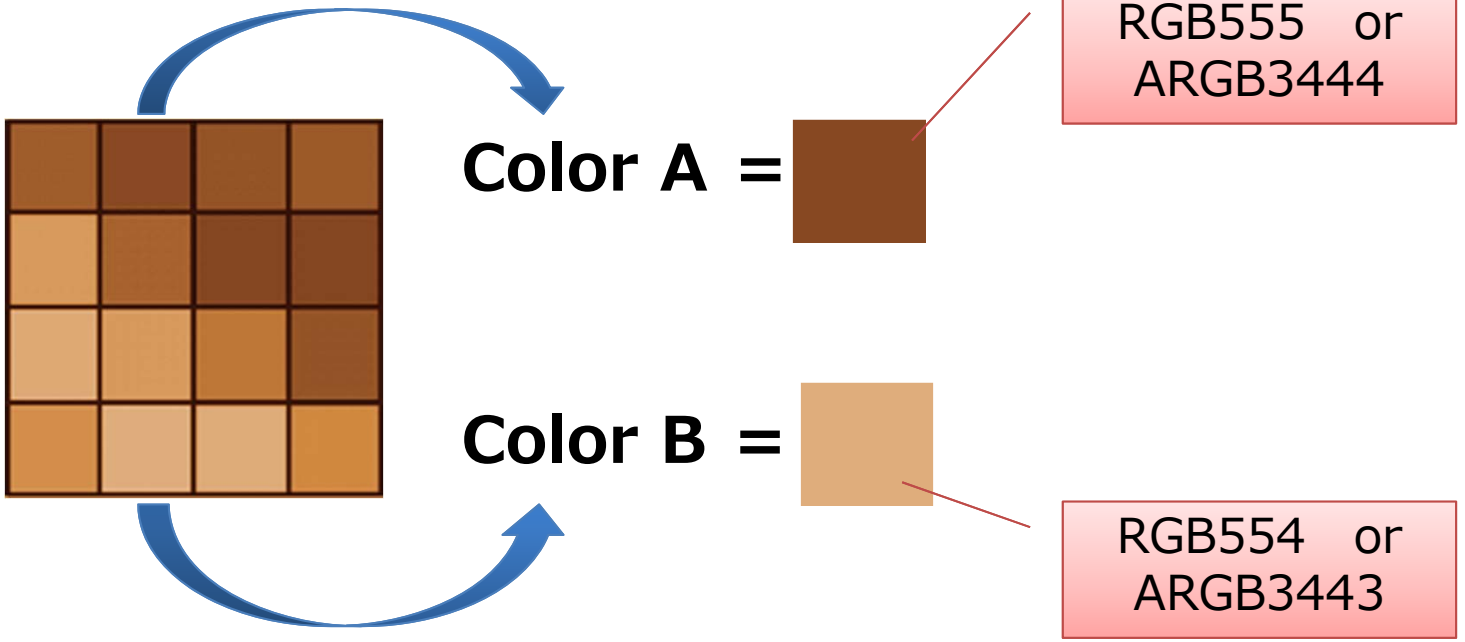
PVRTC 画像をコンバートする (2)

PVRTC　4bppの画像

結論から言うと
滲みを克服することは難しい！

PVRTCの理論 (1) 代表色を決める

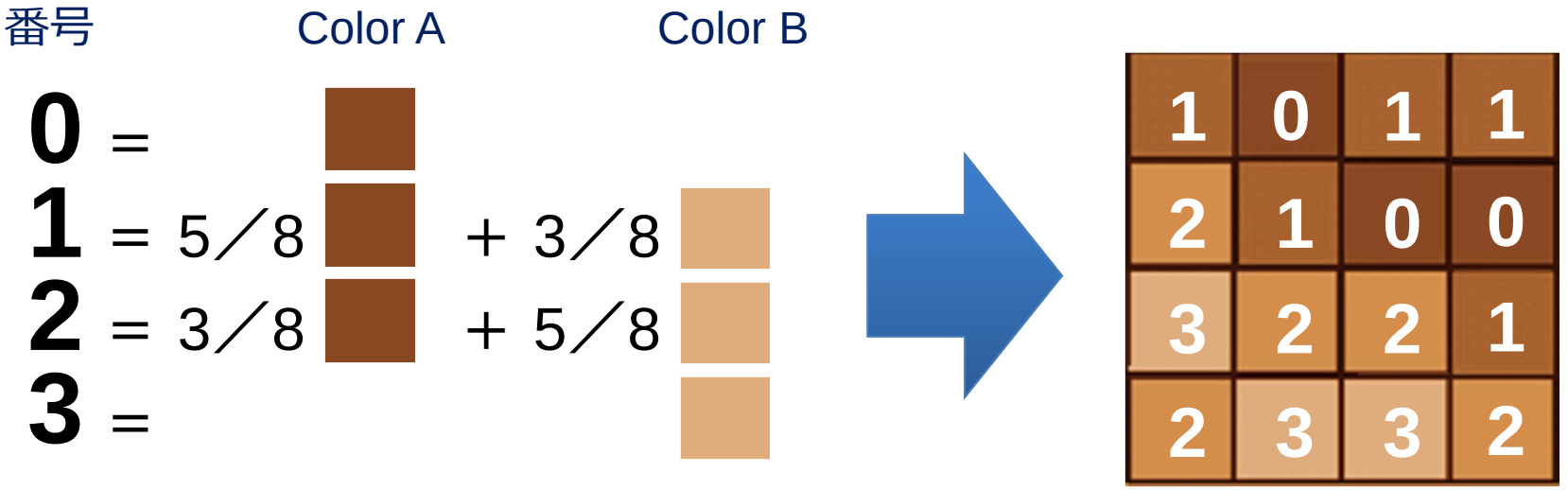
ブロックごとに2色を決める



＊ ブロックに無い色でも指定できます
＊ 実際には複雑な方法で2色を選ぶ必要があります（詳細は後述）

PVRTCの理論 (2) 代表色のブレンド方法を決める

ピクセルの色を、ブレンド方法の番号に置き換える



この数値を Modulation Data と呼び、
配列データとして保存する

PVRTCの理論 (3) ブロック内のデータ

2色とModulation Dataの情報になる

Color A = ＝ **16 ビット**

Color B = ＝ **15 ビット**

Mode Bit ＝ **1 ビット**

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

Modulation Data
2ビット × 16ピクセル
＝ **32 ビット**

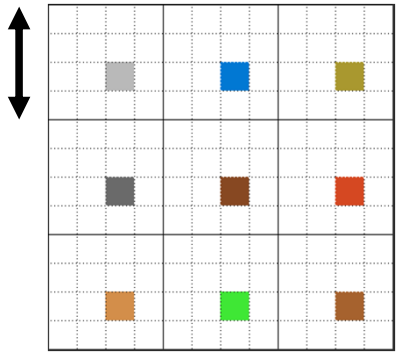
復元方法に特徴あり

1ブロックあたり 64 ビット

PVRTCの理論 (4) 代表色の補間

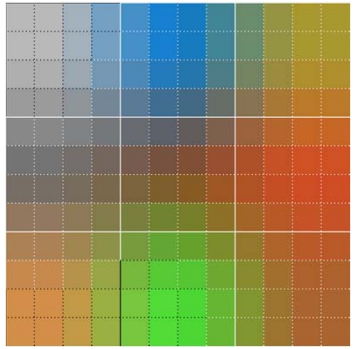
Color A, B をそれぞれ、周囲のブロックと補間する

1ブロック
(4×4)

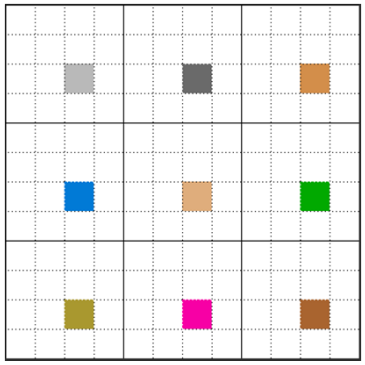


Color A

補間

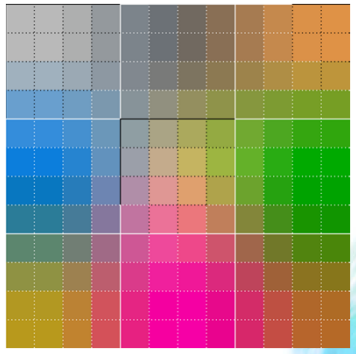


4×4ピクセルを補間色で埋める



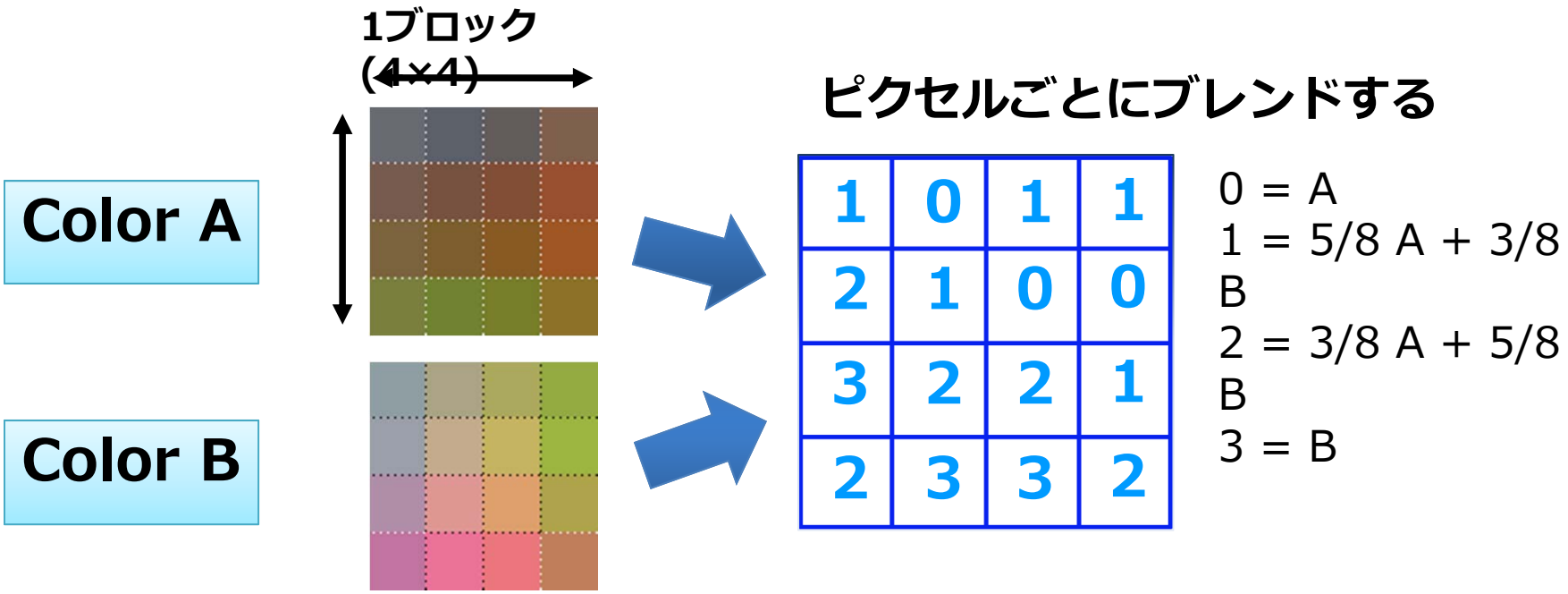
Color B

補間



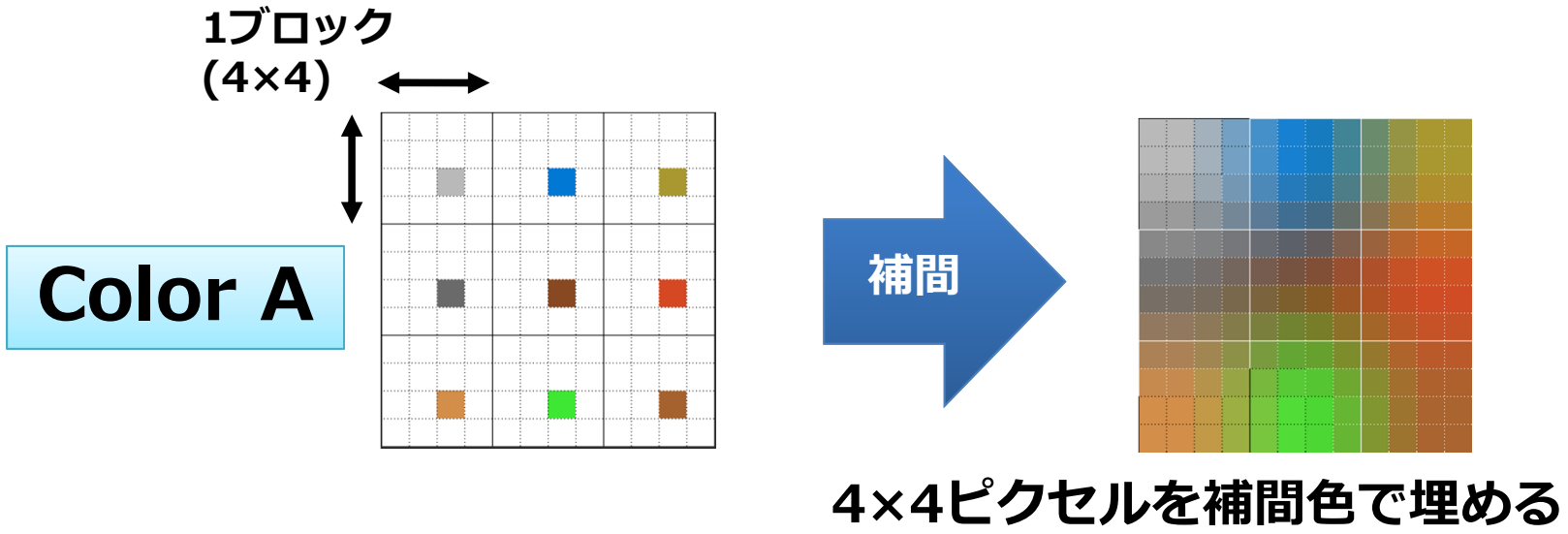
PVRTCの理論 (5) 補間結果をブレンド

補間結果のピクセルごとにブレンドする



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

PVRTCのここがポイント！！



4×4ピクセルを補間色で埋める

（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

輪郭、影など色彩が違いすぎるものを別パーツへ

UIパーツを直接PVRTCで圧縮すると、劣化が大きい

無圧縮画像

PVRTC4bpp画像

輪郭、影など色彩が違いすぎるものを別パーツへ

そこで、一旦文字や輪郭線を別パーツとしてアトラスを作って圧縮し、

この状態で
圧縮しておく

輪郭、影など色彩が違いすぎるものを別パーツへ

表示するときに重ねると、直接圧縮するより綺麗に表示できる

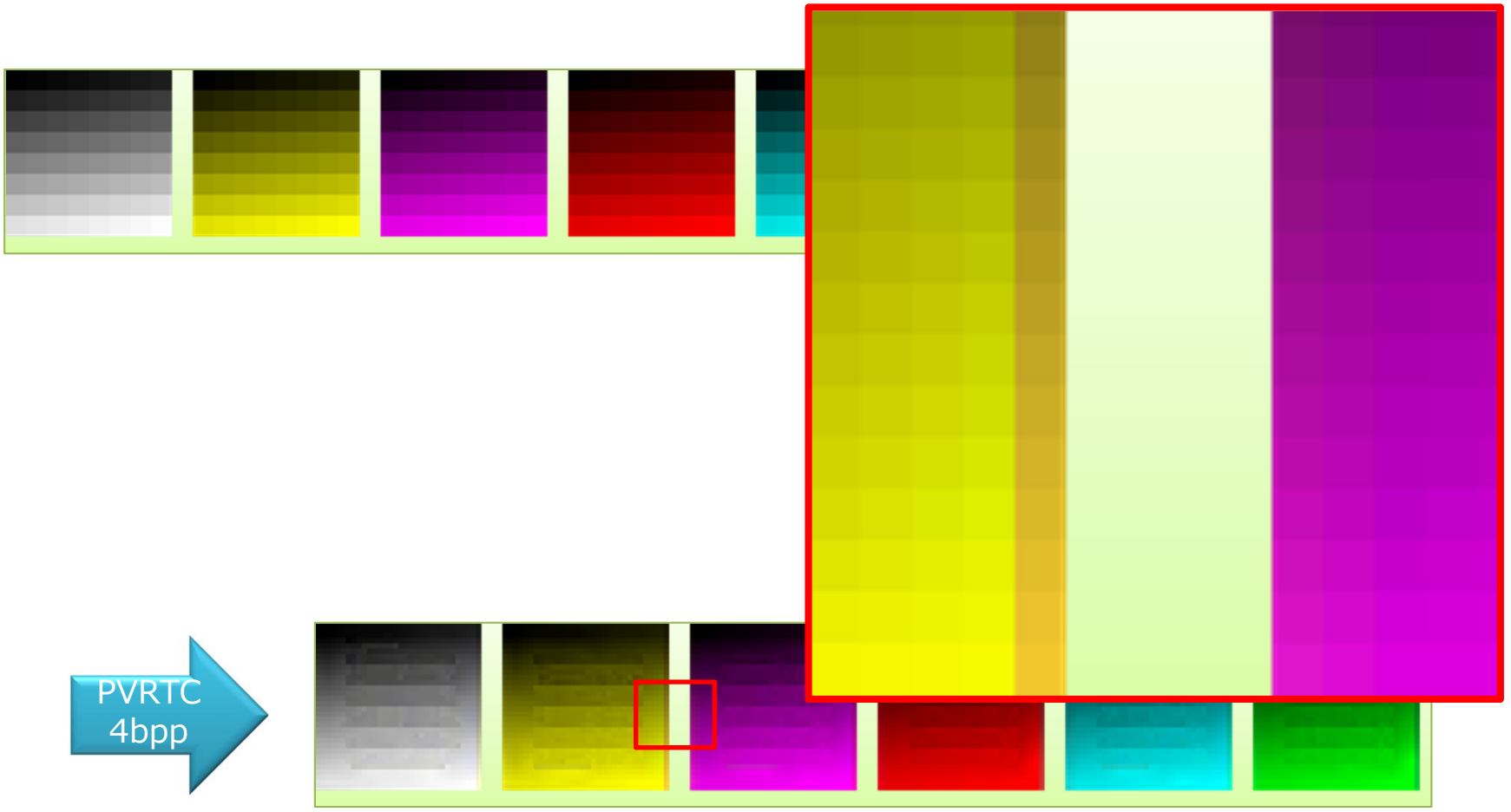
直接PVRTC4bppで圧縮した画像

文字と輪郭を別パーツとして圧縮した画像

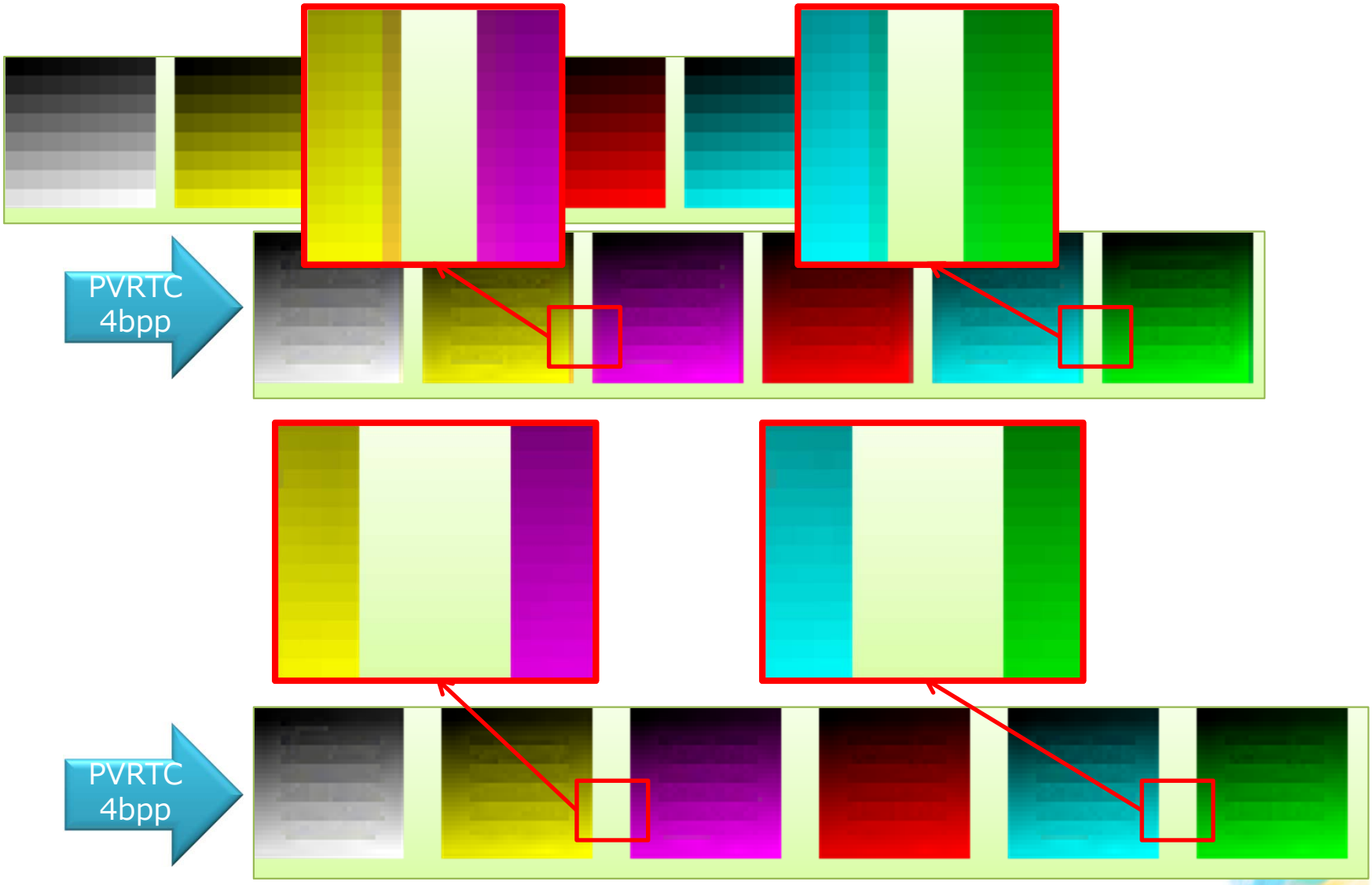
PVRTCの対策

トラブル　その２
距離があっても滲むけど？

離れていても滲むのですが・・・



テクスチャアトラスでのマージンの影響

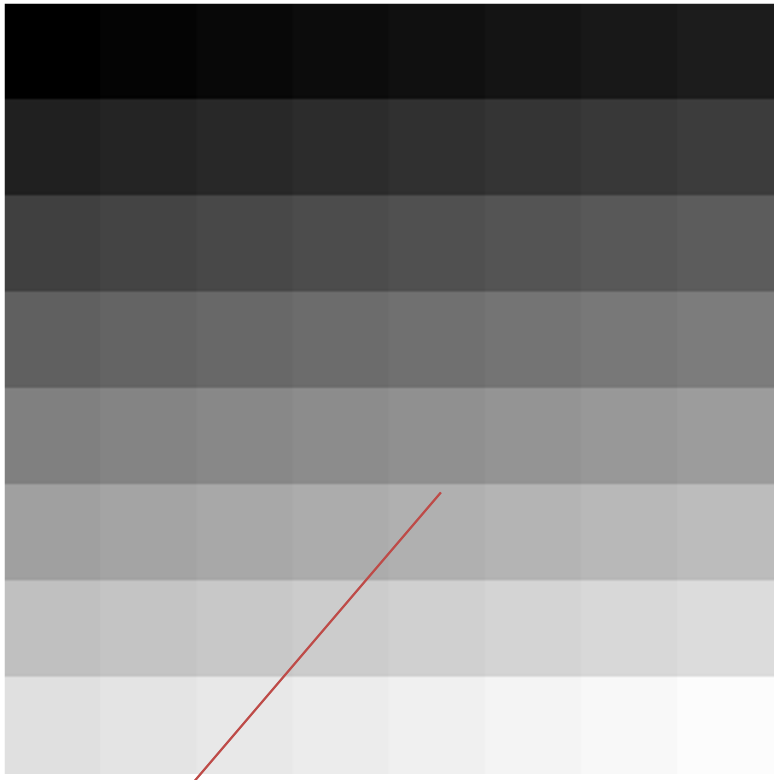


PVRTCの対策

トラブル　その3
ドットが崩れる？？

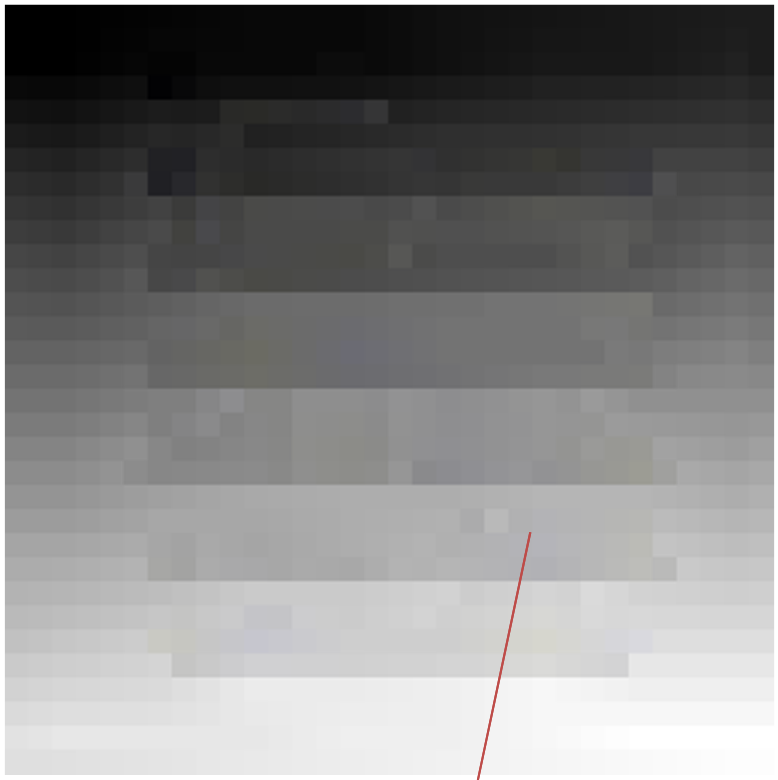
ブロックの形が壊れる

サイズ32x32の画像



細かいグラデーションの
境目のラインが
はっきり見える

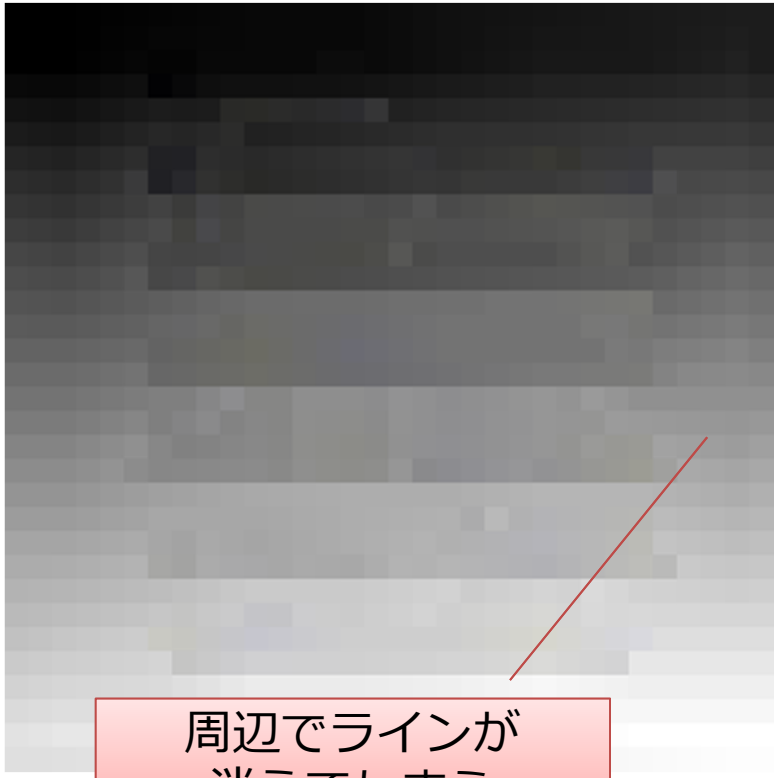
PVRTC 4bpp



周辺でラインが
消えてしまう

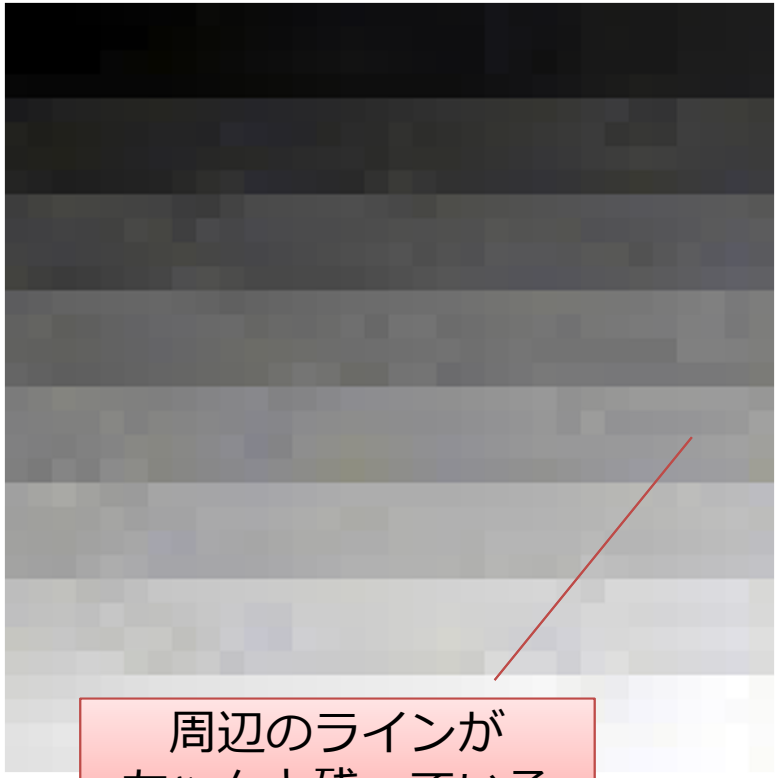
補間による劣化

PVRTC 4bpp



周辺でラインが
消えてしまう

OPTPiX imesta
Clear PVRTC

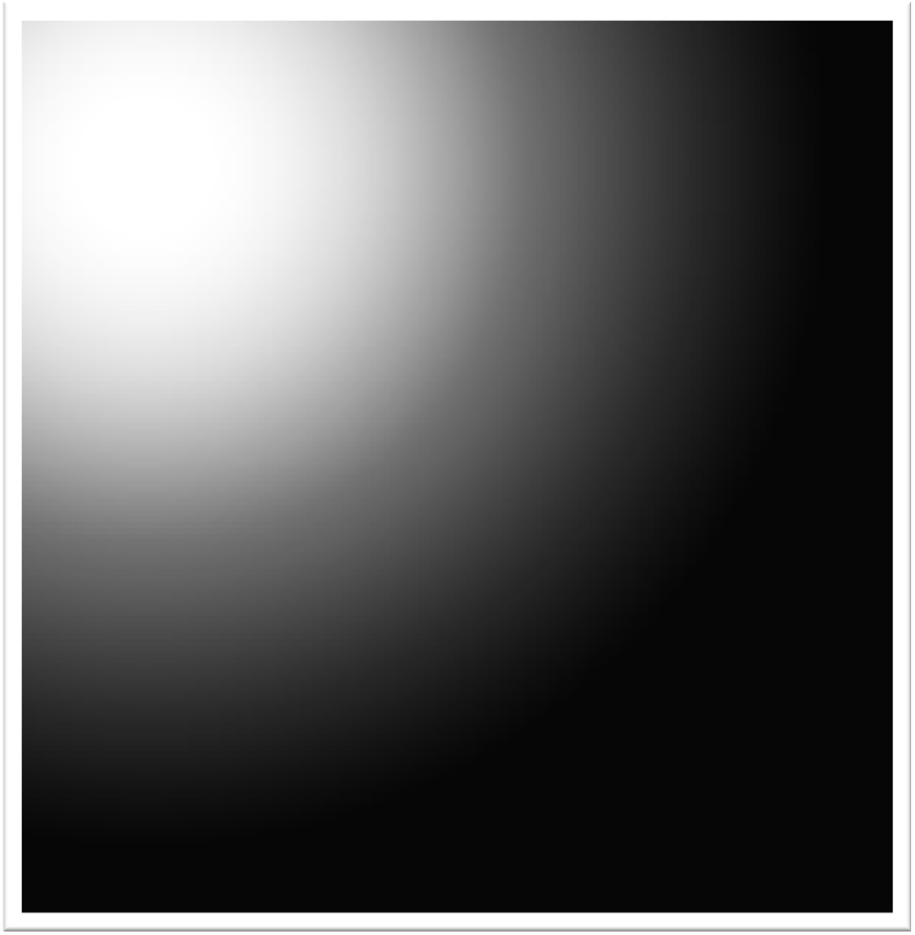


周辺のラインが
ちゃんと残っている

PVRTCの対策

トラブル　その4
半透明グラデーションがもやもや

半透明グラデーションがもやもや

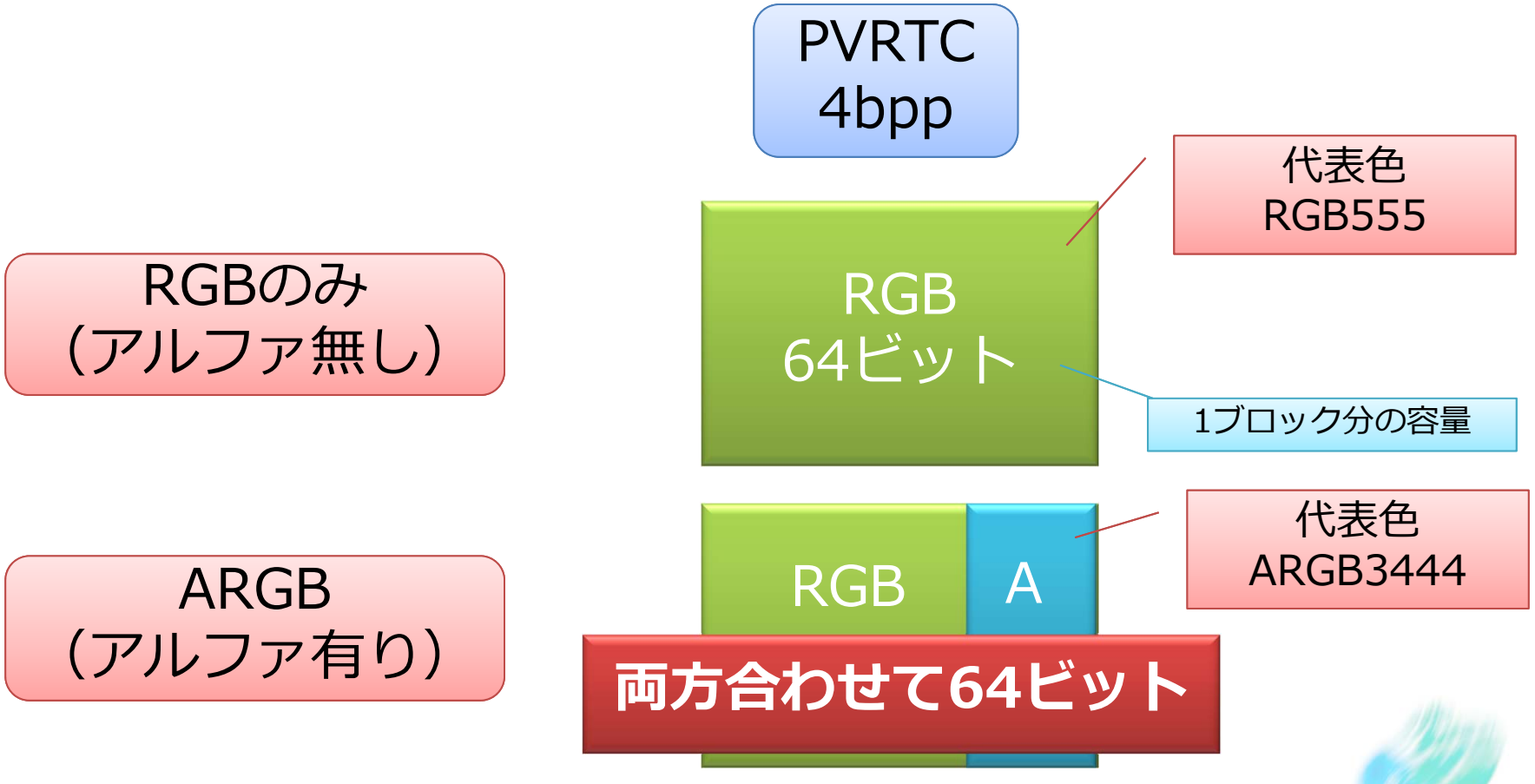


オリジナル
アルファチャンネル

PVRTC4bpp
アルファチャンネル

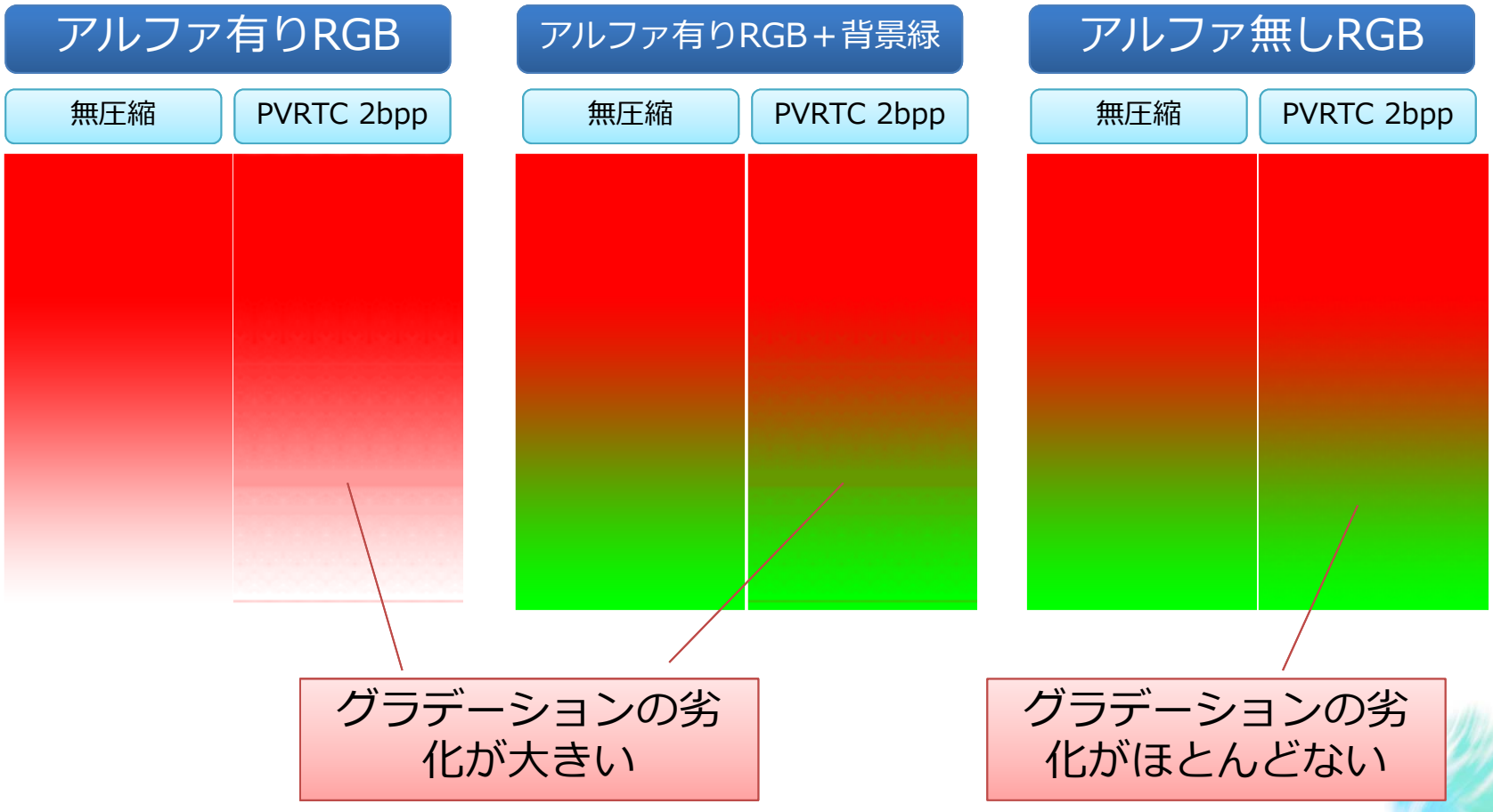
PVRTCでのアルファチャンネルの扱いについて

- 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



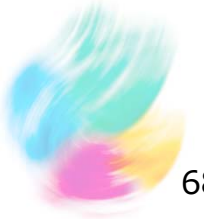
グラデーションの改善

- 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



PVRTCの対策

トラブルその5
PVRTC 2bppでリサイズしたのだけど



PVRTC 2bppでリサイズしたのだけど

PVRTC2bpp 256x256

メモリ節約のため
128x128へ縮小

表示の際に
縦横２倍拡大

劣化が
ひどい

PVRTC2bpp

メモリを節約しながら、もう少し劣化を抑えられないか

PVRTC 2bppモードについて

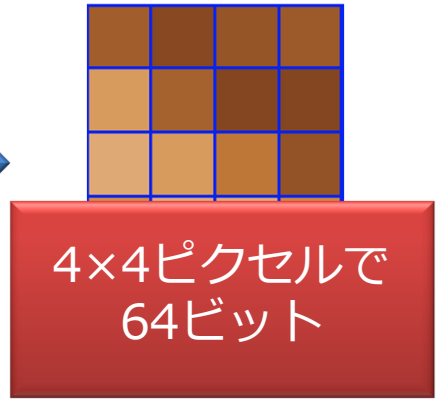
4bpp

Color A = = 16 ビット

Color B = = 15 ビット

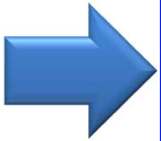
Mode Bit = 1 ビット

Modulation Data = 32 ビット



1	0	1	1	1	0	1	1
0	1	0	0	1	1	0	0
0	1	1	1	0	1	1	1
1	0	0	0	0	1	1	0

2bpp



2bppでは、水平方向の品質が低下しやすい

PVRTC 2bppが横に劣化する秘密　（例）

PVRTC2bpp拡大前

縦横２分の１

縦２分の１

横２分の１

これらを拡大する

PVRTC 2bppが横に劣化する秘密　（例）

PVRTC 2bpp 256x256

PVRTC 2bpp 128x128 を縦横2倍表示

PVRTC2bpp拡大結果

(僅かながら)最も劣化が少ない

PVRTC 2bpp 256x128を縦2倍表示

PVRTC 2bpp 128x256を横2倍表示

PVRTC 2bppが横に劣化する秘密　（例）

PVRTC 4bpp 256x256

PVRTC 4bpp 128x128 を縦横2倍表示

PVRTC4bppの場合

2bppの場合より差異が小さい

PVRTC 4bpp 256x128を縦2倍表示

PVRTC 4bpp 128x256を横2倍表示

PVRTCにおける傾向と対策まとめ

UIパーツは使用に堪えないほど劣化することが多い

- 可能であれば劣化の少ない同系色でパーツ分けを行う

展開時にバイリニア補間をかけるので、アトラス内で近いパーツの色が混ざったり、細かい模様が崩れたりする

- アトラス内のマージンを大きめにとる
- OPTPiX imesta のClearPVRTCを使う

アルファの表現力が低い

- 可能ならグラデーションをRGBで表現する

2bppモードは横方向に劣化しやすい

- 2bppモードの水平方向のクオリティは、垂直方向の1/2程度と考える

ETCの傾向と対策

ETCとは？（基本情報）

Ericsson Texture Compression

フォーマット: 1種類のみ

圧縮単位: 4×4ピクセル

ETCとは？（補足情報）

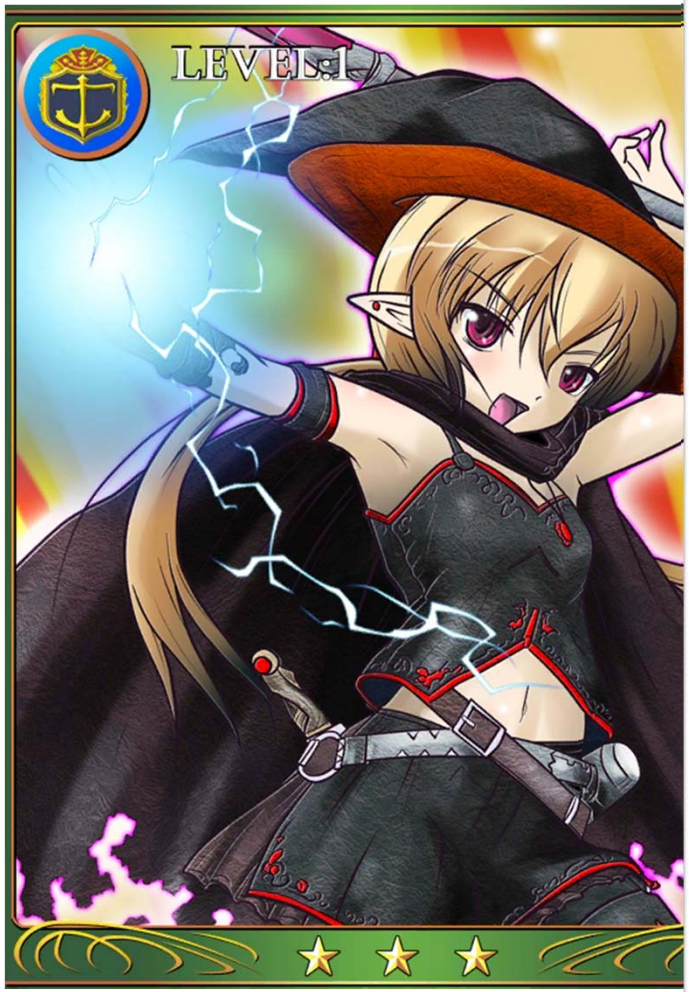
- **Androidでは全ての機種で共通して使える**
- **アルファが使えない(抜き色も使えない)**

- 余談1…ETC2という新バージョンではアルファも使えるが、まだ普及していないので利用できるのはこれから
- 余談2…一部の携帯型コンシューマーゲーム機もサポートしている

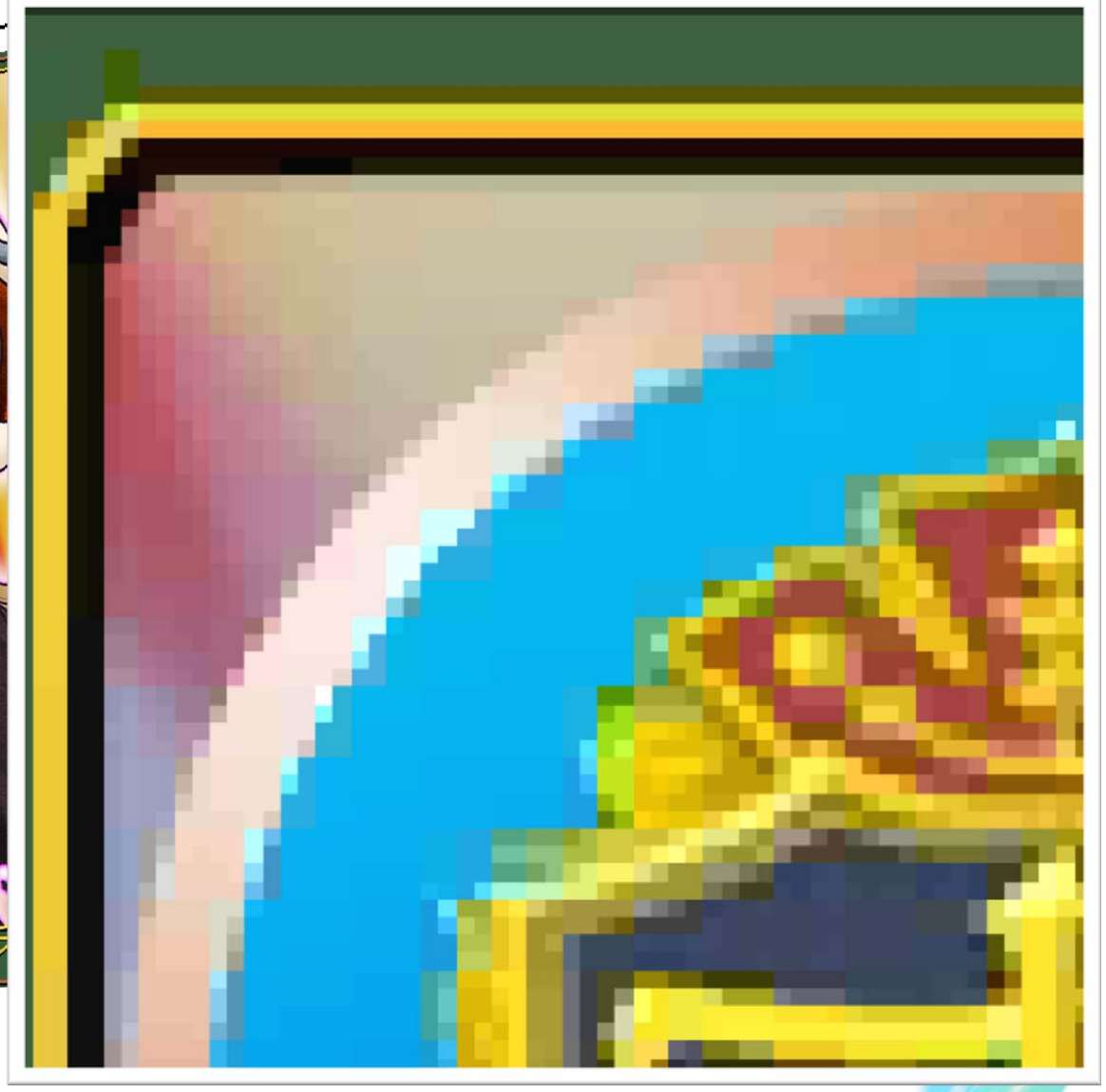
ETCの対策

トラブル　その1
やっぱりブロックノイズが出る？

やっぱりブロックノイズ

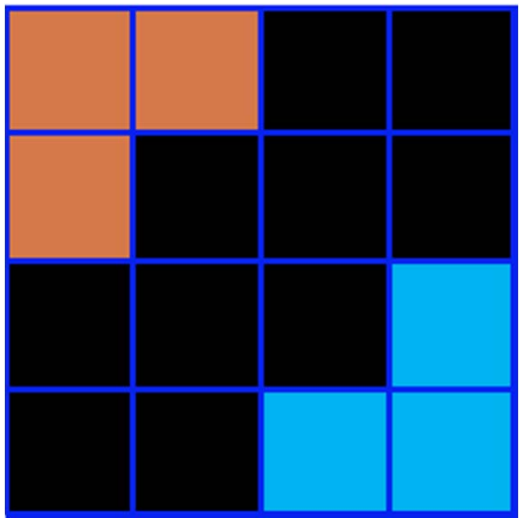


ETC

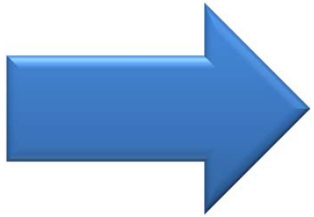


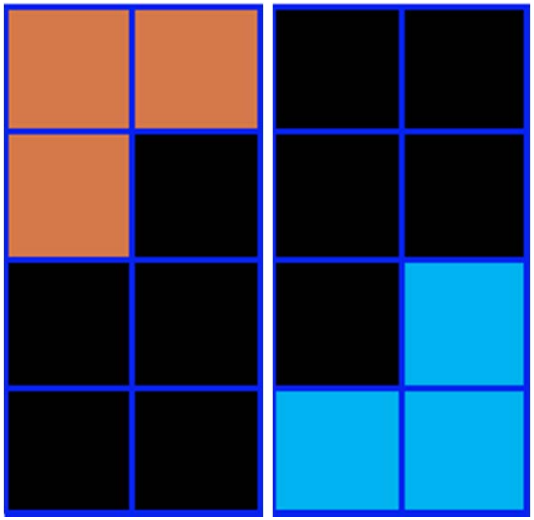
ETCの理論 (1) サブブロック分割

- ４×４のブロックを２×４のサブブロックに分割する

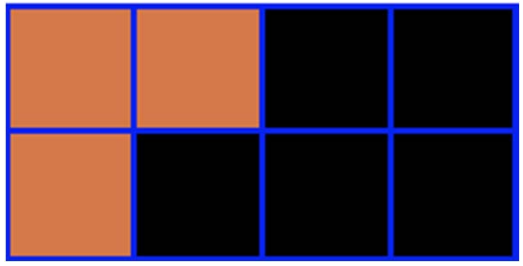


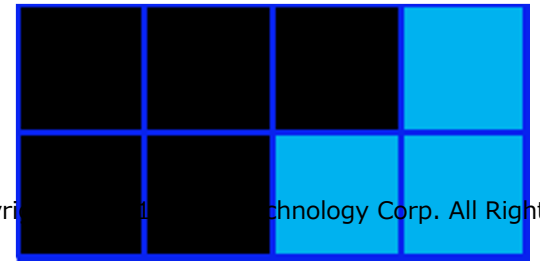
無圧縮画像





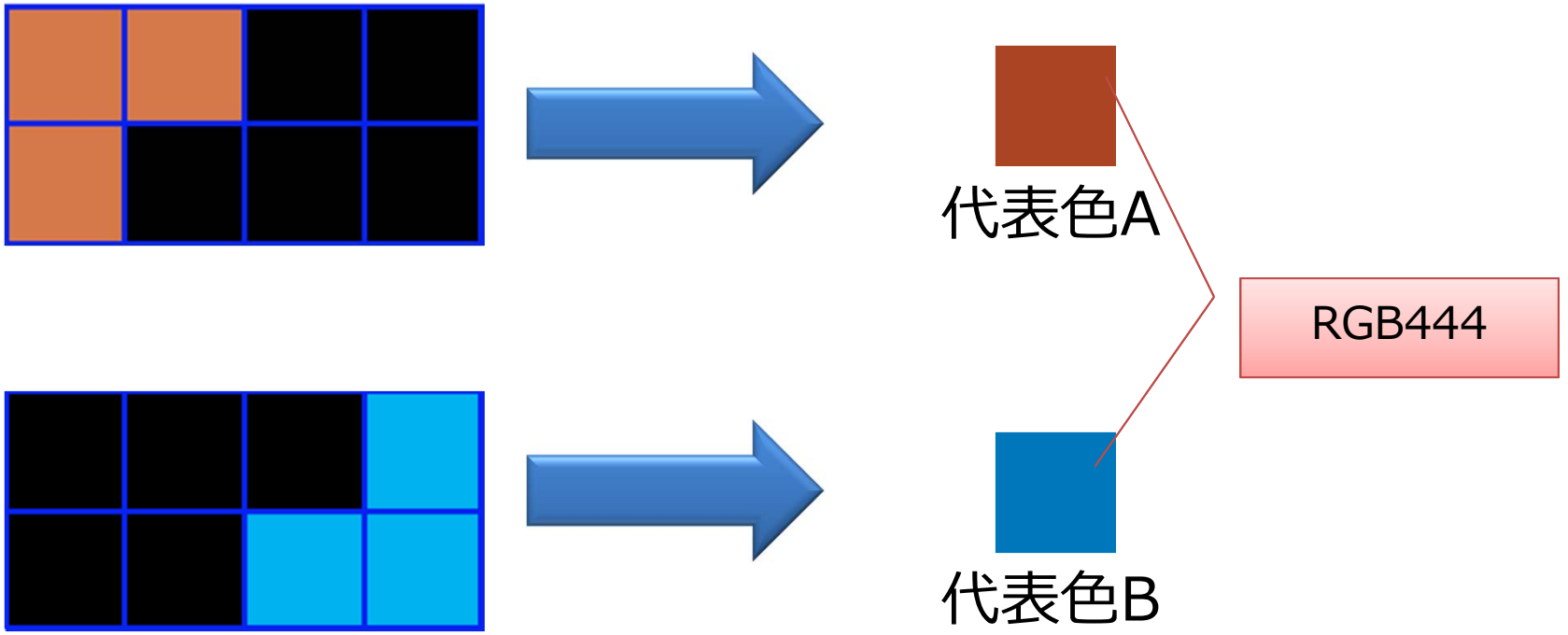
OR





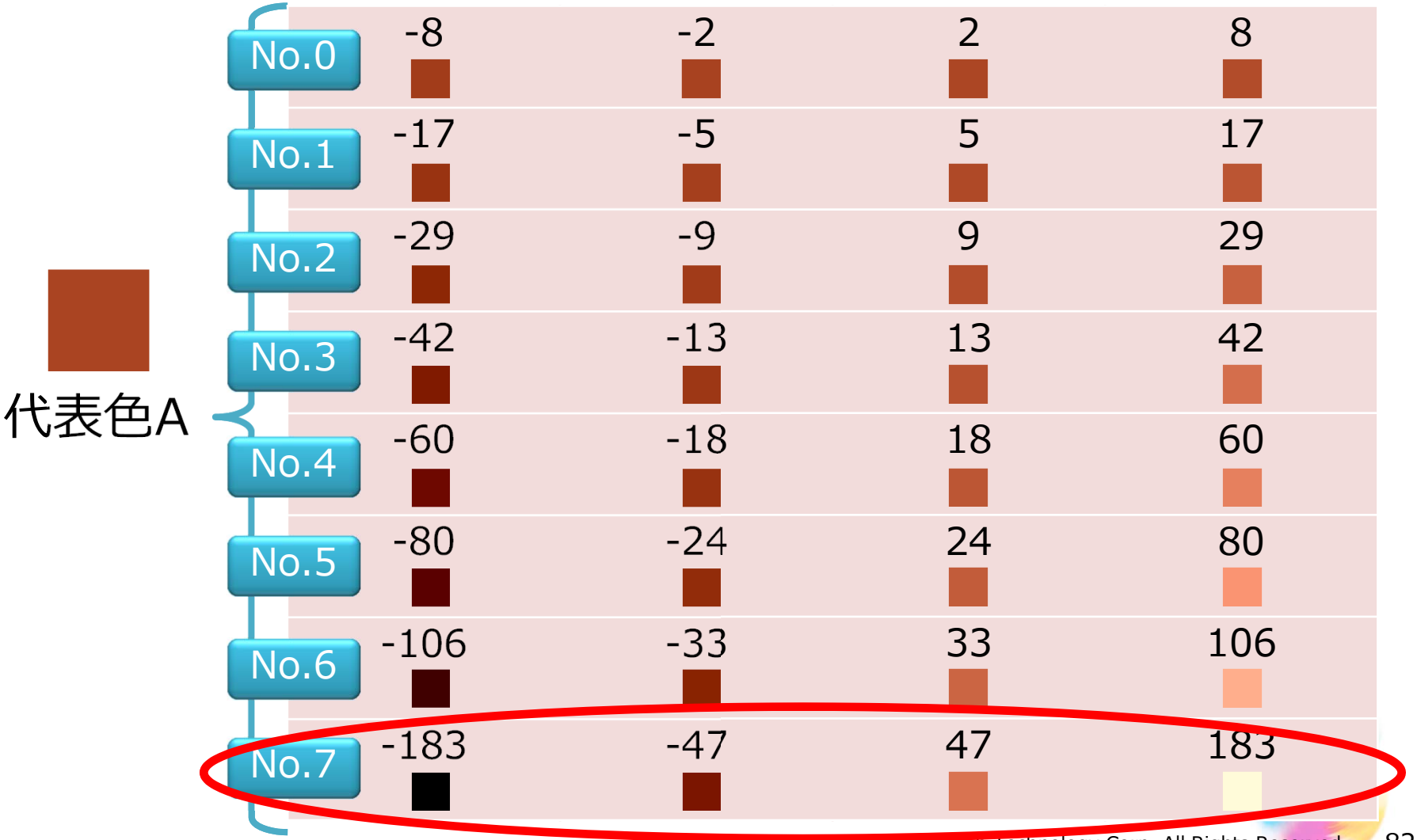
ETCの理論 (2) 代表色を決める

- サブブロック毎に代表色を決める



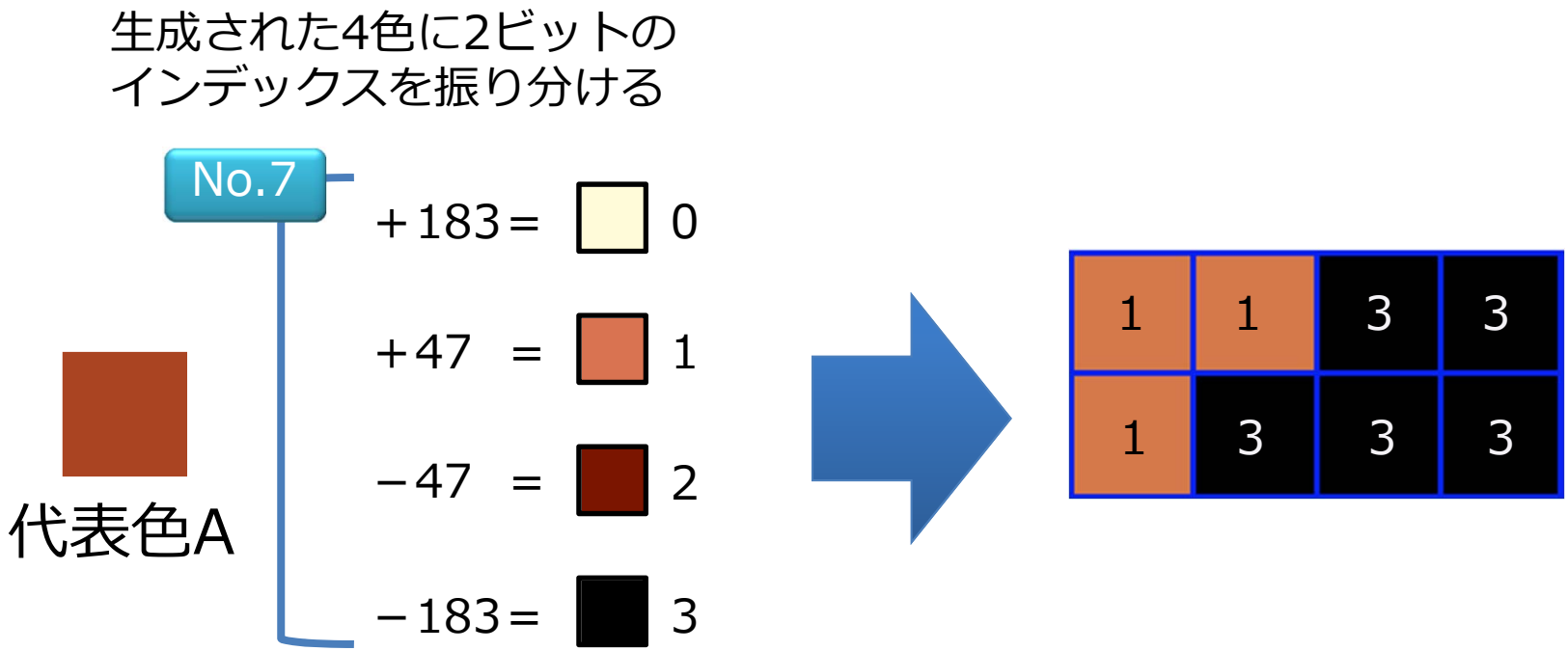
ETCの理論 (3) 代表色から4色を作る

- 代表色に対する輝度の変換テーブルを選ぶ



ETCの理論 (4) 色を振り分ける

- 代表色と変換テーブルから生成された4色を各ピクセルに振り分ける



ETCの理論 (5) ブロック内のデータ

1	1	3	3
1	3	3	3
3	3	3	1
3	3	1	1

インデックス2ビット
× 8ピクセル
× 2
= **32 ビット**

代表色 A =

代表色 B =

12ビット
(RGB444)
× 2色
= **24 ビット**

Aの変換テーブル No.7

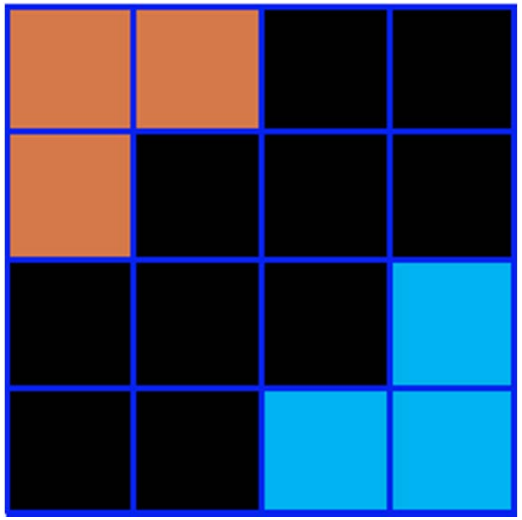
Bの変換テーブル No.7

3ビット × 2色
= **6 ビット**

Flip Bit = **1 ビット**

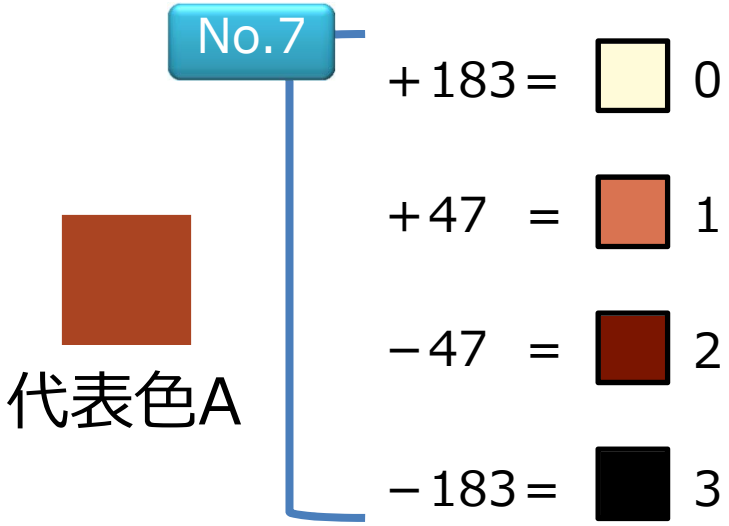
Diff Bit = **1 ビット**

復元



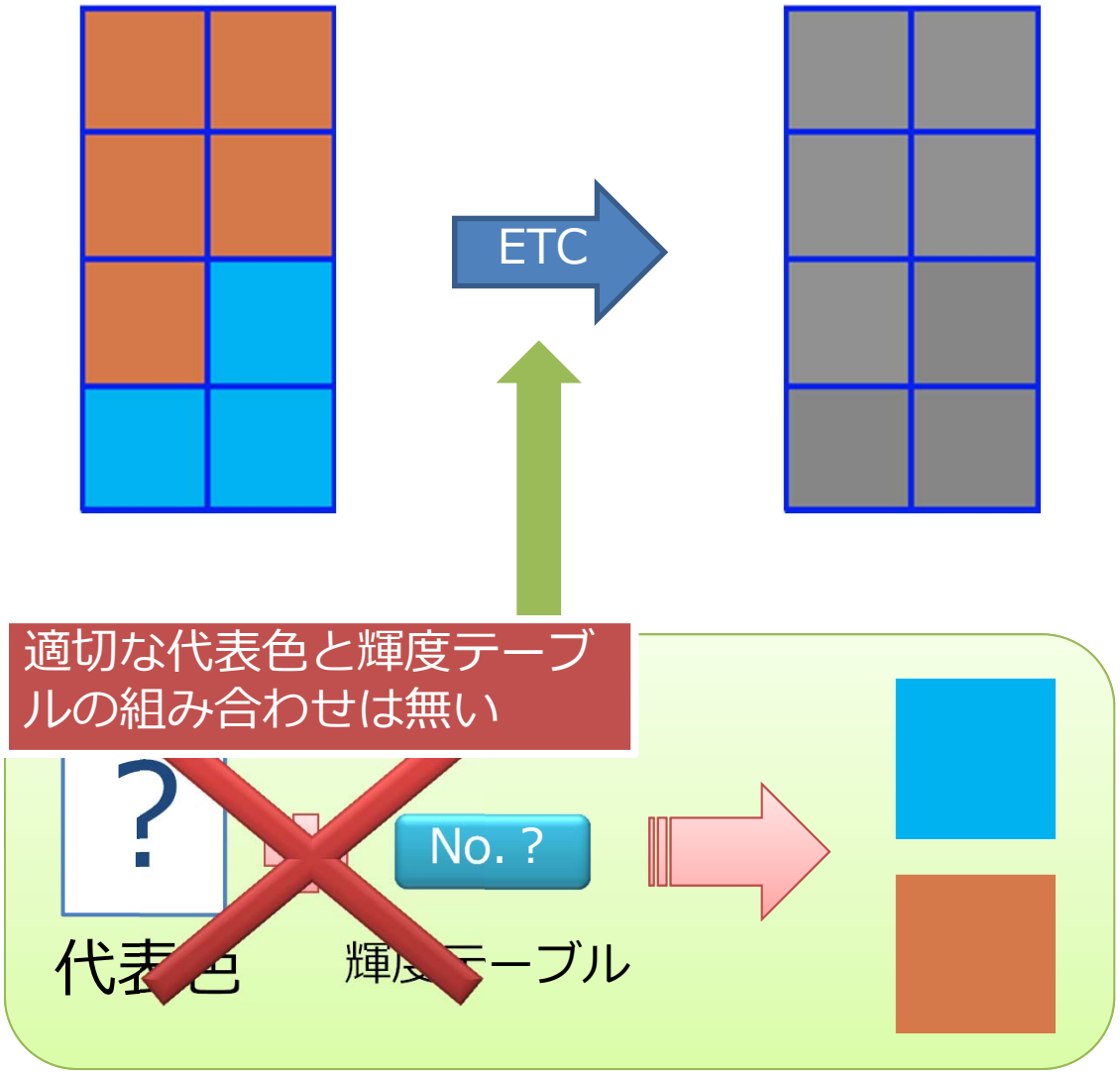
1ブロックあたり 64 ビット

ETCのここがポイント！！

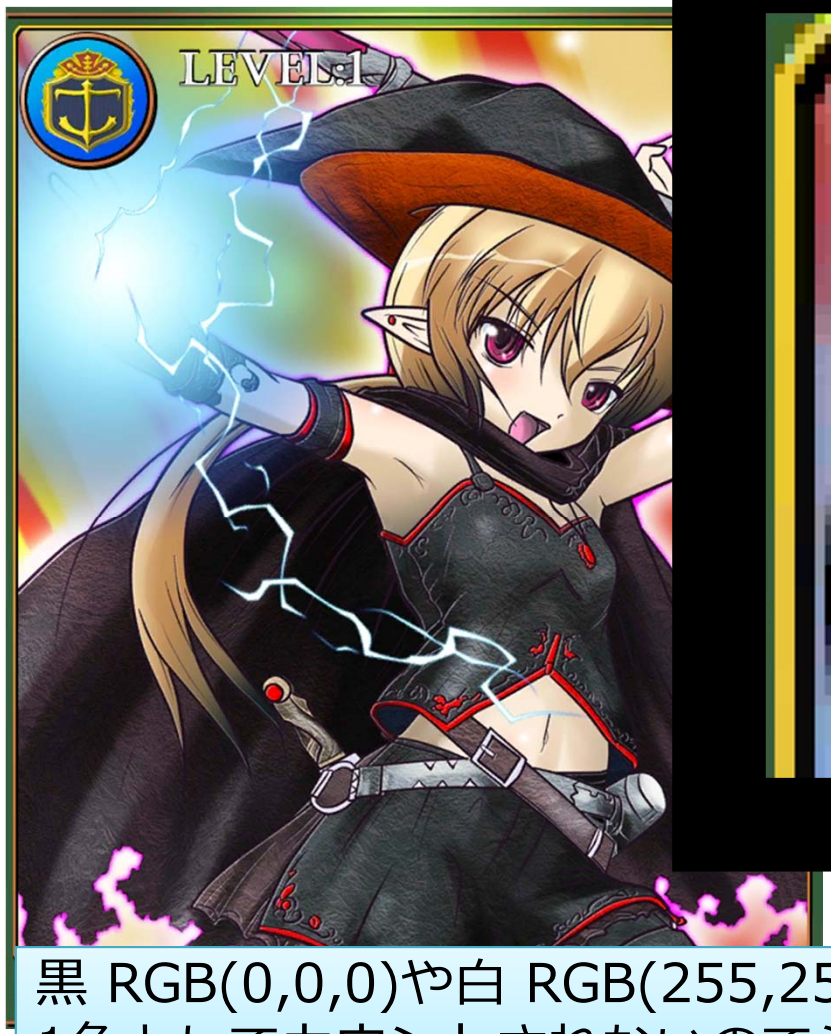


代表色から4色作る時に、輝度の変化しかできない

ETCのブロックノイズの原因



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。

ETCの対策

その2
アルファマスクを作る場合に

ETCでのマスク作成注意点

- ETCでアルファチャンネル抜きを行いたい
- 別な画像にアルファチャンネル画像を作成する
- シェーダーなりで重ねる

絵柄をETCで保存

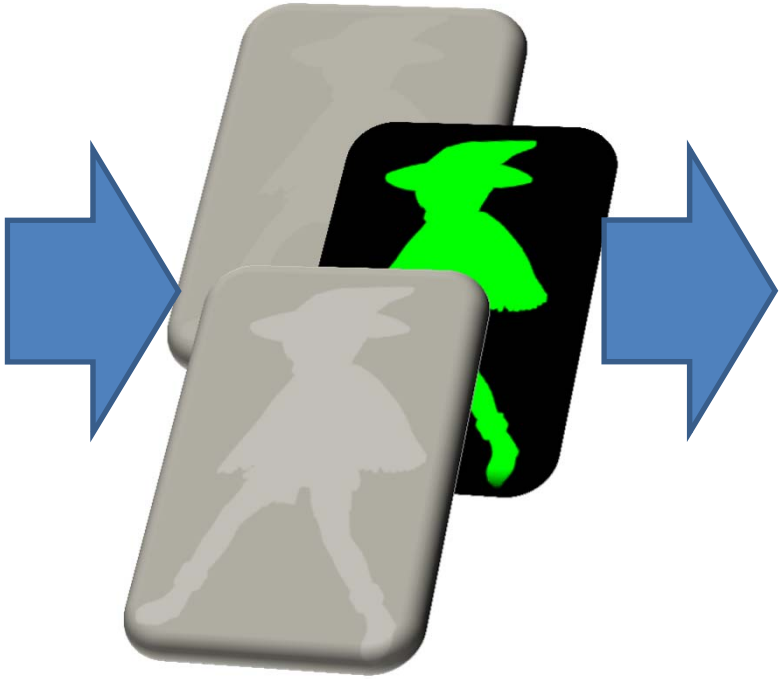
マスクを作りたい

アルファで抜ける

チャンネル作成の注意点

オリジナル　画像絵柄

RGBチャンネル



マスク ETC

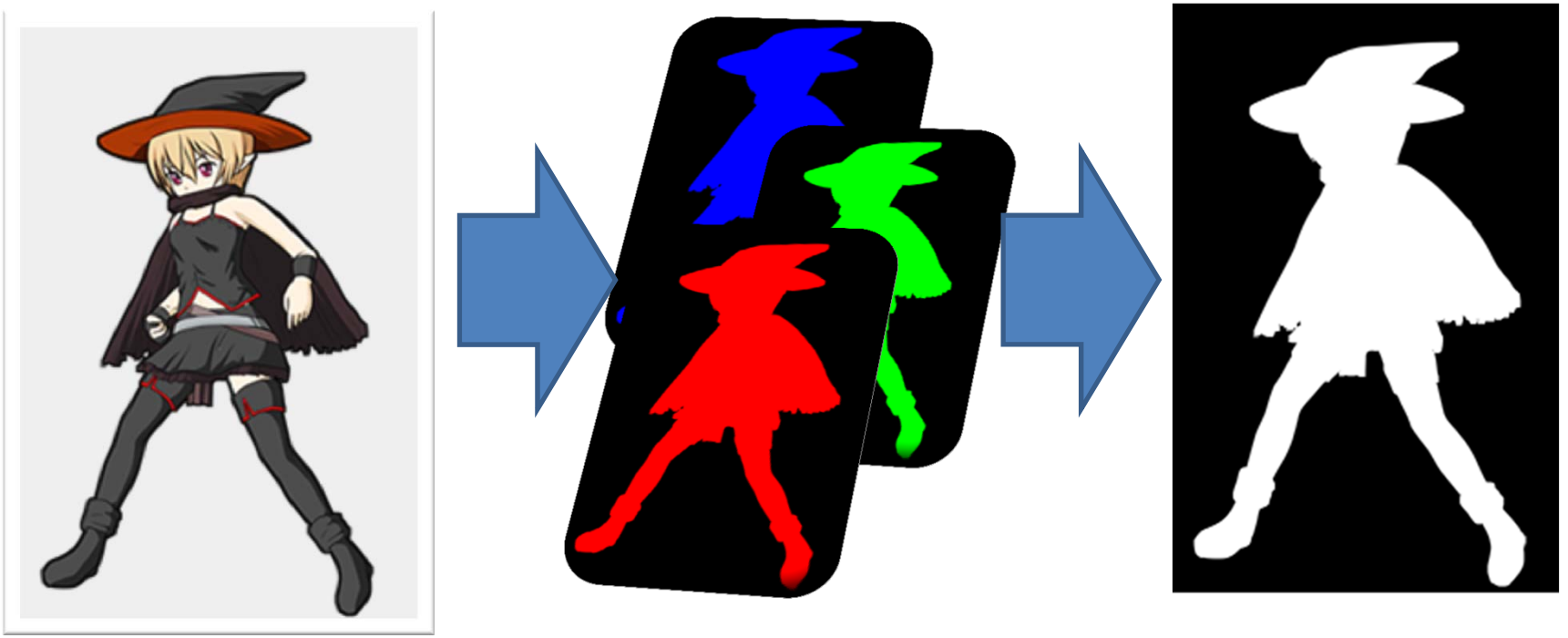
１つだけのチャンネルへのコピーだと劣化が増す。

チャンネル作成の注意点 (2)

オリジナル　画像絵柄

RGBチャンネル

マスク ETC



RGBチャンネルへアルファチャンネルの値を
コピーした方が品質が良い
アルファ値として使うのはどのチャンネルでも良い。

ETCにおける傾向と対策まとめ

黒や白でない2色の境界付近でブロックノイズが出やすい

- 可能なら黒や白で境界線を描く

ETCでアルファチャンネルマスクを作る時の注意点

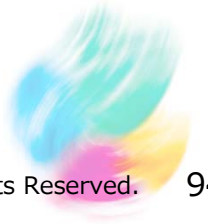
- RGBどれか1チャンネルではなく、
  3チャンネル全てを使ってマスクを作る方がよい

16bitカラーの検討

16bitダイレクトカラーの検討

検討シチュエーション

- 絵柄が圧縮由来の劣化に耐えられない
- 32bppよりを容量を小さくしたい
- 環境依存を小さくしたい



アルファ値ありでの比較

圧縮形式	bpp	比率
32bit ダイレクトカラー	32bpp	1
16bit ダイレクトカラー	16bpp	1/2
DXTC	8bpp	1/4
PVRTC 4bpp	4bpp	1/8
ETC + マスク	8bpp (4bpp + 4bpp)	1/4

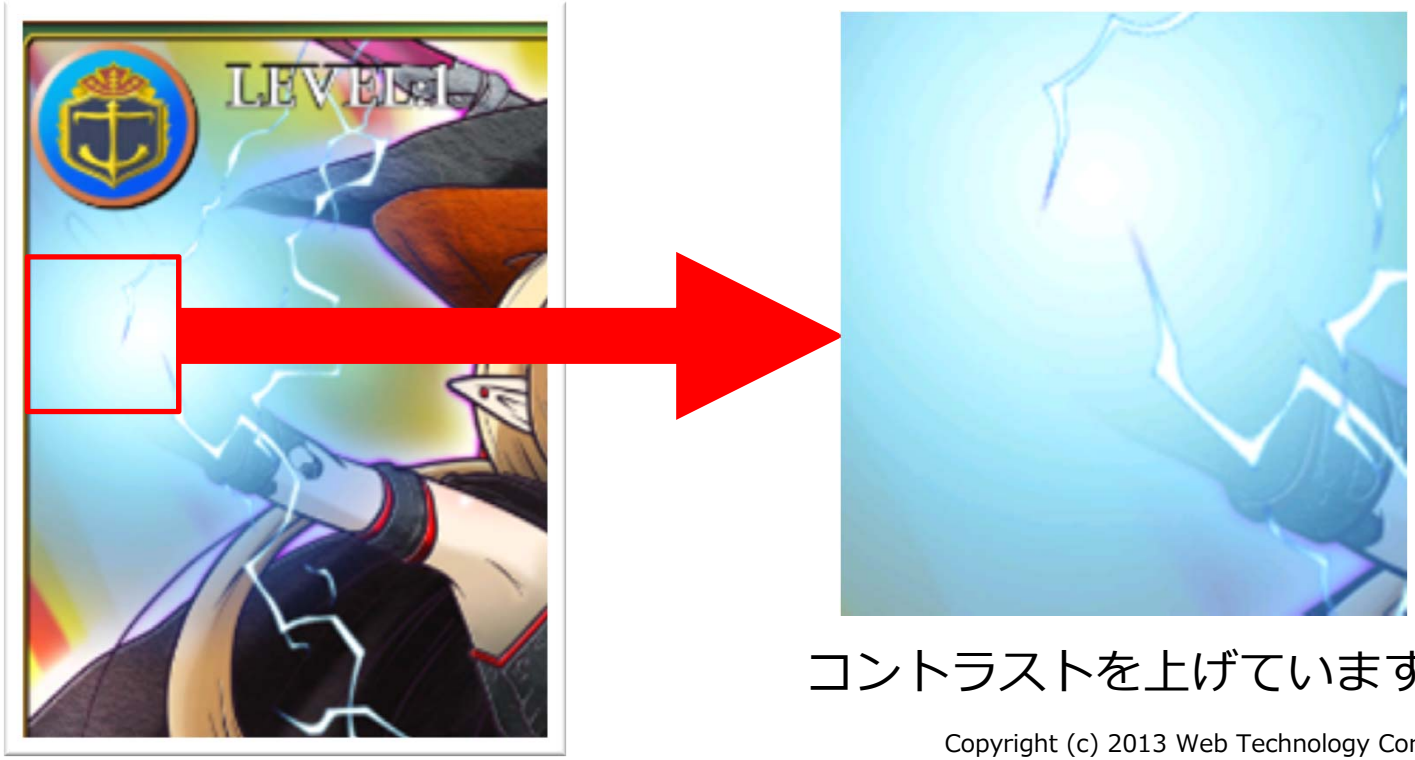
形式の使い分け

フォーマット	カラー値	アルファ値
RGB565	16bit	0bit
RGB5551	15bit	1bit (抜き)
RGBA4444	12bit	4bit (16階調)

カラー値のbitが少ないほど
カラー部分のクオリティは低下します

16bitダイレクトカラーも検討する

- 圧縮テクスチャと比べると補完関係の性質
  - 階調表現が苦手
  - ブロックノイズは出ない
    - ただし「マッハバンド」が出る場合がある。



コントラストを上げています。

マッハバンド対策

RGB565

RGB565 誤差拡散　25%

最後に

手戻りを最小限に

手戻りを最小限に

- 圧縮テクスチャの性質を知る。
- 出力する圧縮フォーマットが何か？を計画する。
  - 出力するツールも吟味して下さい。

- レイヤー分けされたデータの保持
  - ブロックノイズ低減などのため
  - スポットで色補正できた方が都合が良い
  - 枠線もレイヤーになってた方が都合が良い

- 圧縮では無理な場合を見切って16bitもしくは32bitにする
  - フォーマットのメリットデメリットも把握すること
  - バランスが大事です。

ご清聴ありがとうございました。

このファイルは、以下からダウンロードできます
http://www.webtech.co.jp/blog/developer-news/5857/