

# 知っておきたい、画像データ 圧縮の仕組みと特長 ～ETCとPVRTC～

この資料は、[ABC 2013 Autumn](#) の「[デザイン・開発トラック](#)」で発表した資料です。  
<http://www.webtech.co.jp/blog/products/optpix-imesta/6136/> で配布しています。

株式会社ウェブテクノロジー  
R&Dグループ マネージャ  
小野知之  
2013.10.20

## 本講演の目的

- アプリ開発者に向けて、アプリ内で使用可能な圧縮画像フォーマットの特徴を解説します。どんな場合にどのような問題が発生するか、またその対策方法について、理解を深めます。

## 2D用途向け圧縮画像形式の復習

- ABC 2013 Spring の内容を簡単に復習します
- JPEG, PNGなどの圧縮画像の特徴についておさらいします

## 3Dテクスチャ圧縮の解説

- 3Dテクスチャ圧縮の特徴と注意点
- 圧縮アルゴリズム概略



# ウェブテクノロジーのご紹介

WebTechnology®  
OPTPIX imésta

OPTPIX  
imésta



OPTPIX  
SpriteStudio®



OPTPicture



OPTPIX®  
ライブ壁紙メーカー

創業23年目になる、  
池袋のソフト開発会社です。

# Clear PVRTC

元画像



OPTPIX  
imésta7  
for Mobile & Social  
**Clear  
PVRTC**



標準  
**PVRTC**



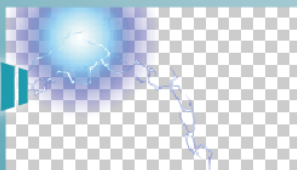
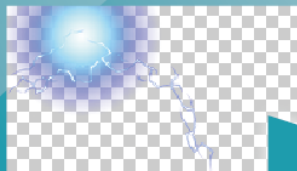
**PNG32**

15.1KB

**PNG8**

6.96KB

**46%**



**PNG24**

223KB

**PNG24**

140KB

**63%**







## SAMURAI SOULS

©2013 CJ Internet Japan Corp.  
 NOW PRODUCTION Co., Ltd. All Rights Reserved.



## Gather of Dragons

©2013 BANDAI NAMCO Games Inc.  
 All rights reserved.



## Mushroom Garden Deluxe

©2012 Beeworks Co.Ltd  
 ©2012 SUCCESS Corporation. All rights reserved.



## 本講演の目的

- アプリ開発者に向けて、アプリ内で使用可能な圧縮画像フォーマットの特徴を解説します。どんな場合にどんな圧縮画像フォーマットが向いているのかが理解できるようになります。

## 2D用途向け圧縮画像形式の復習

- ABC 2013 Spring の内容を簡単に復習します
- JPEG, PNGなどの圧縮画像の特徴についておさらいします

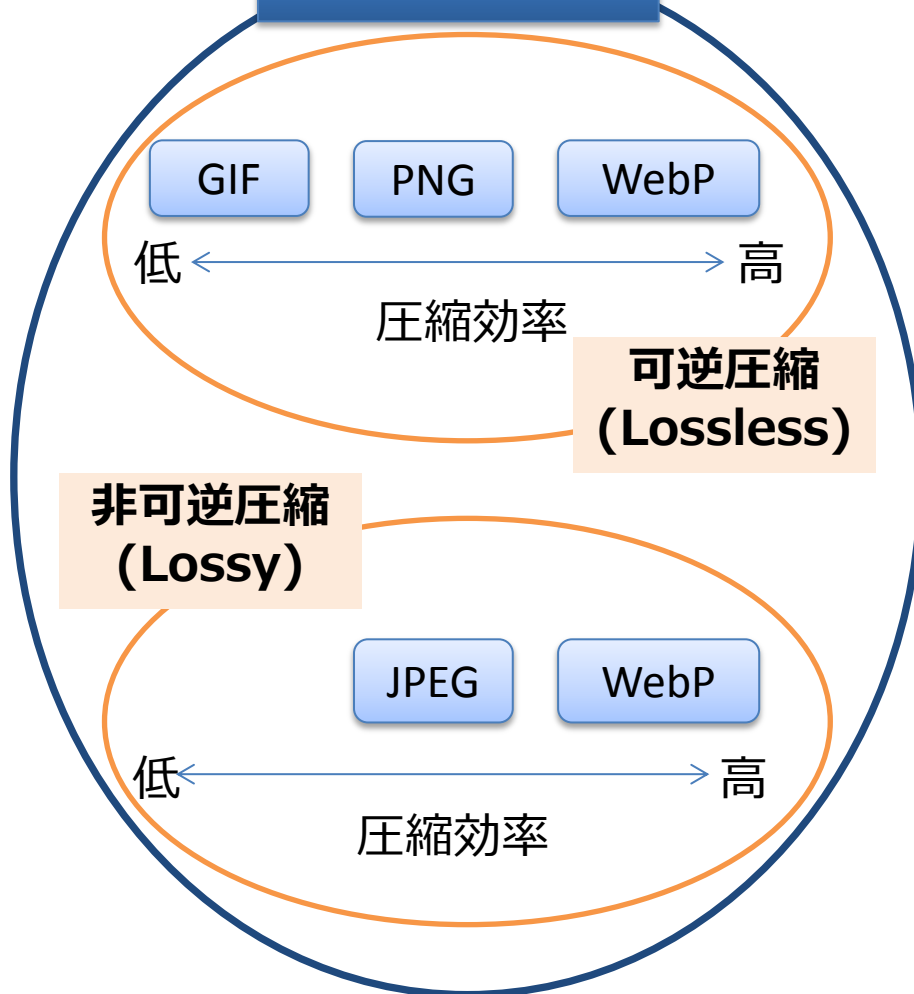
## 3Dテクスチャ圧縮の解説

- 3Dテクスチャ圧縮の特徴と注意点
- 圧縮アルゴリズム概略

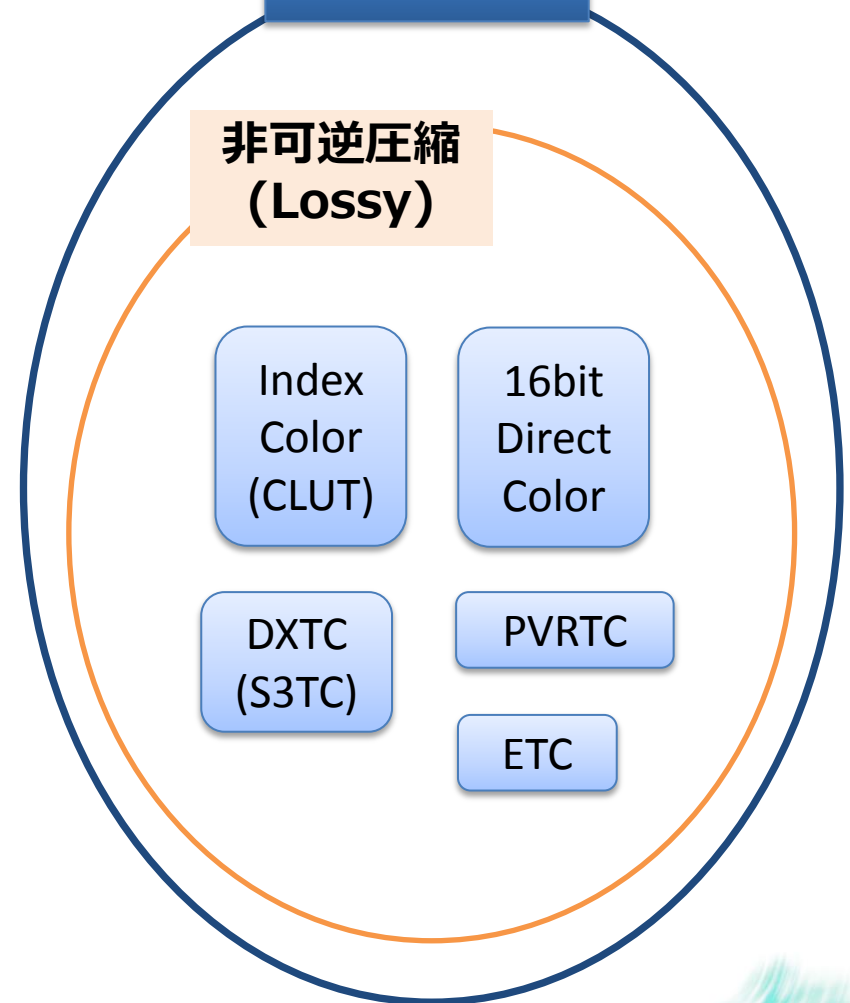


# 画像圧縮の分類

## 可変圧縮率



## 固定圧縮率



\* プラットフォーム・機種によって使用可否があります





## ■ キーポイント

- 写真などの自然画の圧縮に向いている
- GUIパーツのような、くっきりした文字やアイコン主体の画像の圧縮には向いていない \*
- 画面キャプチャやバナーにも向いていない \*
- アルファ(透過度)を持ってない

\* 圧縮効率という意味で





- GUIパーツのような文字・アイコン主体のくっきりした画像に向いている \*
- バナーのような、写真と文字で構成された画像は、PNG8が品質と容量のバランスが一番良い \*
- 階調アルファを持てる(美しいGUIパーツには必須)

\* 圧縮効率という意味で



- **アプリ内で利用する意味はあまりない**  
(PNGを使う方が良い)
- ブラウザベースのアプリでも注意が必要。アニメーション可能だが、Androidのブラウザでは機種によって正しく表示できない場合がある  
(Androidの一部機種で静止GIFも正しく表示されないという報告もある)



- **Android 4.0以上ではアプリから使用可能。  
アプリ内で閉じている画像に使う場合は  
メリットあり**
- **非可逆圧縮はAndroid 3.1以降から使える  
(アンドキュメント)**
- Googleが開発した静止画フォーマット。  
**最後発なので、JPEGとPNGを合わせて高性能化したような規格**
- ブラウザでは、Chromeなど一部が対応するのみ。**普及しているとは言いがたい**
- JPEGのような非可逆圧縮、PNGのような可逆圧縮ともサポート
- 非可逆圧縮では圧縮率可変
- 階調アルファを持てる(非可逆圧縮においても)
- JPEG,PNGより圧縮率が高い



# 2D用途向け圧縮画像形式のまとめ

## JPEG

- 写真などの自然画の圧縮に向いている
- メニューのような、くっきりした文字やアイコン主体の画像の圧縮には向いていない \*
- アルファ(透過度)は持てない

## PNG

- メニューのような文字・アイコン主体のくっきりした画像に向いている \*
- バナーのような、写真と文字で構成された画像は、PNG8が品質と容量のバランスが一番良い \*
- 階調アルファを持てる

## GIF

- アプリ内で利用する意味はあまりない  
(PNGを使う方が良い)

## WebP

- Android 4.0以上ではアプリ内で使用可能。  
アプリ内で閉じている画像に使う場合はメリットあり

\* 圧縮効率という意味で





## 参考 - JPEG の弱点

JPEG

■ 880x200ピクセル JPEG 74,708バイト



赤い部分や周囲に  
ノイズが目立つ



## 参考 - JPEG の弱点

PNG

- 880x200ピクセル 128色 PNG 74,484バイト  
OPTPIX imésta で減色



JPEGとほぼ  
同一サイズなのに  
クッキリしている



## 本講演の目的

- アプリ開発者に向けて、アプリ内で使用可能な圧縮画像フォーマットの特徴を解説します。どんな場合にどんな圧縮画像フォーマットが向いているのかが理解できるようになります。

## 2D用途向け圧縮画像形式の復習

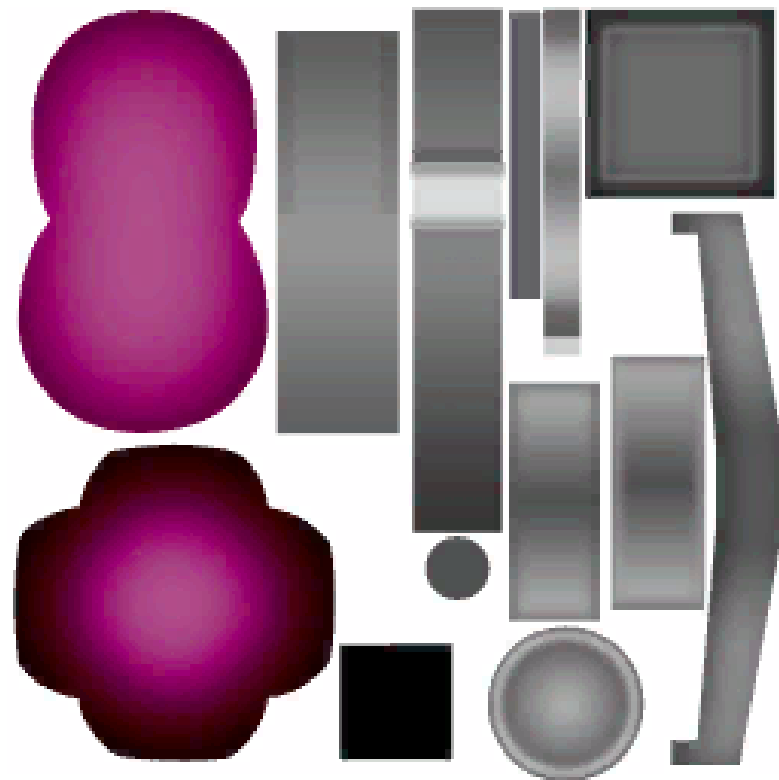
- ABC 2013 Spring の内容を簡単に復習します
- JPEG, PNGなどの圧縮画像の特徴についておさらいします

## 3Dテクスチャ圧縮の解説

- 3Dテクスチャ圧縮の特徴と注意点
- 圧縮アルゴリズム概略



# 3Dテクスチャとは





# 主な3Dテクスチャ

圧縮テクスチャの種類	変換元画像	圧縮率
インデックス カラー	RGB 24bpp	1/3(256色)*1, 1/6(16色)*2
インデックス カラー(アルファ付)	RGBA 32bpp	1/4(256色)*3, 1/8(16色) *4
DXT1(S3TC)	RGB 24bpp	1/6
DXT1(S3TC)	RGBA 32bpp	1/8
DXT2~5	RGBA 32bpp	1/4
PVRTC	RGB 24bpp	1/6(4bpp) 1/12(2bpp)
PVRTC	RGBA 32bpp	1/8(4bpp) 1/16(2bpp)
ETC	RGB 24bpp	1/6

\*1 ピクセルデータの他に、3x256バイトのパレット情報が必要

\*2 ピクセルデータの他に、3x16バイトのパレット情報が必要

\*3 ピクセルデータの他に、4x256バイトのパレット情報が必要

\*4 ピクセルデータの他に、4x16バイトのパレット情報が必要



# 主な3Dハードと主な対応テクスチャ

テクスチャの種類	1995～2000年頃	2000～2005年頃	2005～2010年頃	2010～2013年頃
インデックスカラー	PlayStation			
アルファ付インデックスカラー		PlayStation 2		
DXTC (S3TC)		GAMECUBE Xbox PC(DirextX)	PlayStation 3 PSP Wii Xbox360 PC(DirextX)	Wii U PC(DirextX) 一部のAndroid 端末
PVRTC			iPhone	iPhone/iPad PS Vita 一部のAndroid 端末
ETC				全てのAndroid端末 Nintendo 3DS
ATITC				多くのAndroid端末
BC				PlayStation 4 Xbox One PC(DirextX)

\* 各ハードウェアの発売日を表すものではありません

\* PlayStation ®、GAMECUBE ®、NINTENDO ®3DS、Wii ®、Xbox ®は各社の商標です

\* 事情により一部推定、および未記載の情報が 있습니다



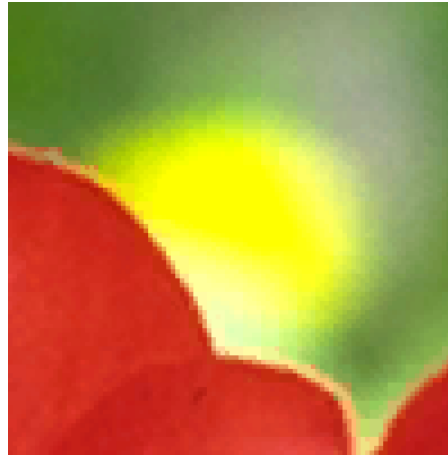
# 圧縮テクスチャの画質比較

自然画のような画像では、ほとんど劣化が目立たない

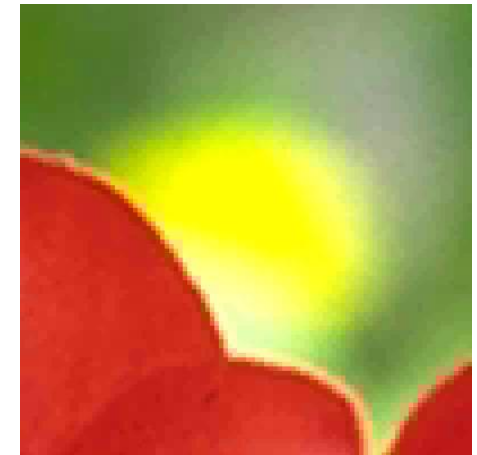
元画像



DXT1画像



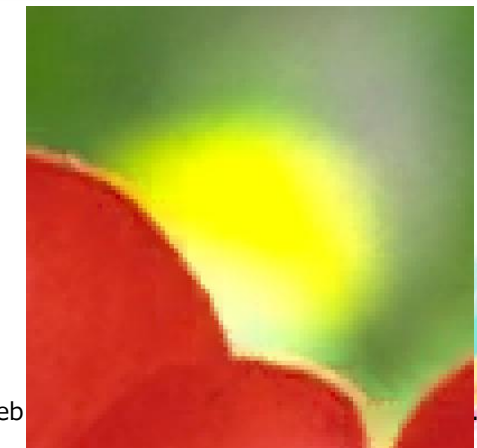
ETC



PVRTC (4bpp) 画像



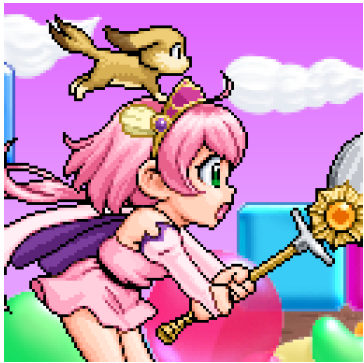
PVRTC (2bpp) 画像



# 圧縮テクスチャの画質比較

アニメ調やUIパーツなどの、クッキリした画像では劣化が目立つ

元画像



DXT1画像



ETC



PVRTC (4bpp) 画像



PVRTC (2bpp) 画像





## DirectX Texture Compression

S3TC (S3 Texture Compression) の拡張

フォーマット: DXT1, DXT2, DXT3, DXT4, DXT5

圧縮単位: 4 × 4ピクセル



# DXTCの画質劣化の例

元画像

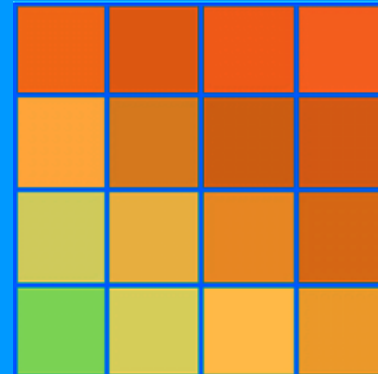


DXTC圧縮後



# DXTCの圧縮の仕組み (1)

4 × 4 ピクセルを圧縮の最小単位 (1 ブロック) とする

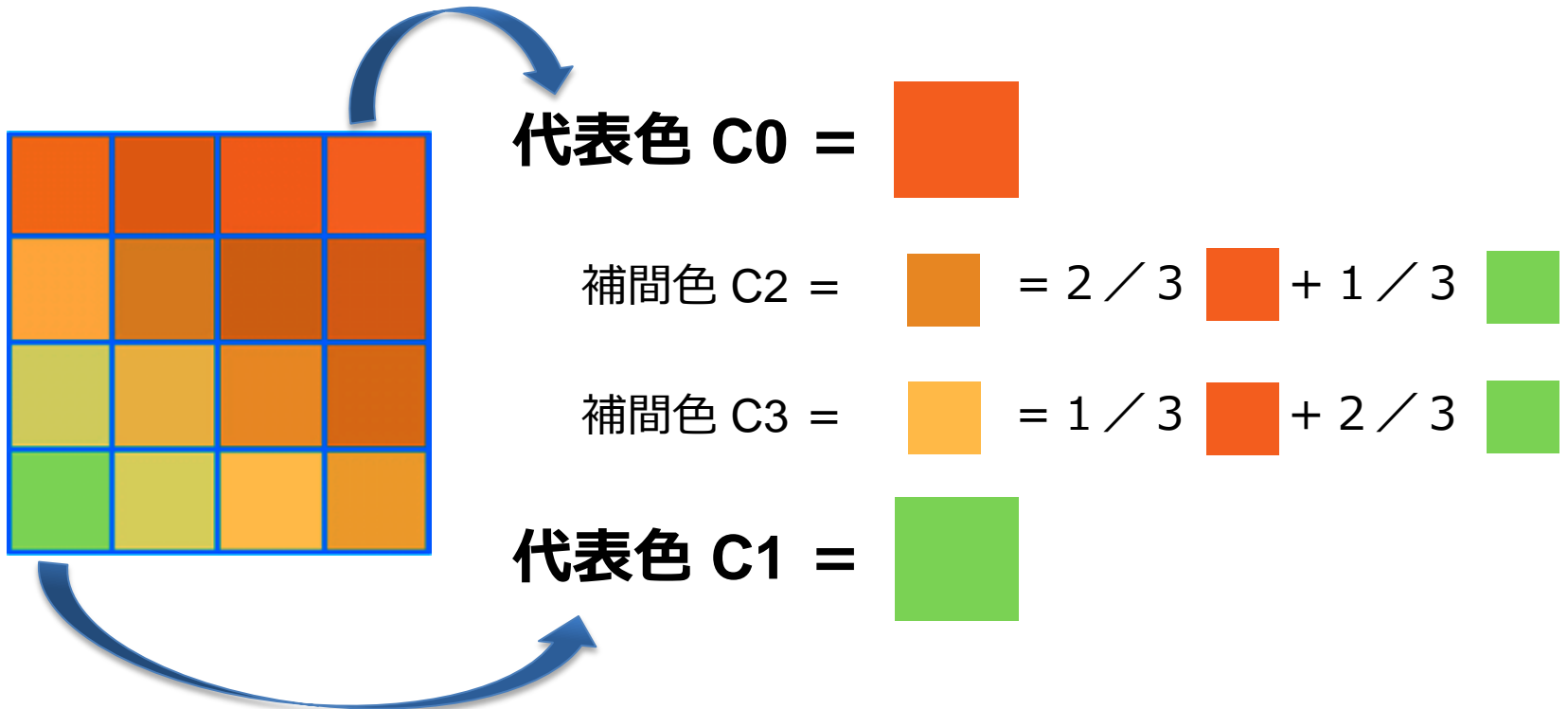


1 ブロック  
(4 × 4 ピクセル)



## DXTCの圧縮の仕組み (2)

代表色を2色選び、それらの補間色を求める



\* ブロックに無い色でも代表色に使用できます





# DXTCの圧縮の仕組み (3)

元のピクセルの色を置き換える

代表色 C0 =

0

補間色 C2 =

2

補間色 C3 =

3

代表色 C1 =

1



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



色にインデックス番号(0~3)を割り当て  
16個の配列データとして保存する



## インデックスと二つの色の情報のみになる

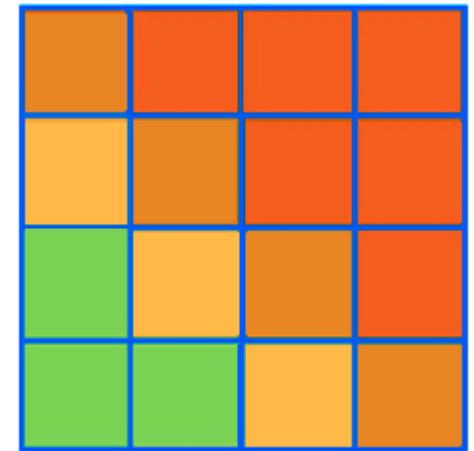
2	0	0	0
3	2	0	0
1	3	2	0
1	1	3	2

インデックス2ビット  
× 16ピクセル  
= 32 ビット

代表色 C0 =   
代表色 C1 = 

16ビット (RGB565)  
× 2色  
= 32 ビット

復元

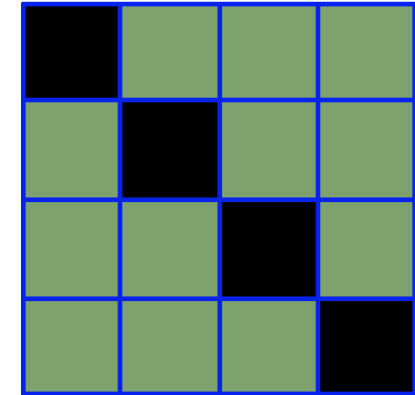
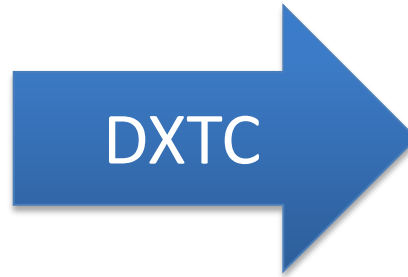
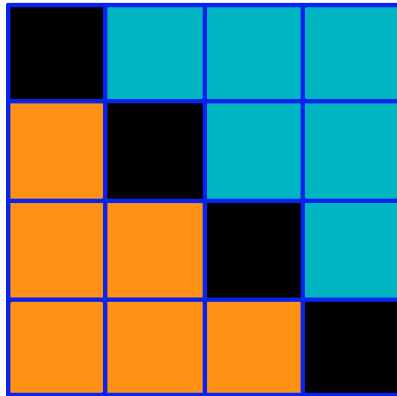


1ブロックあたり 64 ビット

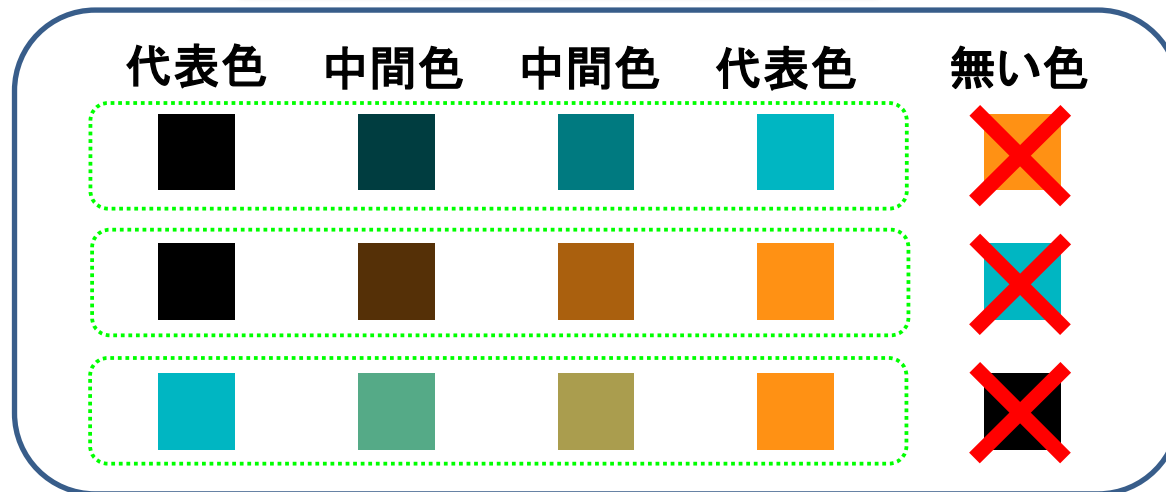
\* DXT2~5のアルファチャンネルはこれと異なります。



## DXTCで表現できないパターン



3色表現できる組み  
合わせが無い



# DXTCの画質劣化の例

もう一度！

元画像



DXTC圧縮後



## 補足 - DXT1の補間モード

## 代表色の順序で補間方法が変わる

代表色  $C_0 > C_1$ 

C0 = #F35D1E

C1 = #7AD254


**補間2色**

代表色 C0 =



補間色 C2 =



= 2 / 3



+ 1 / 3



補間色 C3 =



= 1 / 3



+ 2 / 3



代表色 C1 =

代表色  $C_0 \leq C_1$ 

C0 = #7AD254

C1 = #F35D1E


**補間1色  
+透過色**

代表色 C0 =



補間色 C2 =



= 1 / 2



+ 1 / 2



代表色 C1 =



透過色 C3 =



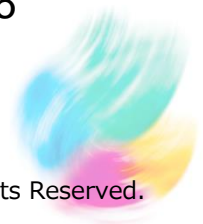


# 補足 - DXTCのフォーマット

名前	RGBチャンネル	アルファチャンネル	圧縮率*	DirectX 10以降での名前
DXT1 (S3TC)	代表 2 値 + 補間 2 値 or 1 値	未使用 or 1ビット(抜き)	1/8**	BC1
DXT2	代表 2 値 + 補間 2 値 (アルファ値乗算)	4 ビット(16階調) 固定アルファ値	1/4	
DXT3	代表 2 値 + 補間 2 値	4 ビット(16階調) 固定アルファ値	1/4	BC2
DXT4	代表 2 値 + 補間 2 値 (アルファ値乗算)	補間(8階調) アルファ値	1/4	
DXT5	代表 2 値 + 補間 2 値	補間(8階調) アルファ値	1/4	BC3

\*32ビット非圧縮ARGB画像からの圧縮率

\*\*24ビット非圧縮RGB画像からの圧縮率は1/6



# DXTCにおける傾向と対策まとめ

## 黒や白の輪郭線周辺でブロックノイズが出やすい

- 可能なら輪郭線を太めに調整する

## フォーマット形式によりアルファ表現に差が出る

- 画像毎に適切なフォーマットを選ぶこと  
アルファなし or 1bitアルファならDXT1  
多階調アルファならDXT5が無難



## PowerVR Texture Compression

フォーマット: 4bpp, 2bpp

圧縮単位:  $4 \times 4$ 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の画質劣化の例(4bpp)

## 元画像



## PVRTC(4bpp)圧縮後



# PVRTCの画質劣化の例(2bpp)

元画像

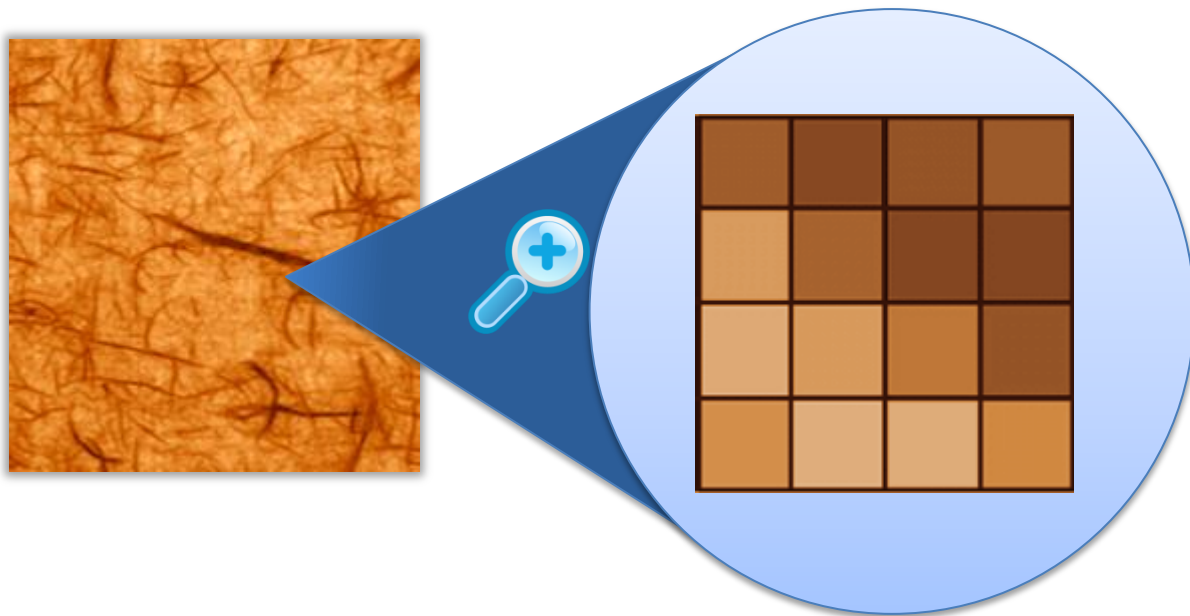


PVRTC(2bpp)圧縮後



# PVRTCの圧縮の仕組み (1)

4 × 4 ピクセルを圧縮の最小単位(1ブロック)とする



1 ブロック  
(4 × 4 ピクセル)

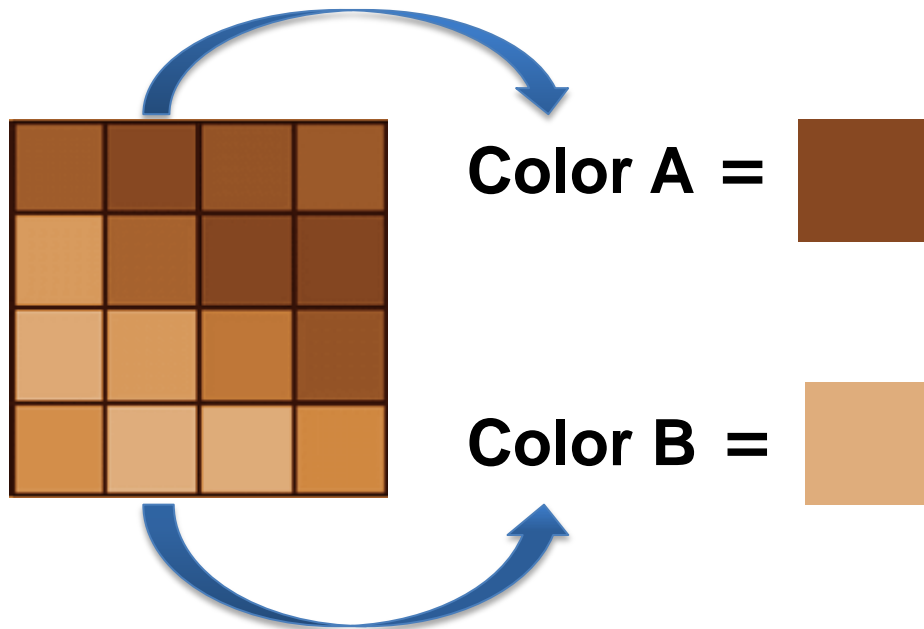
\* 4bppの例で解説しています。





## PVRTCの圧縮の仕組み (2)

### ブロックごとに2色を決める

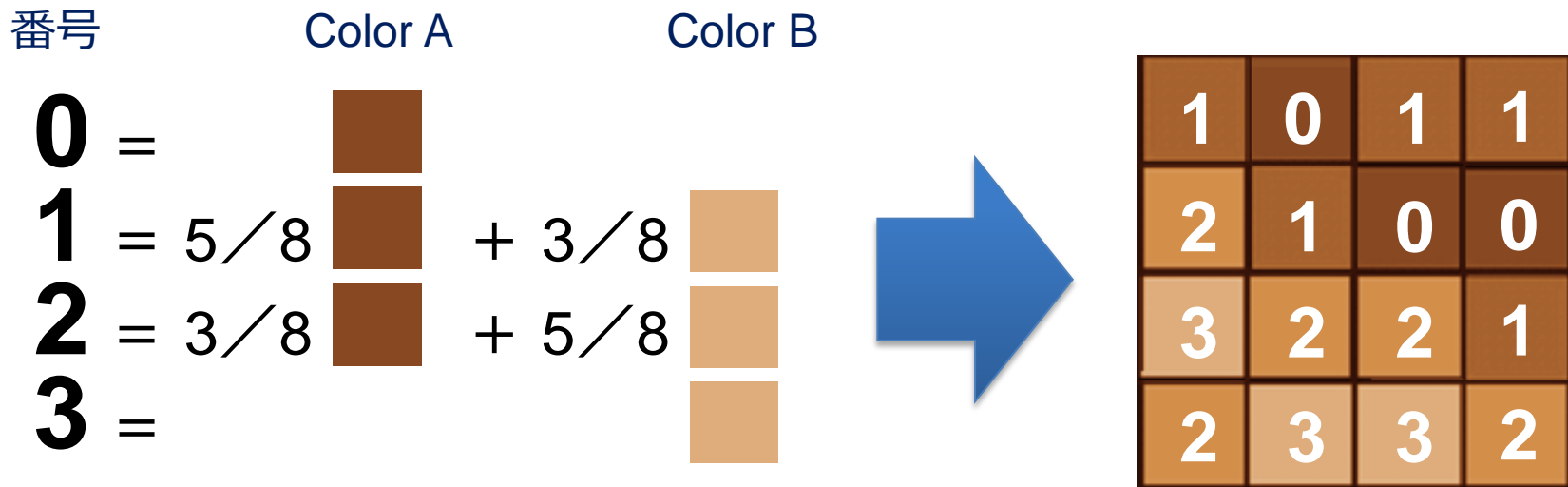


- \* ブロックに無い色でも指定できます
- \* 実際には複雑な方法で2色を選ぶ必要があります (詳細は後述)



# PVRTCの圧縮の仕組み (3)

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 (4bpp) のデータ構造

## 2色とModulation Dataの情報になる

\* 4bppの例で解説しています。

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

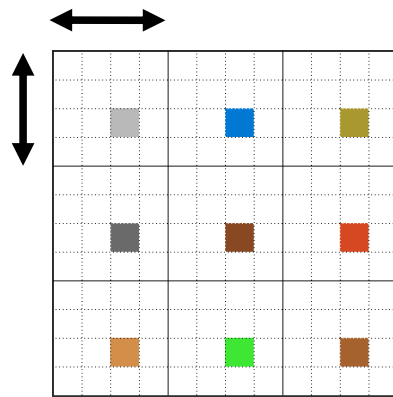
1ブロックあたり 64 ビット



## PVRTCの復元 (1)

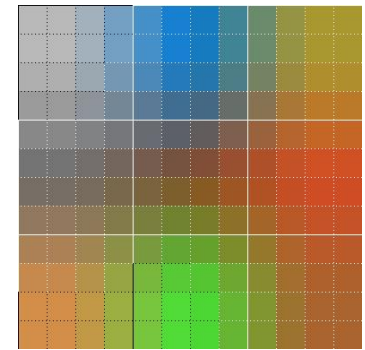
Color A, B をそれぞれ、周囲のブロックと補間する

1ブロック(4×4)



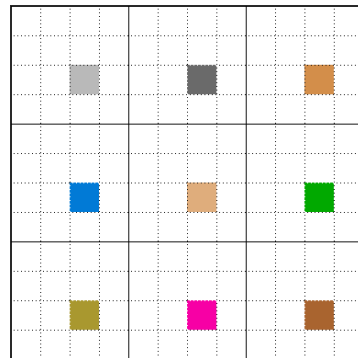
Color A

補間

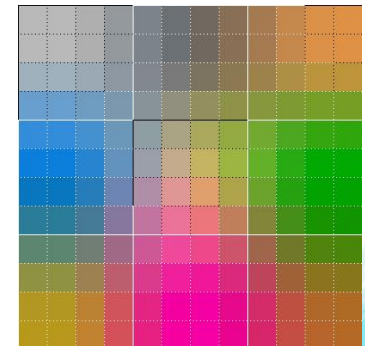


4×4ピクセルを補間色で埋める

Color B

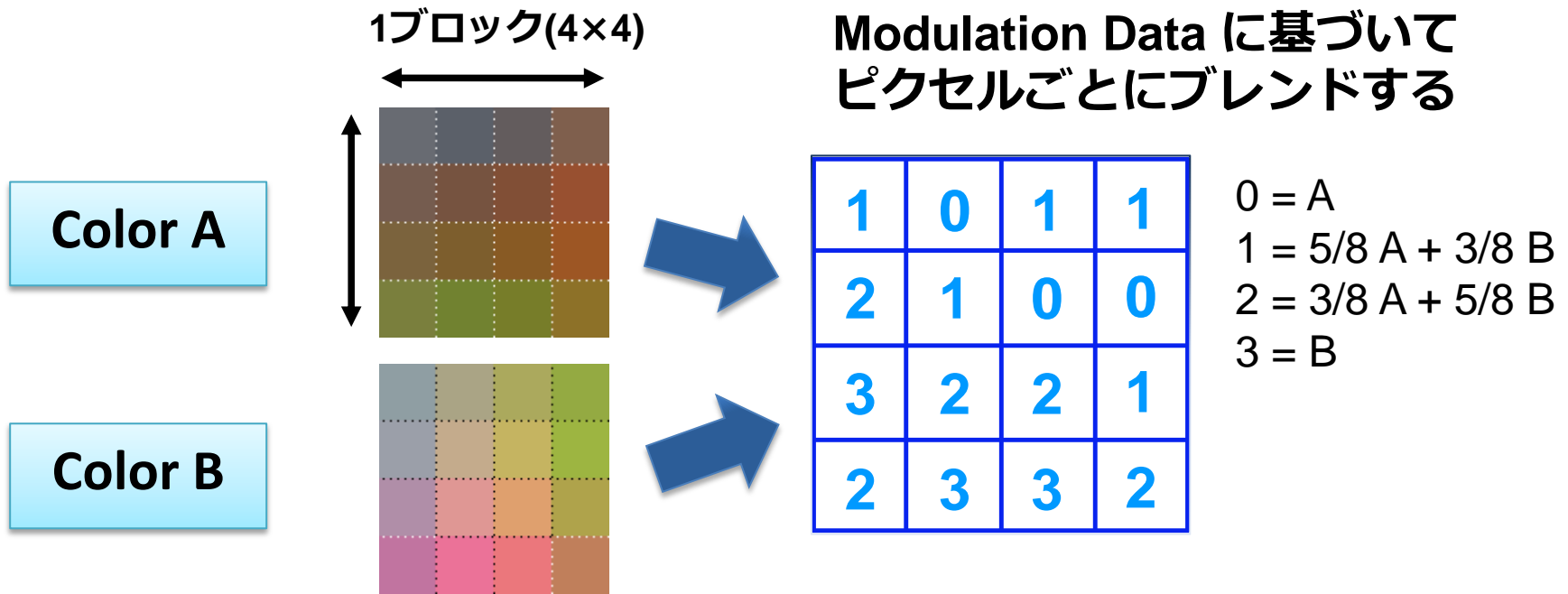


補間



# PVRTCの復元 (2)

## 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の画質劣化の例(4bpp)

もう一度！

元画像



PVRTC(4bpp)圧縮後



# PVRTCの画質劣化の例(2bpp)

もう一度！

元画像



PVRTC(2bpp)圧縮後











## 補足 - PVRTC(4bpp)のモード

## Mode Bit の値でブレンド方法が変わる








Mode Bit = 1

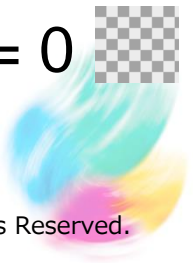
➔ 4段階

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

Mode Bit = 0

➔ 3段階  
+透過

番号	Color A		Color B	
0 =				
1 =		+ 4/8		
2 =		+ 4/8		& Alpha = 0 
3 =				



## 補足 - PVRTC(4bpp)の色形式

## アルファの有無を選択可能

**Color A** = 16 ビットアルファなし: **RGB555**アルファあり: **ARGB3444**

先頭ビットで切り替え

1	Red 5 bit	Green 5 bit	Blue 5 bit
---	--------------	----------------	---------------

0	Alpha 3 bit	Red 4 bit	Green 4 bit	Blue 4 bit
---	----------------	--------------	----------------	---------------

**Color B** = 15 ビットアルファなし: **RGB554**アルファあり: **ARGB3443**

1	Red 5 bit	Green 5 bit	Blue 4 bit
---	--------------	----------------	---------------

0	Alpha 3 bit	Red 4 bit	Green 4 bit	Blue 3 bit
---	----------------	--------------	----------------	---------------

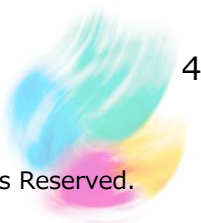


# 補足 - PVRTC(2bpp) の構造

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では、水平方向  
の品質が低下しやすい



展開時にバイリニア補間をかけるので、周囲の色に影響を受ける  
そのためUIパーツのようなクッキリした画像がぼやけやすい

- まわりの影響を受ける部分は、マージンを広め取る
- 可能であればパーツ分割し表示時に重ね合わせる

アルファの表現力が低い

- 可能ならグラデーションなどはアルファを使わずRGBで表現

2bppモードは横方向に劣化しやすい

- 水平方向のクオリティは垂直の半分程度と考えてデザイン



## Ericsson Texture Compression

フォーマット: 1種類のみ

\* アルファに非対応

圧縮単位: 4 × 4ピクセル



# ETCの画質劣化の例

元画像



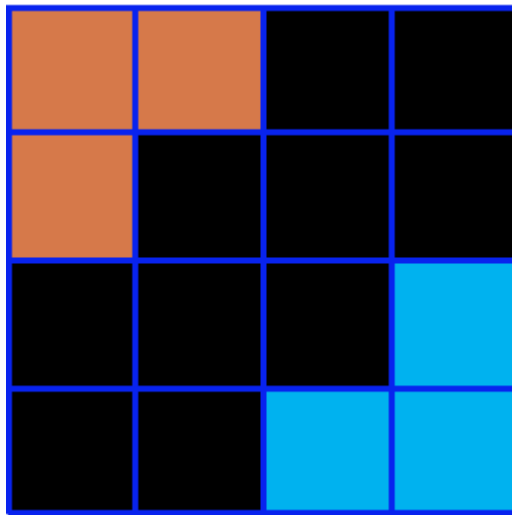
ETC圧縮後



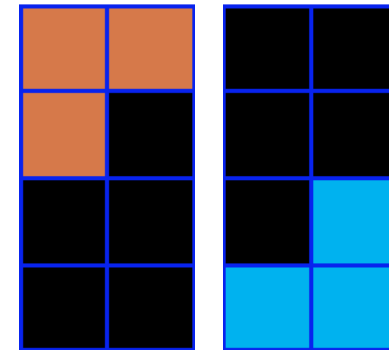
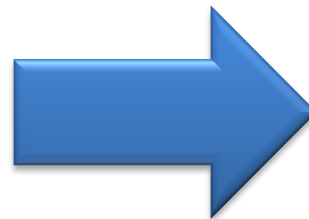


# ETCの圧縮の仕組み (1)

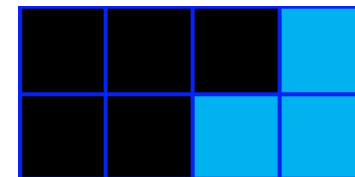
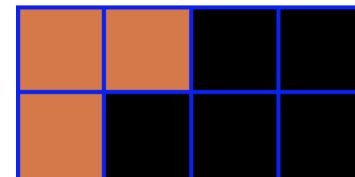
4×4のブロックを2×4のサブブロックに分割する



無圧縮画像



OR

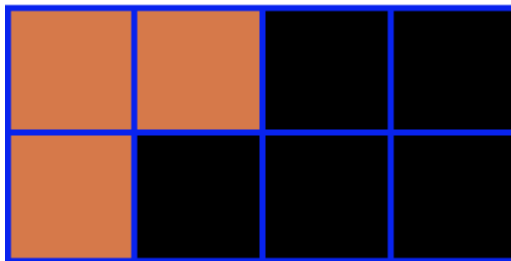


- Individualモードの例で解説しています。  
Differentialモードでは若干異なります。

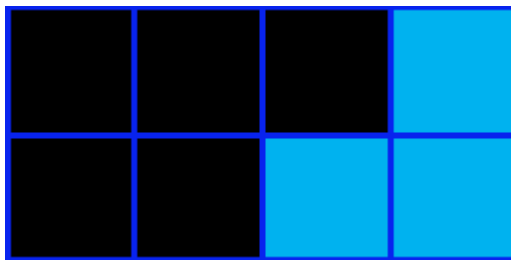


## ETCの圧縮の仕組み (2)

サブブロック毎に代表色を一つずつ決める



  
代表色A



  
代表色B

RGB444



## ETCの圧縮の仕組み (3)

代表色に対する輝度の変換テーブルを選ぶ

変換テーブルの値は固定!!

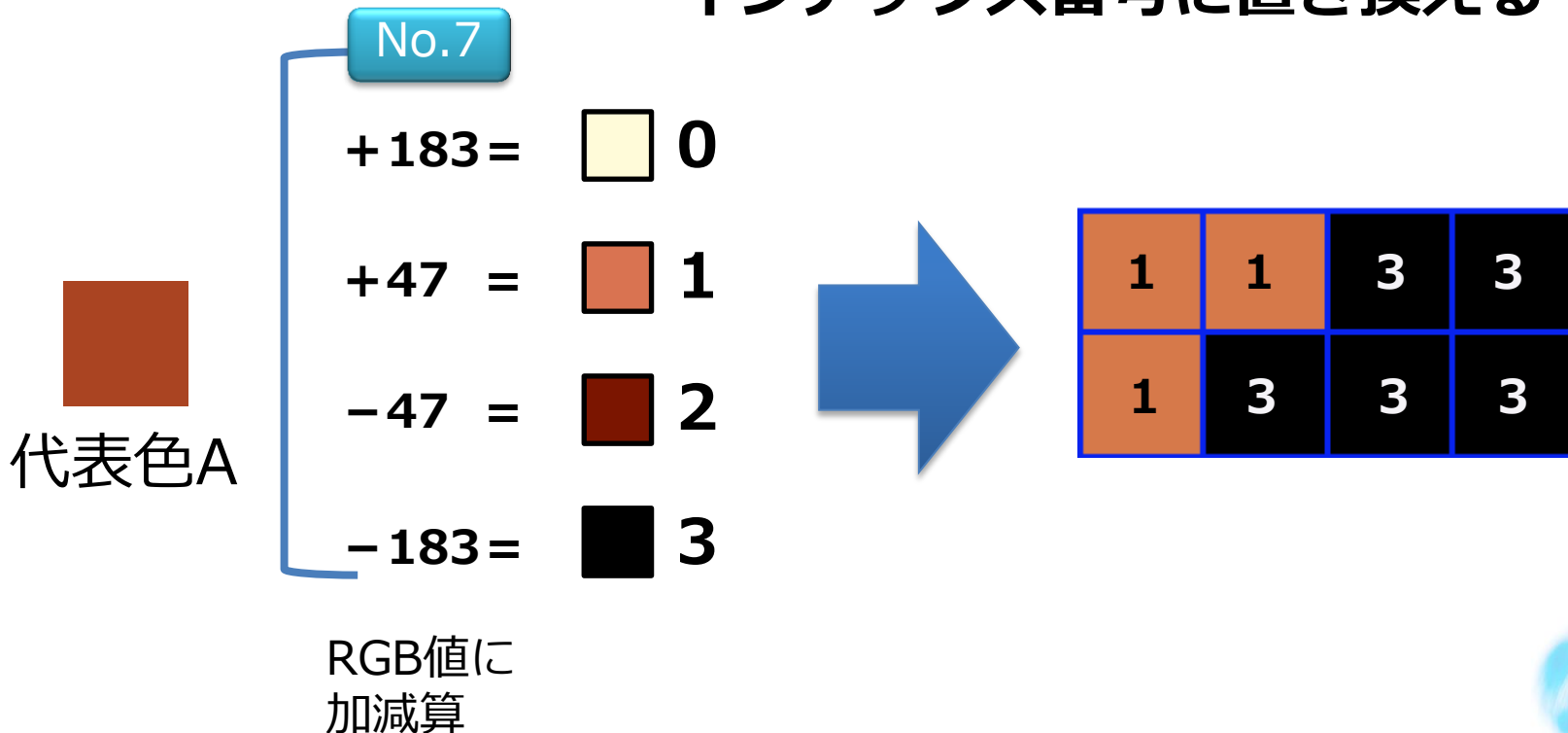
  
代表色A

No.0	-8	-2	2	8
No.1	-17	-5	5	17
No.2	-29	-9	9	29
No.3	-42	-13	13	42
No.4	-60	-18	18	60
No.5	-80	-24	24	80
No.6	-106	-33	33	106
No.7	-183	-47	47	183

## ETCの圧縮の仕組み (4)

各ピクセルの色を  
変換テーブルから生成された4色に置き換える

生成された4色に2ビットの  
インデックス番号に置き換える



# ETCの圧縮の仕組み (5)

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

これらの情報を64bitに格納する

\*Individualモードの例で解説しています。  
Differentialモードでは若干異なります。

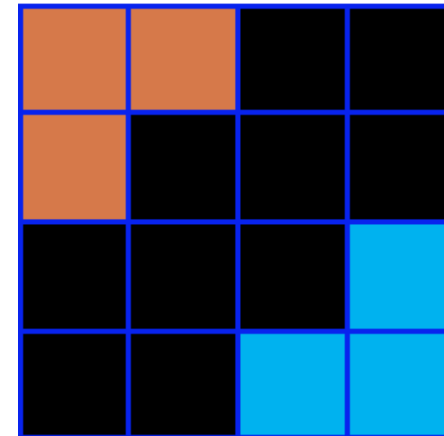
インデックス2ビット  
× 8ピクセル  
× 2  
= **32 ビット**

12ビット  
(RGB444)  
× 2色  
= **24 ビット**

3ビット × 2色  
= **6 ビット**

= **1 ビット**

= **1 ビット**



代表色 A =

代表色 B =

Aの変換テーブル No.7

Bの変換テーブル No.7

Flip Bit

Diff Bit

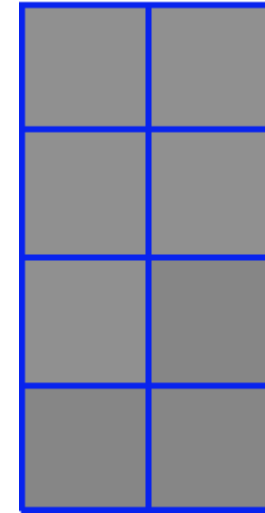
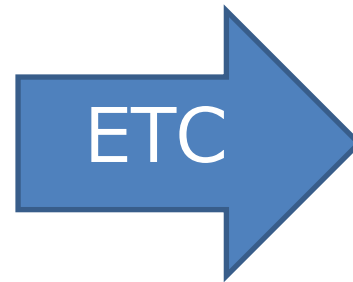
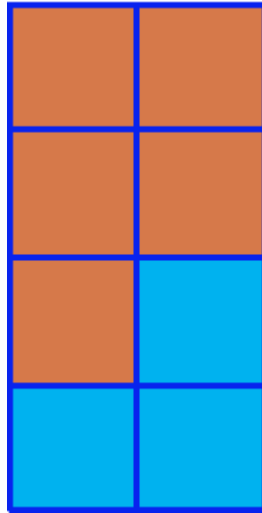
\*縦横の分割方法

\*サブブロック間の関連モード

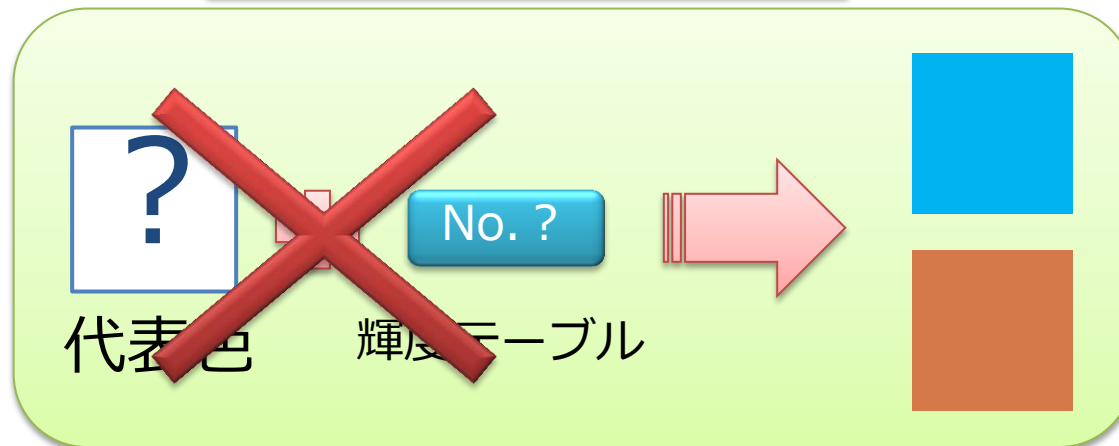
1ブロックあたり 64 ビット



## ETCで表現できないパターン



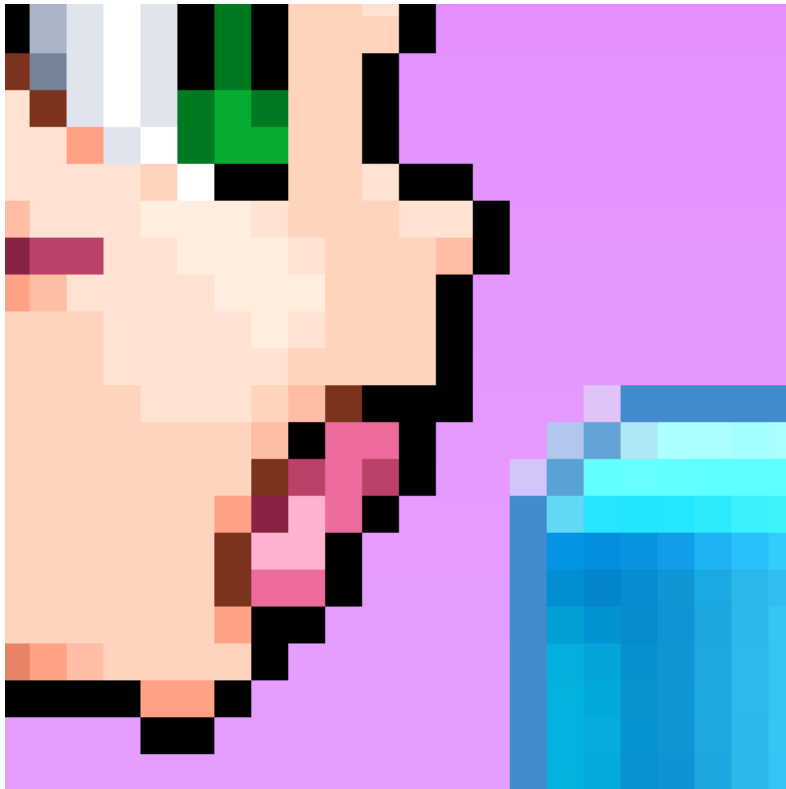
適切な代表色と輝度  
テーブルの組み合わせが無い



# ETCの画質劣化の例

もう一度！

元画像



ETC圧縮後





# 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太めに描く



しかし、  
どうしてもキレイにならないなら…

# 16bitカラーも検討する



# 16bitダイレクトカラーの検討

---

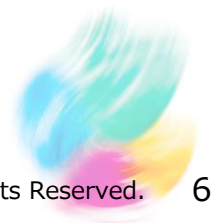
## 検討シチュエーション

- 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 (ARGB8888)	32bpp	1
16bit ダイレクトカラー (ARGB1555, ARGB4444)	16bpp	1/2
DXTC (DXT2~5)	8bpp	1/4
PVRTC (4bpp)	4bpp	1/8
ETC + アルファマスク	8bpp (4bpp × 2)	1/4



## 16bitダイレクトカラー形式の種類

フォーマット	RGBカラー値A	アルファ値
RGB565	16Abit	無し
ARGB1555	15bi	1bit (抜き)
ARGB4444	12bit	4bit (16階調)

RGBカラー値のbitが少ないほど  
色味部分のクオリティは低下します



さらにキレイにするために…

**高性能な圧縮ツールの導入も  
検討してください**



# 最後に少しPR





## 元画像 (アルファ付32bit)

\* 文字の周囲が透明αで  
抜かれている画像



## 標準ツールで圧縮したPVRTC

\* 文字の周囲が透明αで  
抜かれている画像



## OPTPIX iméstaで圧縮したPVRTC

\* 文字の周囲が透明αで  
抜かれている画像



\* 文字の周囲をアルファで抜いたような画像に高い効果があります



## OPTPIX iméstaで256色(PNG8)に減色した結果



OPTPIX iméstaなら、  
256色に減色しても  
これだけの色再現結果が  
得られます。



<http://www.webtech.co.jp/imesta/mobile/>

# OPTPIX imésta 7

for Mobile & Social

**PNG**

**JPEG**

**GIF**

**WebP**

**Index  
Color  
(CLUT)**

**16bit  
Direct  
Color**

**DXTC  
(S3TC)**

**PVRTC**

**ETC**



クリエイターの笑顔のために、  
皆様の困ったことを解決するのが  
ウェブテクノロジーの仕事です。



# 御清聴ありがとうございました

<https://www.facebook.com/OPTPIX>

<http://www.webtech.co.jp/>

