



パレットテクスチャ仕様書

(8/4/98)

目次

1. 概要	3
2. 構造体とマクロ定義	4
2.1 構造体	4
2.2 アスキー出力のマクロ定義	5
2.3 アスキー出力の例	6
3. パレットのファイルフォーマット	7
3.1 index image .pvr	7
3.2 palette data .pvp	8
4. データ作成手順	10

1. 概要

パレットは 1024 エントリから構成されます。各エントリはバンク番号で分割して管理されます。8bpp の場合は 256 色で 4 バンク、4bpp の場合は 16 色で 64 バンクになります。バンク番号は 1024 エントリの色を一意に決めるためのオフセットとして利用されるだけで 8bpp、4bpp を 1024 エントリ上にどう配置するかは自由です。ただしバンクにまたがる 16 色、256 色の利用はできません。パレットのカラーは次の 4 種類があります。

RGB565
ARGB1555
ARGB4444
ARGB8888

ARGB8888 は 32 ビットカラーであり、その他は 16 ビットカラーになります。32 ビットカラーは本ハードウェアではパレットでのみ使うことができます。

(注意事項)

四つのカラータイプは混在して同時に使えません。つまり一フレームに上記の四つから一種類だけを選んでその種類のテクスチャのみを使いパレットを設定する必要があります(ハードウェア仕様)。例えば ARGB1555 と ARGB4444 のパレットテクスチャを同時には使えません。どちらか一方に統一する必要があります。

Ninja ではパレットテクスチャを利用するために texlist にバンク番号 (texaddr の最上位 6 ビットにセット) とテクスチャ連続フラグ (新規フラグを定義したフラグを attr にセット) を格納します。

バンク番号の指定はマテリアルネームからします。文字列 B00 ~ B63 で指定します。例えば B01 の代わりに B1 は使えません。必ず 0 をつけて B01 として設定してください。この番号は 4bpp の場合のバンク番号ですが 8bpp の場合はこの番号の下位 4 ビットを無視した値が有効となります。つまり B00、B16、B32、B48 だけが意味を持ちます。

デザイナーはモデル作成時にテクスチャとバンク番号を指定しモデルをコンバートします。texlist 出力では同じテクスチャは一つのエントリにまとめられますがバンク番号が指定された場合同じテクスチャでもバンク番号が異なれば別のエントリになります。コンバートにより 16 ビットテクスチャが貼られます。ビューアでそのままテクスチャ付きのモデルを見ることができます。その後 8 ビットペインタでテクスチャのパレット化をし.pvr ファイルを差し替えます。

パレットテクスチャ出力は二つのファイルから構成されます。

.pvr ファイル：インデックス画像
.pvp ファイル：パレットデータ

.pvr は 16 ビットカラーテクスチャ同様にハードウェアが期待するバイトイメージです。インデックスイメージは正方形であり twiddled です。ミップマップが利用できますが 16 ビットカラーテクスチャでは一つ上のミップマップのピクセルを複数参照による解像度変換を実現していましたが場合とは異なりもっとも上位のパレットテクスチャイメージに対するパレットの色をそのまま利用するため隣り合うピクセル間の色の変化がきつくなる可能性があります。

.pvp ファイルはピクセルのタイプ、バンク番号、エントリオフセット、エントリカウントをヘッダ情報として持つ PVPL チャンクを格納します。ユーザカスタマイズにより複数の PVPL チャンクを格納することで一括したパレットデータの書き込みが可能です。

2. 構造体とマクロ定義

2.1 構造体

```
typedef struct {  
    void            *filename; /* texture filename strings    */  
    Uint32          attr;     /* texture attribute      */  
    Uint32          texaddr;  /* texture memory list address */  
} NJS_TEXNAME;
```

attrにはライブラリが利用するフラグ群がセットされます。ここに新規に次のフラグを定義します。

<NinjaDef.h>

```
#define NJD_TEXATTR_TEXCONTINUE          BIT_16
```

このフラグは同一テクスチャでバンク番号のみが異なる連続するエントリで一つ前のテクスチャと同じであることを示しカレントのテクスチャファイルのオープンを省略することを可能とします。コンバータで自動設定します。

texaddrは通常ライブラリ用のワーク構造体ポインタとユーザ指定のglobal Index指定に使われます。パレットバンク指定時にはこの上位6ビットにバンク番号0～63を設定します。global Index指定はメモリタイプのテクスチャの場合のみ使われますがメモリタイプでかつパレットタイプでは上位6ビットがバンク番号指定になるため26ビット精度の値までしか使えませんので注意してください。エントリのテクスチャがパレットかそうでないかは.pvrファイルの中のカテゴリコードで検出されます。

<NinjaDef.h>

```
#define NJD_TEXBANK_SHIFT                (26)  
#define NJD_TEXBANKMASK                  (0xFC000000)
```

```
typedef struct {  
    NJS_TEXNAME      *textures; /* texture array          */  
    Uint32           nbTexture; /* texture count          */  
} NJS_TEXLIST;
```

2.2 アスキー出力のマクロ定義

アスキー出力にはNjDef.hに定義されるマクロが利用されます。従来のマクロには次のものがあります。

```
#define TEXN(_texname)      {(_texname), 0x0, 0}
#define TEXN3(_texname, _attr, _texaddr) {(_texname), _attr, _texaddr}
```

通常ファイル出力時はTEXNマクロが使われます。オプションが指定された場合にTEXN3出力がされます。新規に次のマクロが定義されます。

```
#define _SET_CNT(_f) (_f ? NJD_TEXATTR_TEXCONTINUE : 0)
#define _SET_BNK(_b) (((_b) << NJD_TEXBANK_SHIFT) & NJD_TEXBANK_MASK)
#define _SET_BNK2(_b, _g) (_SET_BNK(_b)|((_g) & ~NJD_TEXBANK_MASK))

#define PTEXN(_tname, _cnt, _bnk) {(_tname), _SET_CNT(_cnt), _SET_BNK(_bnk)}
#define PTEXN5(_tname, _attr, _texaddr, _cnt, _bnk)                                ￥
    {(_tname), _SET_CNT(_cnt)|(_attr), _SET_BNK2(_bnk, _texaddr)}
```

_SET_CNT、_SET_BNK、_SET_BNK2は内部で使われるマクロです。PTEXN、PTEXN5はマテリアルネームからバンク番号が指定された場合そのエントリの出力はTEXNからPTEXNに自動で切り替わります。TEXN3の場合はPTEXN5になります。

2.3 アスキー出力の例

```
TEXTURENAME textures_MODEL[]
START
    PTEXN( "texturePAL1", 0, 20 ),   バンク番号が指定された。
    PTEXN( "texturePAL1", 1, 21 ),   バンク番号が指定された。
    TEXN( "texture1" ),
    PTEXN( "texturePAL2", 0, 63 ),   バンク番号が指定された。
    TEXN( "texture2" ),
END

TEXTURELIST texlist_MODEL
START
    TextureList textures_MODEL,
    TextureNum 5,
END
```

PTEXN の第二引数の 1 は一つ前のテクスチャと同じテクスチャが指定されている場合に利用する。このフラグが 1 の場合は二つ目のテクスチャ読み込み（ここでは 2 ライン目の texturePAL1 のロード）を省略し一つ前のテクスチャの情報を利用する。パレットではテクスチャが同じで異なるパレットバンクを使う可能性がありこれに効率よく対応するためにこのフラグを活用します。同一テクスチャ名はコンバータにより連続するように出力されます。またこのフラグがセットされていなくても globalIndex のチェックで texlist 上の同じファイルが二重読み込みされることはありません。

3. パレットのファイルフォーマット

3.1 index image .pvr

'PVRT'(4)		PowerVR用テクスチャチャンクであることを示す。
bytesize(4)		次のチャンクまでのバイトサイズ。
category code(4)		カラータイプ、4bpp、8bpp、ミップマップありなしを指定。
xsize(2)	ysize(2)	テクスチャの幅と高さ。
Data		インデックス画像。

category code:

NinjaDef.hに定義されます。

```
<palette texture format>
#define NJD_TEXFMT_PALETTE4      (0x0400)
#define NJD_TEXFMT_PALETTE4_MM  (0x0500)
#define NJD_TEXFMT_PALETTE8      (0x0600)
#define NJD_TEXFMT_PALETTE8_MM  (0x0700)
```

```
<palette color format>
#define NJD_TEXFMT_ARGB1555      (0x0000)
#define NJD_TEXFMT_RGB565        (0x0001)
#define NJD_TEXFMT_ARGB4444      (0x0002)
#define NJD_TEXFMT_ARGB8888      (0x0006)
```

texture formatとcolor formatのorをとったものがカテゴリコードとして設定されます。

Data:

4bpp, 8bpp の xsize×ysize のインデックス画像、twiddledのみ。ミップマップを使うこともできます。

3.2 palette data .pvp

チャンクネーム 'PVPL'を持つパレットチャンクを格納します。

'PVPL'(4)	PowerVR 用パレットデータチャンクであることを示します。
bytesize(4)	次のチャンクまでのバイトサイズ。
category code(2)	カラータイプ、4bpp、8bpp を指定。
bankId(2)	バンク指定。0 ~ 63、パレットでない場合 - 1 を設定。
entryoffset(2)	バンク内のエントリオフセット。
entrycount(2)	palette data に格納される色の数。
<div>palette data</div> <div><u>RGB565 の場合</u> [16bitRGB]の一次元配列。</div> <div><u>ARGB1555、ARGB4444 の場合</u> [16bitARGB]の一次元配列。</div> <div><u>ARGB8888 の場合</u> [32bitARGB]の一次元配列。</div>	

category code:

<palette color format>

```
#define NJD_TEXFMT_ARGB1555      (0x0000)
#define NJD_TEXFMT_RGB565        (0x0001)
#define NJD_TEXFMT_ARGB4444      (0x0002)
#define NJD_TEXFMT_ARGB8888      (0x0006)
```

texture format は設定しません。color format のみカテゴリコードとして設定。このカテゴリコードが ARGB8888 の場合は Data のピクセルサイズが 32 ビット、それ以外の場合は 16 ビットにます。

bank:

0-63 の数字で示します。4bpp の場合は 0-63 ですが 8bpp の場合はこの数字の上位 2 ビットのみが有効。バンク番号は 0, 16, 32, 48 のみが利用できます。これ以外のバンク番号はまるめられます。例えば 1 ~ 15 は 0 になります。

entryoffset:

バンクの先頭からのオフセット。バンクの先頭のエントリを 0 番として指定。

entrycount:

データに格納される色のエントリ数。

Data:

 エントリ数分の色データ。32 ビット、16 ビットの 2 タイプ。

 RGB565、ARGB1555、ARGB4444 の場合は 16 ビットカラーデータの一次元配列。ARGB8888 の場合は 32 ビットカラーデータの一次元配列がデータとなります。エントリ数は entrycount で与えられます。

 .pvp ファイル内部において PVPL チャンクは複数定義することができます。複数バンク分のパレットデータを一つのファイルの中で定義でき一括してライブラリでロードできます。デフォルト動作では PVPL は .pvp ファイルにひとつです。パレットテクスチャは正方形のみで twiddled になります。ミップマップを使う場合は親画像のパレット色が共有されますのでピクセルの単純な間引きによりミップマップ画像を得るため 16 ビットカラーテクスチャに比べミップマップが汚くなることに注意してください。

4. データ作成手順

現バージョンにおけるパレットテクスチャデータの生成の手順を示します。モデラーで直接パレットテクスチャを貼ることは今回はしません。データ作成後に.pvr ファイルをパレットデータに置き換えることによりパレットを利用します。

<step1>

通常の手順でモデルにテクスチャを貼ります。この時貼り込むテクスチャはパレット化する元画像となるテクスチャです。

<step2>

マテリアルネームからバンク番号を設定します。

4bpp (16 色カラー) の場合

B00 ~ B63 をマテリアルネームに設定します。

8bpp (256 色カラー) の場合

B00、B16、B32、B48 の 4 つが有効です。これをマテリアルネームに設定します。こ例外のバンク番号は丸められます。B01 ~ B15 は B0 に B17 ~ B31 は B16 に B33 ~ B47 は B32 に B49 ~ B63 は B48 として扱われます。

4bpp と 8bpp のどちらも同じバンク番号表記 0 ~ 63 を使いますがカテゴリーコードに格納されるテクスチャのタイプにより意味が異なることに注意してください。

<step3>

コンバータで texlist を出力します。バンク番号が指定されたテクスチャに関してはこのバンク番号が texlist に格納されます。同一テクスチャでもマテリアルに設定されたバンク番号が異なる場合別のエントリとして texlist に登録されます。マテリアルネームからバンク番号を指定しなかったテクスチャに関しては従来の通りの動作をします。全テクスチャが 16 ビットカラーとしてコンバートされ.pvr ファイルが出力されます。

<step4>

モデルに貼った元画像テクスチャを 8 ビットペインタでロードしパレットタイプへ変換、修正します。そのデータを 8 ビットペインタから.pvr ファイルと.pvp ファイルで出力します。.pvp ファイル出力時にバンク番号 0 ~ 63 を指定します。**このバンク番号はモデル出力時に設定したバンク番号と一致する必要があります。**

<step5>

モデルコンバート時に生成された.pvr データ群のパレットにしたい部分のテクスチャファイルを上書きします。

現在各テクスチャごとに.pvp ファイルが生成されますがこれをマージするツールを用意する予定です。また.pvm ファイルでパレット情報も含めて扱うように仕様拡張をする予定です。急ぎのユーザで.pvp ファイルをマージしたい場合は Unix 環境では cat コマンドを PC 環境では copy コマンドを利用してマージしてください。

以 上