



Chunk Model 仕様書

(7/12/98)

目次

1. 概要	3
2. モデル構造体	4
3. Chunk 仕様	7
3.1 Chunk の種類	7
3.2 Chunk の構造	8
3.3 Chunk NULL	9
3.4 Chunk End	10
3.5 Chunk Bits	10
3.6 Chunk Tiny	15
3.7 Chunk Material	17
3.8 Chunk Vertex	25
3.9 Chunk Volume	45
3.10 Chunk Strip	49
4. アスキー出力の注意事項	61

1. 概要

Ninja は Basic Model、Chunk Model の二つの形式のモデル構造をサポートしています。Chunk Model 形式は、描画実行中に SH4 のキャッシュを壊さないようにデータを連続するメモリ空間に配置します。拡張性、柔軟性、データの表現効率に優れています。今後は Chunk Model 形式のモデルを中心としたチューニングを実施します。Basic Model はサポートされますが、新機能はサポートされません。

Chunk Model 形式は Model 構造体の中身を大幅に変更しますが、Object 構造体は Model 構造体のポインタから Chunk Model 構造体のポインタへの変更以外に変更されません。

モデル以外のモーションおよびテクスチャについては、引き続き現行の形式が使用されます(ただしカメラ、ライト対応のために構造体メンバの型が変更されます)。

Basic Model 及びテクスチャ構造については、Basic Model 仕様書を参照してください。Chunk Model の特徴を以下に示します。

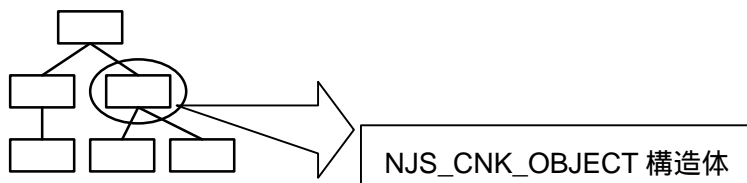
Chunk Model の特徴

- トライアングルストリップ描画を基本とします。現在は三角形、四角形、N角形を描画できません。性能優先で設計されています。
- データは、頂点リスト vlist とポリゴンリスト plist からなります。vlist、plist 上に IFF チャンク形式でデータを配置してメモリ領域を一本化し、描画実行中にキャッシュを壊さないようにしています。
- ポリゴン側と頂点側のどちらかに、頂点カラー情報を持つことができます。ポリゴン側では各ポリゴンごとに頂点に色が付けられます。
- ポリゴン側に頂点法線を持つことができます。頂点法線がポリゴン単位で持てるので、softimage の頂点法線をそのまま出力できます。Discontinuity データを出力できます。
- ポリゴン側 (最大 16bit × 3) と頂点側 (32bit × 1) にユーザフラグ領域を持つことができます。現在は頂点カラーデータをユーザフラグ領域へ出力するときに、この領域を使用します。今後この部分にユーザデータを書き込めるツールを用意する予定です。
- マテリアルは plist に格納され、以前に設定したマテリアルの変更部分 (差分) のみの設定だけを更新します。これにより、Basic Model の場合よりもマテリアル設定回数が減少します。
- コンバータ出力時にマテリアルを削除できます。同じモデルの描画 (例えば木) などすべてのマテリアルをなくし、ユーザが外で設定することでデータを最適化することができます。
- データ量削減のために 10 ビット法線がサポートされています。XYZ の法線が Uint32 の中に各 10 ビットで格納されます。2 ビットはリザーブ領域として 0 を入れます。
- コリジョン用データ Chunk Volume の出力がサポートされています。三角形、四角形、トライアングルストリップで出力できます。マテリアル情報を持ちません。ユーザフラグ領域を持つことができます。現在この領域にマテリアルカラーを出力できます。
- 独立三角形の Chunk Volume (volume3) は、モディファイアボリュームにも利用できます。
- コンバータで volume34 を指定すると、面間角度が 0.1 度の三角形同士を接続し四角形を作ります。3D Studio MAX は三角形データしか出力できませんが、この場合でも四角形コリジョンが生成できます。
- SH4 のハードウェアを効率良く利用し、高速処理を可能とする頂点形式がサポートされています (NJD_CV_SH、NJD_CV_VN_SH)。これは、SH4 のマトリックス演算命令を効果的に利用することによって、高速化を実現しています。データ量よりも性能優先の特殊な場合に利用します。
- 二種類の UV 値表現を持ちます。0-255 による UVN、分解能を高めた 0-1023 による UVH です。UVN は、Basic Model でも利用していた従来の表現です。256 を超えるテクスチャでは、1 ピクセル単位で指定できません。UVH はハイレゾモード、1024x1024 のテクスチャで、1 ピクセル単位での指定が可能です。ただし UVH は UVN よりも分解能を上げた分だけ、テクスチャのリピートの表現回数が減少します (UVN が 128 回に対し UVH では 32 回)。コンバートオプションによりモデルツリー全体に対しまたマテリアルネームにより各モデル単位で UVN、UVH を切り替えることができます。マテリアルネームからの指定では単一のモデルに使われる複数のマテリアルのどれか一つに設定することでモデルに対する UV 表現が変更されます。デフォルトは UVN。

2. モデル構造体

構造体図

Chunk Object Tree



```
typedef struct cnkobj {
    Uint32      evalflags; /* 評価方法最適化 */
    NJS_CNK_MODEL *model; /* モデル構造体 */
    Float       pos[3]; /* 平行移動 */
    Angle       ang[3]; /* 回転 */
    Float       scl[3]; /* スケール */
    struct cnkobj *child; /* 子どもポインタ */
    struct cnkobj *sibling; /* 兄弟ポインタ */
} NJS_CNK_OBJECT;
```

NJS_CNK_MODEL 構造体

```
typedef struct {
    Sint32 *vlist; /* vertex list */
    Sint16 *plist; /* polygon list */
    NJS_POINT3 center; /* model center */
    Float r; /* radius */
} NJS_CNK_MODEL;
```

vlist (32bit ベース Chunk List)

頂点リストを与えます。必要に応じてチャンクの種類を切り替え、チャンクに頂点、法線、頂点カラー、ユーザデータを格納します。

plist (16bit ベース Chunk List)

ポリゴンインデックスリストを与えます。必要に応じてチャンクの種類を切り替え、チャンクにUV値、頂点カラー、頂点法線、ユーザデータを格納します。

vlist の頂点リストはマトリクス演算され、Ninja の内部中間バッファに格納されます。その中間バッファに対し、plist のポリゴンインデックスが適用されます。つまり vlist の計算後に plist が呼び出されるため、この二つのデータ領域が同時に（交互に）呼び出されることはなく、plist、vlist に関しては別ポインタでもキャッシュを壊すことはありません。

構造体解説

Float, Angle

```
typedef float Float      /* 浮動小数点演算型 */
typedef Sint32 Angle     /* 回転角度 */
アングルは 0x0000 ~ 0xFFFF が 0 ~ 360 度。
```

ポイント構造体

```
typedef struct {
    Float    x;          /* X 値 */
    Float    y;          /* Y 値 */
    Float    z;          /* Z 値 */
} NJS_POINT3, NJS_VECTOR;
頂点の xyz 値を与えます。
```

チャンクモデル構造体

```
typedef struct {
    Sint32    *vlist;    /* 頂点チャンクリスト */
    Sint16    *plist;    /* ポリゴンチャンクリスト */
    NJS_POINT3 center;   /* モデルの中心 */
    Float      r;        /* モデルの半径 */
} NJS_CNK_MODEL;
```

- vlist には頂点リストのデータを、Sint32 配列上に iff チャンク形式で並べます。plist にはポリゴンインデックスリストのデータを、Sint16 配列上に iff チャンク形式で並べます。center はモデルの外接球の中心位置であり、r はその半径です。

チャンクオブジェクト構造体

```
typedef struct cnkobj {
    Uint32      evalflags; /* マトリクス演算評価フラグ */
    NJS_CNK_MODEL *model;  /* チャンクモデルポインタ */
    Float       pos[3];    /* 移動量 */
    Angle       ang[3];    /* 回転量 */
    Float       scl[3];    /* スケール */
    struct cnkobj *child;   /* 子供へのポインタ */
    struct cnkobj *sibling; /* 兄弟ポインタ */
} NJS_CNK_OBJECT;
```

- モデルの親子階層を与えます。evalflags はマトリクス演算最適化のためのフラグを与え、model にはチャンクモデル構造体ポインタがフックされます。ポリゴンを持たないノードでは、このポインタに NULL を設定します。pos は移動量、rot は回転量、scl はスケール、child、sibling は階層の子供、兄弟のオブジェクトポインタを与えます。

evalflags フラグ説明

```
#define NJD_EVAL_UNIT_POS BIT_0/* 移動が無視できる */
#define NJD_EVAL_UNIT_ANG BIT_1/* 回転が無視できる */
#define NJD_EVAL_UNIT_SCL BIT_2/*スケールが無視できる。*/
#define NJD_EVAL_HIDE BIT_3/* モデルを描画しない。 */
#define NJD_EVAL_BREAK BIT_4/*child のトレースの打ち切り*/
#define NJD_EVAL_ZXY_ANG BIT_5
/* LightWave3D で期待される回転の評価の指定 */
#define NJD_EVAL_SKIP BIT_6 /* モーションをスキップ */
#define NJD_EVAL_SHAPE_SKIP BIT_7
/* シェイプモーションをスキップ */
#define NJD_EVAL_MASK 0xff
/* 上記ビットを抽出するためのマスク */
```

これらのフラグはコンバータで設定されます。

- NJD_EVAL_UNIT_POS は、平行移動量がゼロの場合に設定されます。平行移動のマトリクス計算が省略されます。
- NJD_EVAL_UNIT_ANG は、回転量がゼロの場合に設定されます。回転のマトリクス計算が省略されます。
- NJD_EVAL_UNIT_SCL は、スケールが xyz とともに 1 の場合に設定されます。スケールのマトリクス計算が省略されます。
- NJD_EVAL_UNIT_POS, NJD_EVAL_UNIT_ANG, NJD_EVAL_UNIT_SCL の 3 つがセットされていると、すべてのマトリクス演算に加え、マトリクスの pushpop も省略されます。
- NJD_EVAL_HIDE は、ユーザによって立てられます。このフラグが立っている場合、モデルが描画されません。モデルに装着されたガンとブレードの切り替えなどに使用します。
- NJD_EVAL_BREAK は、ユーザによって立てられます。このフラグが立っている場合、**child** の探索をここで打ち切ります。例えば、ルートノードでこのフラグを立てると、モデル全体が消えます。NJD_EVAL_BREAK は、モーションと組み合わせるとデータがずれるので、ルートノードのみで使用することをお勧めしますが、ユーザの責任において途中のノードで使用することもできます。
- LightWave3D では、回転の評価の順番は ZXY となります。Ninja は通常 XYZ の順番であるため、LightWave3D の場合の評価順をライブラリで実行するためのフラグとして、NJD_EVAL_ZXY_ANG が用意されます。このフラグが ON の場合、回転の計算順序が ZXY に変更されます。
- NJD_EVAL_SKIP は、このノードがモーションデータに含まれないことを示します。モーション実行時はモーションのノードを次に進めずに、オブジェクト構造体の値でマトリクス計算を行い、次のノードに進みます。これにより、ポリゴンモデルの構成が異なるモデルでも、骨の構造が同じであればモーションが行えます。
- NJD_EVAL_SHAPE_SKIP は、このノードがシェイプモーションデータに含まれないことを示します。
- NJD_EVAL_SKIP、NJD_EVAL_SHAPE_SKIP はマテリアルネームからの指定が可能です。

3. Chunk 仕様

3.1 Chunk の種類

Chunk の名前	略号	サイズ	説明
<u>C</u> hunk <u>N</u> ULL	CN	16bit	ロングワードアライメントあわせ。
<u>C</u> hunk <u>E</u> nd	CE	16bit	チャンクデータリストの終わりを示します。
<u>C</u> hunk <u>B</u> its	CB	16bit	フラグを設定。Blend Alpha など。
<u>C</u> hunk <u>T</u> iny	CT	32bit	フラグと単一の値を設定します。TexId など。
<u>C</u> hunk <u>M</u> aterial	CM	可変	Diffuse, Specular, Exponent, Ambient を設定します。
<u>C</u> hunk <u>V</u> ertex	CV	可変	頂点リストを与えます。
<u>C</u> hunk <u>V</u> olume	CO	可変	コリジョン、モディファイアボリューム用データを与えます。
<u>C</u> hunk <u>S</u> trip	CS	可変	ストリップデータを与えます。

基本構造をベースに、目的に合わせて構造を簡略化した Chunk(Bits, Tiny 等)を定義しています。頂点用の Chunk Vertex は Chunk Model 構造体の vlist に格納されます。それ以外のチャンクは plist に格納されます。これらは NinjaCnk.h に定義されています。

3.2 Chunk の構造

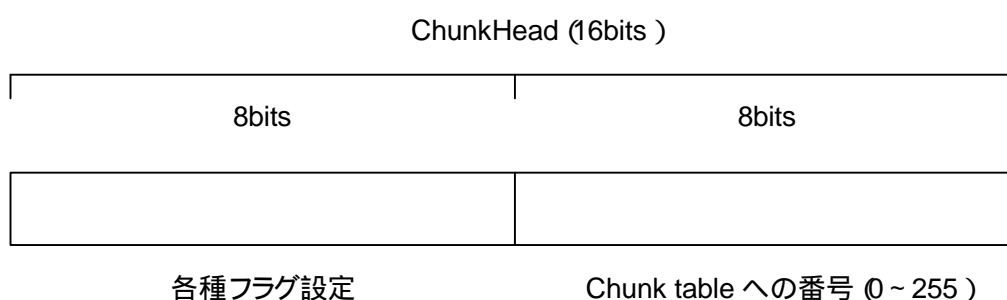
チャンクの基本構造は次の通りです。

Chunk Vertex の場合

[headbits(15-8) | ChunkHead(7-0)][longsize(15-0)][data]

Chunk Vertex 以外の場合

[headbits(15-8) | ChunkHead(7-0)][shortsize(15-0)][data]



ChunkHead は、そのチャンクを処理する関数 table のエントリ番号を与えます。ライブラリはこの番号から関数を選択し、実行します。チャンクごとに処理関数を分けることにより、描画ルーチンを単純化し、高速化ルーチンを実現することができます。テーブルサイズは最大 256 個とするため上位 8bit が余ります。この部分 (headbits と呼ぶ) 8bit へ、チャンクの種類により目的に合わせてアトリビュートフラグの一部を格納し、データサイズの効率を高めます。チャンクテーブルの関数エントリ番号を得るには、上位 8bit をマスクする必要があることに注意してください。

shortsize、longsize は、次のチャンクの先頭までのオフセットを与えます。通常 iff 形式では次のチャンクまでのデータオフセットをバイトで与えますが対象とする plist が short の配列、vlist が long の配列であることからそれぞれ shortsize(2 バイト)単位、longsize (4 バイト) 単位でオフセットを表現します。これは次のチャンクまでのオフセットの表現可能最大数を大きくする効果があります。

チャンク形式を利用しているため例えばユーザがマテリアルだけを書き換えたい場合は、マテリアルのチャンクが見つかるまで shortsize を利用してデータをスキップし、マテリアルを発見したところで値を書き換えるようなことができます。

3.3 Chunk NULL

ChunkName : 'NJD_CN'

(Chunk NULL)

概要 :

plist においてロングワードアライメント調整のために、チャンクとチャンクの間に挿入されます。ChunkHead (16bit) のみのチャンク。

形式 :

```
[ ChunkHead ( 15-0 ) ]  
  ChunkHead:  
    NJD_CN
```

説明 :

```
#define NJD_CN (NJD_NULLOFF+0)
```

plist は Sint16 をベースとした一次元配列です。そのため、ストリップの終わりが必ずしも Sint32 バウンダリにならない場合があります。Chunk Material は Sint16 の配列上であっても、Sint32 のデータとして読み込んだ方が、効率が上がります。従って、Chunk Material の開始が Sint32 バウンダリになるように、Chunk Strip の後ろに付けられます。性能面からライブラリとの擦りあわせによりバウンダリあわせをするかどうかは、変更される可能性があります。バウンダリあわせが行われない場合、NJD_CN を使用しません。

3.4 Chunk End

ChunkName : 'NJD_CE'

(Chunk End)

概要 :

plist、vlist のチャンクリストの最後を与えます。ChunkHead(16bit)のみのチャンクです。vlist の場合は、ChunkHead (32bit) として扱われます。実際の値は最上位ビットに 1 が立つことにより検出されます。

形式 :

plist の場合 [ChunkHead(15-0)] (16 bits chunk)
vlist の場合 [ChunkHead(31-0)] (32 bits chunk)
ChunkHead:
NJD_CE

説明 :

```
#define NJD_CE (NJD_ENDOFF+0)
```

リストの完了を検出します。

3.5 Chunk Bits

Chunk Bits は、アトリビュートフラグのようなフラグを書き換えるために利用します。

[headbits(15-8) | ChunkHead(7-0)](16 bits chunk)

上位 8bit にフラグを格納し、下位 8bit でチャンクの番号を与えます。ChunkHead (16bit) のみのチャンクです。次に実際の Chunk Bits について説明します。

ChunkName : 'NJD_CB_BA'
(Chunk Bits Blend Alpha)

概要 :

plist においてアトリビュートフラグの Blend Alpha を設定します。

形式 :

```
[headbits(13-8) | ChunkHead(7-0)]
headbits:
    13-11 = SRC Alpha Instruction(3bit)
    10- 8 = DST Alpha Instruction(3bit)
ChunkHead:
    NJD_CB_BA
```

説明 :

```
#define NJD_CB_BA    (NJD_BITSOFF+0)
```

Blend Alpha は、2 種類の方法で設定されます。Chunk Material (後述) の headbits として、diffuse、specular、ambient の設定とともに Blend Alpha を設定できます。マテリアルを変更せず Blend Alpha のみ設定する場合に、NJD_CB_BA を利用します。

ブレンディング機能では、2 つの RGBA 値、SRC と DST が下記のように合成され、その結果は DST に書き戻されます。

```
DST := SRC * BlendFunction(SRC Alpha Instruction) +
      DST * BlendFunction(DST Alpha Instruction)
```

ここで、BlendFunction(Instruction) には、SRC/DST カラーとともに 3 ビットの Instruction が入力されます。そして、それぞれの RGBA に対して、4 つの 値によって重み付けされた係数値を戻します。

Instruction	Field Value	Values Returned
Zero	0	(0, 0, 0, 0)
One	1	(1, 1, 1, 1)
'Other' Colour	2	(O _R , O _G , O _B , O _A)
Inverse 'Other' Colour	3	(1 - O _R , 1 - O _G , 1 - O _B , 1 - O _A)
SRC Alpha	4	(S _A , S _A , S _A , S _A)
Inverse SRC Alpha	5	(1 - S _A , 1 - S _A , 1 - S _A , 1 - S _A)
DST Alpha	6	(D _A , D _A , D _A , D _A)
Inverse DST Alpha	7	(1 - D _A , 1 - D _A , 1 - D _A , 1 - D _A)

'Other Colour'と'Inverse Other Colour'は、SRC インストラクションに指定されたときには DST のカラーが使用され、DST インストラクションに指定されたときには SRC のカラーが使用されることを意味します。

略号の意味は次の通りです。

ZER: Zero
ONE: One
OC: `Other' Color
IOC: Inverse `Other' Color
SA: Src Alpha
ISA: Inverse SRC Alpha
DA: DST Alpha
IDA: Inverse DST Alpha

Flag Blending Src :

```
#define NJD_FBS_SHIFT      11
#define NJD_FBS_ZER        0<<NJD_FBS_SHIFT)
#define NJD_FBS_ONE        (1<<NJD_FBS_SHIFT)
#define NJD_FBS_OC         (2<<NJD_FBS_SHIFT)
#define NJD_FBS_IOC        (3<<NJD_FBS_SHIFT)
#define NJD_FBS_SA         (4<<NJD_FBS_SHIFT)
#define NJD_FBS_ISA        (5<<NJD_FBS_SHIFT)
#define NJD_FBS_DA         (6<<NJD_FBS_SHIFT)
#define NJD_FBS_IDA        (7<<NJD_FBS_SHIFT)

#define NJD_FBS_MASK        (0x7<<NJD_FBS_SHIFT)
```

Flag Blending Dst :

```
#define NJD_FBD_SHIFT      8
#define NJD_FBD_ZER        (0<<NJD_FBD_SHIFT)
#define NJD_FBD_ONE        (1<<NJD_FBD_SHIFT)
#define NJD_FBD_OC         (2<<NJD_FBD_SHIFT)
#define NJD_FBD_IOC        (3<<NJD_FBD_SHIFT)
#define NJD_FBD_SA         (4<<NJD_FBD_SHIFT)
#define NJD_FBD_ISA        (5<<NJD_FBD_SHIFT)
#define NJD_FBD_DA         (6<<NJD_FBD_SHIFT)
#define NJD_FBD_IDA        (7<<NJD_FBD_SHIFT)

#define NJD_FBD_MASK        (0x7<<NJD_FBD_SHIFT)
```

ChunkName : 'NJD_CB_DA'
(Chunk Bits 'D' Adjust)

概要 :

plist において Mipmap 'D' adjust 値を設定します。

形式 :

```
[headbits(11-8) | ChunkHead(7-0)]
headbits:
    11- 8 = Mipmap 'D' adjust(4)
ChunkHead:
    NJD_CB_DA
```

説明 :

```
#define NJD_CB_DA                (NJD_BITSOFF+1)
```

Mipmap の切り替えの深さを調整します。デフォルトは 1.00 です。これは頻繁に切り替えるものではなく、mipmap の切り替えが目立つ場合に、これを抑制する目的で利用します。

Flag 'D' Adjust :

```
#define NJD_FDA_SHIFT            8
#define NJD_FDA_025              (1<<NJD_FDA_SHIFT)      /* 0.25 */
#define NJD_FDA_050              (2<<NJD_FDA_SHIFT)      /* 0.50 */
#define NJD_FDA_075              (3<<NJD_FDA_SHIFT)      /* 0.75 */
#define NJD_FDA_100              (4<<NJD_FDA_SHIFT)      /* 1.00 */
#define NJD_FDA_125              (5<<NJD_FDA_SHIFT)      /* 1.25 */
#define NJD_FDA_150              (6<<NJD_FDA_SHIFT)      /* 1.50 */
#define NJD_FDA_175              (7<<NJD_FDA_SHIFT)      /* 1.75 */
#define NJD_FDA_200              (8<<NJD_FDA_SHIFT)      /* 2.00 */
#define NJD_FDA_225              (9<<NJD_FDA_SHIFT)      /* 2.25 */
#define NJD_FDA_250              (10<<NJD_FDA_SHIFT)     /* 2.25 */
#define NJD_FDA_275              (11<<NJD_FDA_SHIFT)     /* 2.25 */
#define NJD_FDA_300              (12<<NJD_FDA_SHIFT)     /* 3.00 */
#define NJD_FDA_325              (13<<NJD_FDA_SHIFT)     /* 3.25 */
#define NJD_FDA_350              (14<<NJD_FDA_SHIFT)     /* 3.50 */
#define NJD_FDA_375              (15<<NJD_FDA_SHIFT)     /* 3.75 */

#define NJD_FDA_MASK              (0xf<<NJD_FDA_SHIFT)
```

ChunkName : 'NJD_CB_EXP'

(Chunk Bits Exponent)

概要 :

plist においてスペキュラの Exponent を設定します。0 ~ 16 の間が有効です。

形式 :

```
[headbits(12-8) | ChunkHead(7-0)]
headbits:
    12- 8 = Exponent(5) range:0-16
ChunkHead:
    NJD_CB_EXP
```

説明 :

```
#define NJD_CB_EXP (NJD_BITSOFF+2)
```

exponent は、2 種類の方法で設定されます。Chunk Material (後述) の中で specular を設定する場合は、Chunk Material の specular 成分の上位 8bit で exponent を設定します(ERGB8888 となる)。その前に設定した specular 値が変化せず exponent のみが増加した場合に限り、NJD_CB_EXP が使用されます。

Flag EXPonent(range 0-16) :

```
#define NJD_FEXP_SHIFT      8
#define NJD_FEXP_00         (0<<NJD_FEXP_SHIFT) /* 0.0 */
#define NJD_FEXP_01         (1<<NJD_FEXP_SHIFT) /* 1.0 */
#define NJD_FEXP_02         (2<<NJD_FEXP_SHIFT) /* 2.0 */
#define NJD_FEXP_03         (3<<NJD_FEXP_SHIFT) /* 3.0 */
#define NJD_FEXP_04         (4<<NJD_FEXP_SHIFT) /* 4.0 */
#define NJD_FEXP_05         (5<<NJD_FEXP_SHIFT) /* 5.0 */
#define NJD_FEXP_06         (6<<NJD_FEXP_SHIFT) /* 6.0 */
#define NJD_FEXP_07         (7<<NJD_FEXP_SHIFT) /* 7.0 */
#define NJD_FEXP_08         (8<<NJD_FEXP_SHIFT) /* 8.0 */
#define NJD_FEXP_09         (9<<NJD_FEXP_SHIFT) /* 9.0 */
#define NJD_FEXP_10         (10<<NJD_FEXP_SHIFT) /* 10.0 */
#define NJD_FEXP_11         (11<<NJD_FEXP_SHIFT) /* 11.0 */
#define NJD_FEXP_12         (12<<NJD_FEXP_SHIFT) /* 12.0 */
#define NJD_FEXP_13         (13<<NJD_FEXP_SHIFT) /* 13.0 */
#define NJD_FEXP_14         (14<<NJD_FEXP_SHIFT) /* 14.0 */
#define NJD_FEXP_15         (15<<NJD_FEXP_SHIFT) /* 15.0 */
#define NJD_FEXP_16         (16<<NJD_FEXP_SHIFT) /* 16.0 */

#define NJD_FEXP_MASK        (0x1f<<NJD_FEXP_SHIFT)
```

3.6 Chunk Tiny

Chunk Tiny は、フラグと一つの値を設定するために利用します。これに対応するものとして TexId があります。形式は ChunkHead16bit と 16bit の定数の 32bit サイズ固定となります。

[headbits(15-8) | ChunkHead(7-0)][value(15-0)] (32 bits chunk)

現在定義されている Chunk Tiny には、TexId 及びテクスチャ関係のアトリビュートフラグを設定する NJD_CT_TID だけが定義されています。

ChunkName : 'NJD_CT_TID'

(Chunk Tiny TexId)

概要 :

plist において、TexList のエントリ番号である TexId を設定します。

形式 :

```
[headbits(15-8) | ChunkHead(7-0)][texbits(15-13) | TexId(12-0)]
headbits:
    15-14 = FlipUV(2)
    13-12 = ClampUV(2)
    11- 8 = Mipmap 'D' adjust(4)
ChunkHead:
    NJD_CT_TID
texbits:
    15-14 = Filter Mode(2)
    13    = Super Sample(1)
TexId:
    0 ~ 8191 (TexId Max = 8191)
```

説明 :

```
#define NJD_CT_TID (NJD_TINYOFF+0)
```

テクスチャ関連のアトリビュートフラグと TexId を設定します。テクスチャの切り替えタイミングでそれに関係するフラグを設定することにより、効率よくテクスチャの切り替えを行います。TexId に割り振られたビット数は 13bit のため、TexId の最大値は 8191 となる。

Flag FLip (headbits) :

```
#define NJD_FFL_U (BIT_15)
#define NJD_FFL_V (BIT_14)
```

UV 値のフリップを制御します。

Flag CLamp (headbits) :

```
#define NJD_FCL_U (BIT_13)
#define NJD_FCL_V (BIT_12)
```

UV 値のクランプを制御します。

Flag Filter Mode (texbits) :

PS : Point Sampled

BF : Bilinear Filter (default)

TF : Tri-linear Filter

```
#define NJD_FFM_SHIFT      14
#define NJD_FFM_PS        ( 0<<NJD_FFM_SHIFT )
#define NJD_FFM_BF        1<<NJD_FFM_SHIFT )
#define NJD_FFM_TF        ( 2<<NJD_FFM_SHIFT )

#define NJD_FFM_MASK      ( 0x3<<NJD_FFM_SHIFT )
```

フィルタリングモードを制御します。

Flag Super Sample (texbits) :

```
#define NJD_FSS            ( BIT_13 )
```

スーパーサンプリングを制御します。

```
#define NJD_TID_MASK      ( ~( NJD_FSS | NJD_FFM_MASK ) )
```

3.7 Chunk Material

Chunk Material は、diffuse、specular、ambient を設定します。その前に設定されている値から変化があった差分だけ設定されます。Chunk Strip(後述)の headbits で specular 無視のフラグ (NJD_FST_IL) が設定されている場合、specular の設定は省略されます。ambient 無視のフラグ (NJD_FST_IA) が設定されていると ambient の設定が省略されます。headbits (上位 8bit) に Blend Alpha を設定します。この部分は NJD_CB_BA と等価です。Blend Alpha の詳細については、NJD_CB_BA を参照してください。specular の最上位 8bit は、exponent 設定に使用されます。この部分は NJD_CB_EXP と機能的には等価です。NJD_CB_EXP では、0 ~ 16 までのフラグビット (NJD_FEXP_*) を用意しこれに設定していたが、Chunk Material の specular では 0 ~ 16 の値を直接設定します。

```
[headbits(13-8) | ChunkHead(7-0)][shortsize(15-0)][Data]
  headbits:
    13-11 = SRC Alpha Instruction(3)
    10- 8 = DST Alpha Instruction(3)
  ChunkHead:
    NJD_CM_D、NJD_CM_A、NJD_CM_DA、NJD_CM_S、
    NJD_CM_DS、NJD_CM_AS、NJD_CM_DAS
  Data:
    Diffulse = ARGB8888
    Specular = ERGB8888 (E : exponent 0 ~ 16)
    Ambient = NRGB8888 (N : NOOP)
```

ChunkName : 'NJD_CM_D'

(Chunk Material Diffuse)

概要 :

plist において diffuse の値を設定します。diffuse の上位 8bit には、値が格納されます。

形式 :

```
[headbits(13-8) | ChunkHead(7-0)][2(shortsize)][ARGB]
  headbits:
    13-11 = SRC Alpha Instruction(3)
    10- 8 = DST Alpha Instruction(3)
  ChunkHead:
    NJD_CM_D
  ARGB:
    ARGB8888
```

説明 :

```
#define NJD_CM_D (NJD_MATOFF+1)
```

Blend Alpha、diffuse のみを設定します。specular、ambient については、現在の設定値のままにします。

ChunkName : 'NJD_CM_A'

(Chunk Material Ambient)

概要 :

plist において ambient の値を設定します。ambient の上位 8bit は、ダミー (NOOP) として 255 が格納されます。

形式 :

```
[headbits(13-8) | ChunkHead(7-0)][2(shortsize)][NRGB]
  headbits:
    13-11 = SRC Alpha Instruction(3)
    10- 8 = DST Alpha Instruction(3)
  ChunkHead:
    NJD_CM_A
  NRGB:
    NRGB8888 (N : NOOP 255)
```

説明 :

```
#define NJD_CM_A (NJD_MATOFF+2)
```

Blend Alpha、ambient のみを設定します。diffuse、specular については現在の設定値のままにします。

ChunkName : 'NJD_CM_DA'

(Chunk Material Diffuse and Ambient)

概要 :

plist において diffuse、ambient の値を設定します。diffuse の上位 8bit には、値が設定されます。ambient の上位 8bit は、ダミー (NOOP) として 255 が格納されます。

形式 :

```
[headbits(13-8) | ChunkHead(7-0)][4(shortsize)][ARGB][NRGB]
headbits:
    13-11 = SRC Alpha Instruction(3)
    10- 8 = DST Alpha Instruction(3)
ChunkHead:
    NJD_CM_DA
ARGB:
    ARGB8888
NRGB:
    NRGB8888 (N : NOOP 255)
```

説明 :

```
#define NJD_CM_DA (NJD_MATOFF+3)
```

Blend Alpha、diffuse、ambient を設定します。specular については、現在の設定値のままにします。

ChunkName : 'NJD_CM_S'

(Chunk Material Specular)

概要 :

plist において specular の値を設定します。specular の上位 8bit には、exponent (0 ~ 16) が設定されます。

形式 :

```
[headbits(13-8) | ChunkHead(7-0)][2(shortsize)][ERGB]
  headbits:
    13-11 = SRC Alpha Instruction(3)
    10- 8 = DST Alpha Instruction(3)
  ChunkHead:
    NJD_CM_S
  ERGB:
    ERGB8888(E : Exponent 0 ~ 16)
```

説明 :

```
#define NJD_CM_S (NJD_MATOFF+4)
```

Blend Alpha、specular のみを設定します。diffuse、ambient については、現在の設定値のままにします。

ChunkName : 'NJD_CM_DS'

(Chunk Material Diffuse and Specular)

概要 :

plist において diffuse、specular の値を設定します。diffuse の上位 8bit には、値が設定されます。specular の上位 8bit には、exponent (0 ~ 16) が設定されます。

形式 :

```
[headbits(13-8) | ChunkHead(7-0)][4(shortsize)][ARGB][ERGB]
headbits:
    13-11 = SRC Alpha Instruction(3)
    10- 8 = DST Alpha Instruction(3)
ChunkHead:
    NJD_CM_DS
ARGB:
    ARGB8888
ERGB:
    ERGB8888(E : Exponent 0 ~ 16)
```

説明 :

```
#define NJD_CM_DS (NJD_MATOFF+5)
```

Blend Alpha、diffuse、specular を設定します。ambient については、現在の設定値のままにします。

ChunkName : 'NJD_CM_AS'

(Chunk Material Ambient and Specular)

概要 :

plist において diffuse、specular の値を設定します。diffuse の上位 8bit には、値が設定されます。specular の上位 8bit には、exponent (0 ~ 16) が設定されます。

形式 :

```
[headbits(13-8) | ChunkHead(7-0)][4(shortsize)][NRGB][ERGB]
headbits:
    13-11 = SRC Alpha Instruction(3)
    10- 8 = DST Alpha Instruction(3)
ChunkHead:
    NJD_CM_AS
NRGB:
    NRGB8888(N : NOOP 255)
ERGB:
    ERGB8888(E : Exponent 0 ~ 16)
```

説明 :

```
#define NJD_CM_AS (NJD_MATOFF+6)
```

Blend Alpha、ambient、specular を設定します。diffuse については、現在の設定値のままにします。

ChunkName : 'NJD_CM_DAS'

(Chunk Material Diffuse, Ambient and Specular)

概要 :

plist において diffuse、specular の値を設定します。diffuse の上位 8bit には、値が設定されます。specular の上位 8bit には、exponent (0 ~ 16) が設定されます。

形式 :

```
[headbits(13-8) | ChunkHead(7-0)][6(shortsize)][ARGB][NRGB][ERGB]
headbits:
    13-11 = SRC Alpha Instruction(3)
    10- 8 = DST Alpha Instruction(3)
ChunkHead:
    NJD_CM_DAS
ARGB:
    ARGB8888
NRGB:
    NRGB8888(N : NOOP 255)
ERGB:
    ERGB8888(E : Exponent 0 ~ 16)
```

説明 :

```
#define NJD_CM_DAS                (NJD_MATOFF+7)
```

Blend Alpha、diffuse、ambient、specular を設定します。

3.8 Chunk Vertex

Chunk Vertex はモデルの頂点リストを与えます。頂点リストにユーザの目的とするデータを格納するために、頂点リストもチャンク形式を利用します。チャンクの種類を切り替えることで、頂点、頂点法線、頂点カラー、ユーザフラグ等のデータをケースバイケースで頂点リストに持たせることができます。ChunkHead の上位 8bit の部分 (headbits) は、Chunk Vertex に関して未使用です。モデルに複数のチャンクタイプを同時に使用することは、ライブラリ側の処理を複雑にするため現在のところ対応していません。一つのモデルに対しては、一種類の Chunk Vertex を利用します。Sint16 ベースでヘッダは構成されますが、格納する配列が Sint32 のため Sint16 のデータ二つを Sint32 に合成して配置します。

[ChunkHead(31-16) | longsize(15-0)]

[IndexOffset(31-16) | nbIndices(15-0)][Data]

ChunkHead:

< SH4 で高速な計算が可能な形式 >

NJD_CV_SH、NJD_CV_VN_SH

< 標準的な形式 (頂点法線なし) >

NJD_CV、NJD_CV_D8、NJD_CV_UF、

NJD_CV_NF、NJD_CV_S5、NJD_CV_S4、NJD_CV_IN

< 標準的な形式 (頂点法線あり) >

NJD_CV_VN、NJD_CV_VN_D8、NJD_CV_VN_UF、

NJD_CV_VN_NF、NJD_CV_VN_S5、NJD_CV_VN_S4、

NJD_CV_VN_IN

< 32bit 頂点法線(x,y,z に各 10bit)形式 >

NJD_CV_VNX、NJD_CV_VNX_D8、NJD_CV_VNX_UF

略号の意味は以下のとおりです。

VN : use vertex normal

VNX : 32bits vertex normal reserved(2) | x(10) | y(10) | z(10)

SH : SH4 optimize

D8 : Diffuse ARGB8888 only

S5 : Diffuse RGB565 and Specular RGB565

S4 : Diffuse RGB4444 and Specular RGB565

IN : Diffuse(16) | Specular(16)

NF : NinjaFlags32 for extention

UF : UserFlags32

IndexOffset は、ライブラリの頂点中間バッファの使用開始位置オフセットを与えます。例えば、オフセット 0 である親ノードの頂点を計算し、その子供が親の頂点数と等しいオフセットを指定して、その位置から頂点計算結果を格納したとすると、親の頂点計算結果は上書きされずに中間バッファに残ります。この状態で親の頂点インデックスまでさかのぼる番号を指定することにより、親と子の頂点を結んだポリゴンを表現することができます。

nbIndices は、チャンクに格納される頂点数を与えます。

ChunkName : 'NJD_CV_SH'
(Chunk Vertex for SH4 Optimize)

概要 :

vlist において頂点リストを定義します。頂点法線なし。SH4 のマトリクス演算命令の特性を考慮したデータ配置により、計算処理を高速化します。

形式 :

```
[ChunkHead(31-16) | longsize(15-0)]  
[IndexOffset(31-16) | nbIndices(15-0)][Data]  
  ChunkHead:  
    NJD_CV_SH  
  longsize:  
    次のチャンクまでのオフセットです。  
  IndexOffset:  
    頂点中間バッファの開始位置を与えます。  
  nbIndices:  
    頂点の数を与えます。  
  Data:  
    x,y,z,1.0F, ...
```

説明 :

```
#define NJD_CV_SH (NJD_VERTOFF+0)
```

マトリクス演算命令に、128 ビット単位でデータを読み込むためのダミー1.0F を x,y,z の後ろに挿入します。そのままマトリクス演算が可能なため、高速処理を実現できます（その効果はこれから検証する予定です）。頂点法線なし。光計算を行わず頂点カラーだけで描画するモデルなどでは、頂点法線は不要です。このような場合に使用します。

ChunkName : 'NJD_CV_VN_SH'

(Chunk Vertex VertexNormal for SH4 Optimize)

概要 :

vlist において頂点リストを定義します。頂点法線あり。SH4 のマトリクス演算命令の特性を考慮したデータ配置により、計算処理を高速化します。

形式 :

```
[ChunkHead(31-16) | longsize(15-0)]
[IndexOffset(31-16) | nbIndices(15-0)] [Data]
  ChunkHead:
    NJD_CV_VN_SH
  longsize:
    次のチャンクまでのオフセットです。
  IndexOffset:
    頂点中間バッファの開始位置を与えます。
  nbIndices:
    頂点の数を与えます。
  Data:
    x,y,z,1.0F,nx,ny,nz,0.0F,...
```

説明 :

```
#define NJD_CV_VN_SH (NJD_VERTOFF+1)
```

マトリクス演算命令に、128 ビット単位でデータを読み込むためのダミー1.0F を x,y,z の後ろに、0.0F を法線 nx,ny,nz の後ろに挿入します。そのままマトリクス演算が可能のため、高速処理が実現できます (その効果はこれから検証する予定です)。

ChunkName : 'NJD_CV'

(Chunk Vertex)

概要 :

vlist において頂点リストを定義します。頂点法線なし。

形式 :

```
[ChunkHead(31-16) | longsize(15-0)]  
[IndexOffset(31-16) | nbIndices(15-0)] [Data]  
  ChunkHead:  
    NJD_CV  
  longsize:  
    次のチャンクまでのオフセットです。  
  IndexOffset:  
    頂点中間バッファの開始位置を与えます。  
  nbIndices:  
    頂点の数を与えます。  
  Data:  
    x,y,z, ...
```

説明 :

```
#define NJD_CV (NJD_VERTOFF+2)
```

頂点リストを与えます。頂点法線なし。

ChunkName : 'NJD_CV_D8'
(Chunk Vertex Diffuse ARGB8888)

概要 :

vlist において頂点リストを定義します。頂点法線なし。頂点カラーあり。

形式 :

```
[ChunkHead(31-16) | longsize(15-0)]  
[IndexOffset(31-16) | nbIndices(15-0)] [Data]  
  ChunkHead:  
    NJD_CV_D8  
  longsize:  
    次のチャンクまでのオフセットです。  
  IndexOffset:  
    頂点中間バッファの開始位置を与えます。  
  nbIndices:  
    頂点の数を与えます。  
  Data:  
    x,y,z,D8888,...
```

説明 :

```
#define NJD_CV_D8 (NJD_VERTOFF+3)
```

頂点リストを与えます。頂点法線なし。頂点カラーあり。頂点カラーは Sint32 にパックされます。

ChunkName : 'NJD_CV_UF'
(Chunk Vertex UserFlag)

概要 :

vlist において頂点リストを定義します。頂点法線なし。ユーザフラグ領域を 32bit 持ちます。

形式 :

```
[ChunkHead(31-16) | longsize(15-0)]  
[IndexOffset(31-16) | nbIndices(15-0)] [Data]  
  ChunkHead:  
    NJD_CV_UF  
  longsize:  
    次のチャンクまでのオフセットです。  
  IndexOffset:  
    頂点中間バッファの開始位置を与えます。  
  nbIndices:  
    頂点の数を与えます。  
  Data:  
    x,y,z,UserFlags32, ...
```

説明 :

```
#define NJD_CV_UF (NJD_VERTOFF+4)
```

頂点リストを与えます。頂点法線なし。ユーザフラグ領域を 32bit 持ちます。現在この領域に頂点カラーを出力できます。将来ツールからここにユーザデータを書き込めるようにする予定です。

ChunkName : 'NJD_CV_NF'
(Chunk Vertex NinjaFlags32)

概要 :

vlist において頂点リストを定義します。頂点法線なし。Ninja 拡張フラグ領域を 32bit 持ちます。

形式 :

```
[ChunkHead(31-16) | longsize(15-0)]  
[IndexOffset(31-16) | nbIndices(15-0)][Data]  
  ChunkHead:  
    NJD_CV_NF  
  longsize:  
    次のチャンクまでのオフセットです。  
  IndexOffset:  
    頂点中間バッファの開始位置を与えます。  
  nbIndices:  
    頂点の数を与えます。  
  Data:  
    x,y,z,NinjaFlags32, ...
```

説明 :

```
#define NJD_CV_NF (NJD_VERTOFF+5)
```

頂点リストを与えます。頂点法線なし。Ninja 拡張フラグ領域を 32bit 持ちます。Ninja 機能拡張のためにリザーブされます。

ChunkName : 'NJD_CV_S5'

(Chunk Vertex Diffuse RGB565 and Specular RGB565)

概要 :

vlist において頂点リストを定義します。頂点法線なし。頂点カラーとして diffuse、specular を持ちます。

形式 :

```
[ChunkHead(31-16) | longsize(15-0)]
[IndexOffset(31-16) | nbIndices(15-0)] [Data]
  ChunkHead:
    NJD_CV_S5
  longsize:
    次のチャンクまでのオフセットです。
  IndexOffset:
    頂点中間バッファの開始位置を与えます。
  nbIndices:
    頂点の数を与えます。
  Data:
    x,y,z,D565(31-16) | S565(15-0),...
```

説明 :

```
#define NJD_CV_S5 (NJD_VERTOFF+6)
```

頂点リストを与えます。頂点法線なし。頂点カラーあり。diffuse と specular を設定できます。表現効果向上の目的で、頂点カラーには specular が用意されています。現在、specular 頂点カラーの設定方法は用意されていません。コンバート時にシーンの光源位置から各頂点カラーの diffuse、specular を計算し、設定できるようにする (pre light) 予定です。

ChunkName : 'NJD_CV_S4'

(Chunk Vertex Specular RGB565 and Diffuse ARGB4444)

概要 :

vlist において頂点リストを定義します。頂点法線なし。頂点カラーとして diffuse、specular を持ちます。

形式 :

```
[ChunkHead(31-16) | longsize(15-0)]
[IndexOffset(31-16) | nbIndices(15-0)] [Data]
  ChunkHead:
    NJD_CV_S4
  longsize:
    次のチャンクまでのオフセットです。
  IndexOffset:
    頂点中間バッファの開始位置を与えます。
  nbIndices:
    頂点の数を与えます。
  Data:
    x,y,z,D4444(31-16) | S565(15-0),...
```

説明 :

```
#define NJD_CV_S4 (NJD_VERTOFF+7)
```

頂点リストを与えます。頂点法線なし。頂点カラーあり。diffuse と specular を設定できます。表現効果向上の目的で、頂点カラーには specular が用意されています。diffuse を ARGB4444 とし 値を設定できます。現在、specular 頂点カラーの設定方法は用意されていません。コンバート時にシーンの光源位置から各頂点カラーの diffuse、specular を計算し、設定できるようにする(pre light)予定です。

ChunkName : 'NJD_CV_IN'

(Chunk Vertex Intensity Diffuse and Specular)

概要 :

vlist において頂点リストを定義します。頂点法線なし。処理の高速な Intensity モードにおいて頂点カラーを与えます。頂点カラーとして diffuse、specular を持ちます。

形式 :

```
[ChunkHead(31-16) | longsize(15-0)]
[IndexOffset(31-16) | nbIndices(15-0)] [Data]
  ChunkHead:
    NJD_CV_IN
  longsize:
    次のチャンクまでのオフセットです。
  IndexOffset:
    頂点中間バッファの開始位置を与えます。
  nbIndices:
    頂点の数を与えます。
  Data:
    x,y,z,D16|S16,...
```

説明 :

```
#define NJD_CV_IN (NJD_VERTOFF+8)
```

頂点リストを与えます。頂点法線なし。Intensity モードにおいて頂点カラーを与えます。表現効果向上の目的で、頂点カラーには specular が用意されています。Intensity モードでは、diffuse と specular は輝度値だけで与えられます。その値をそれぞれ 16bit レンジで D16、S16 に設定します。現在、specular 頂点カラーの設定方法は用意されていません。コンバート時にシーンの光源位置から各頂点カラーの diffuse、specular を計算し、設定できるようにする(pre light)予定です。

ChunkName : 'NJD_CV_VN'
(Chunk Vertex VertexNormal)

概要 :

vlist において頂点リストを定義します。頂点法線あり。

形式 :

```
[ChunkHead(31-16) | longsize(15-0)]  
[IndexOffset(31-16) | nbIndices(15-0)] [Data]  
  ChunkHead:  
    NJD_CV_VN  
  longsize:  
    次のチャンクまでのオフセットです。  
  IndexOffset:  
    頂点中間バッファの開始位置を与えます。  
  nbIndices:  
    頂点の数を与えます。  
  Data:  
    x,y,z,nx,ny,nz, ...
```

説明 :

```
#define NJD_CV_VN (NJD_VERTOFF+9)
```

頂点リストを与えます。頂点法線あり。もっとも標準的な頂点リストであり頻繁に使用されます。

ChunkName : 'NJD_CV_VN_D8'

(Chunk Vertex VertexNormal and Diffuse ARGB8888)

概要 :

vlist において頂点リストを定義します。頂点法線あり。頂点カラーあり。

形式 :

```
[ChunkHead(31-16) | longsize(15-0)]
[IndexOffset(31-16) | nbIndices(15-0)] [Data]
```

ChunkHead:
NJD_CV_VN_D8

longsize:
次のチャンクまでのオフセットです。

IndexOffset:
頂点中間バッファの開始位置を与えます。

nbIndices:
頂点の数を与えます。

Data:
x,y,z,nx,ny,nz,D8888,...

説明 :

```
#define NJD_CV_VN_D (NJD_VERTOFF+10)
```

頂点リストを与えます。頂点法線あり。頂点カラーあり。頂点カラーは Sint32 にパックされます。

ChunkName : 'NJD_CV_VN_UF'

(Chunk Vertex VertexNormal and UserFlags32)

概要 :

vlist において頂点リストを定義します。頂点法線あり。ユーザフラグ領域を 32bit 持ちます。

形式 :

```
[ChunkHead(31-16) | longsize(15-0)]
[IndexOffset(31-16) | nbIndices(15-0)] [Data]
  ChunkHead:
    NJD_CV_VN_UF
  longsize:
    次のチャンクまでのオフセットです。
  IndexOffset:
    頂点中間バッファの開始位置を与えます。
  nbIndices:
    頂点の数を与えます。
  Data:
    x,y,z,nx,ny,nz,UserFlags32,...
```

説明 :

```
#define NJD_CV_VN_UF (NJD_VERTOFF+11)
```

頂点リストを与えます。頂点法線あり。ユーザフラグ領域を 32bit 持ちます。現在この領域に頂点カラーを出力できます。将来的には、ツールからここにユーザデータを書き込めるようにする予定です。

ChunkName : 'NJD_CV_VN_NF'
(Chunk Vertex VertexNormal and NinjaFlags32)

概要 :

vlist において頂点リストを定義します。頂点法線あり。Ninja 拡張フラグ領域を 32bit 持ちます。

形式 :

```
[ChunkHead(31-16) | longsize(15-0)]  
[IndexOffset(31-16) | nbIndices(15-0)] [Data]  
  ChunkHead:  
    NJD_CV_VN_NF  
  longsize:  
    次のチャンクまでのオフセットです。  
  IndexOffset:  
    頂点中間バッファの開始位置を与えます。  
  nbIndices:  
    頂点の数を与えます。  
  Data:  
    x,y,z,nx,ny,nz,NinjaFlags32,...
```

説明 :

```
#define NJD_CV_VN_NF (NJD_VERTOFF+12)
```

頂点リストを与えます。頂点法線あり。Ninja 拡張フラグ領域を 32bit 持ちます。Ninja 機能拡張のためにリザーブされます。

ChunkName : 'NJD_CV_VN_S5'

(Chunk Vertex VertexNormal, Diffuse RGB565 and Specular RGB565)

概要 :

vlist において頂点リストを定義します。頂点法線あり。頂点カラーとして diffuse、specular を持ちます。

形式 :

```
[ChunkHead(31-16) | longsize(15-0)]
[IndexOffset(31-16) | nbIndices(15-0)] [Data]
  ChunkHead:
    NJD_CV_VN_S5
  longsize:
    次のチャンクまでのオフセットです。
  IndexOffset:
    頂点中間バッファの開始位置を与えます。
  nbIndices:
    頂点の数を与えます。
  Data:
    x,y,z,nx,ny,nz,D565(31-16) | S565(15-0),...
```

説明 :

```
#define NJD_CV_VN_S5 (NJD_VERTOFF+13)
```

頂点リストを与えます。頂点法線あり。頂点カラーあり。頂点カラーは Sint32 にパックされます。diffuse と specular を設定できます。表現効果向上の目的で、頂点カラーには specular が用意されています。現在、specular 頂点カラーの設定方法は用意されていません。コンバート時にシーンの光源位置から各頂点カラーの diffuse、specular を計算し、設定できるようにする (pre light) 予定です。

ChunkName : 'NJD_CV_VN_S4'

(Chunk Vertex VertexNormal, Specular RGB565 and Diffuse ARGB4444)

概要 :

vlist において頂点リストを定義します。頂点法線あり。頂点カラーとして diffuse、specular を持ちます。

形式 :

```
[ChunkHead(31-16) | longsize(15-0)]  
[IndexOffset(31-16) | nbIndices(15-0)] [Data]  
  ChunkHead:  
    NJD_CV_VN_S4  
  longsize:  
    次のチャンクまでのオフセットです。  
  IndexOffset:  
    頂点中間バッファの開始位置を与えます。  
  nbIndices:  
    頂点の数を与えます。  
  Data:  
    x,y,z,nx,ny,nz,D4444(31-16) | S565(15-0),...
```

説明 :

```
#define NJD_CV_VN_S4 (NJD_VERTOFF+14)
```

頂点リストを与えます。頂点法線あり。頂点カラーあり。頂点カラーは Sint32 にパックされます。diffuse と specular を設定できます。表現効果向上の目的で、頂点カラーには specular が用意されています。diffuse を ARGB4444 とし、値を設定できます。現在、specular 頂点カラーの設定方法は用意されていません。コンバート時にシーンの光源位置から各頂点カラーの diffuse、specular を計算し、設定できるようにする(pre light)予定です。

ChunkName : 'NJD_CV_VN_IN'

(Chunk Vertex VertexNormal, Intensity Diffuse and Specular)

概要 :

vlist において頂点リストを定義します。頂点法線あり。処理の高速な Intensity モードにおいて頂点カラーを与えます。頂点カラーとして diffuse、specular を持ちます。

形式 :

```
[ChunkHead(31-16) | longsize(15-0)]
[IndexOffset(31-16) | nbIndices(15-0)] [Data]
  ChunkHead:
    NJD_CV_VN_IN
  longsize:
    次のチャンクまでのオフセットです。
  IndexOffset:
    頂点中間バッファの開始位置を与えます。
  nbIndices:
    頂点の数を与えます。
  Data:
    x,y,z,nx,ny,nz,D16|S16,...
```

説明 :

```
#define NJD_CV_VN_IN (NJD_VERTOFF+15)
```

頂点リストを与えます。頂点カラーあり。頂点カラーは Sint32 にパックされます。Intensity モードにおいて頂点カラーを与えます。表現効果向上の目的で、頂点カラーには specular が用意されています。Intensity モードでは、diffuse と specular は輝度値だけで与えられます。その値をそれぞれ 16bit レンジで D16、S16 に設定します。現在、specular 頂点カラーの設定方法は用意されていません。コンバート時にシーンの光源位置から各頂点カラーの diffuse、specular を計算し、設定できるようにする(pre light)予定です。

ChunkName : 'NJD_CV_VNX'

(Chunk Vertex VertexNormal 32bits(X))

概要 :

vlist において頂点リストを定義します。32bit 頂点法線あり。

形式 :

```
[ChunkHead(31-16) | longsize(15-0)]  
[IndexOffset(31-16) | nbIndices(15-0)] [Data]  
  ChunkHead:  
    NJD_CV_VNX  
  longsize:  
    次のチャンクまでのオフセットです。  
  IndexOffset:  
    頂点中間バッファの開始位置を与えます。  
  nbIndices:  
    頂点の数を与えます。  
  Data:  
    x,y,z,nxyz32, ...
```

説明 :

```
#define NJD_CV_VNX (NJD_VERTOFF+16)
```

頂点リストを与えます。32bit 頂点法線あり。頂点法線データを削減し、データ量を少なくします。x,y,z にそれぞれ 10bit を割り振ります。残り 2bit はリザーブされます。分解能は 1024 です。この分解能で求められた頂点法線で、グロー計算が行われます。

ChunkName : 'NJD_CV_VNX_D8'

(Chunk Vertex VertexNormal 32bits(X) and Diffuse ARGB8888)

概要 :

vlist において頂点リストを定義します。32bit 頂点法線あり。頂点カラーあり。

形式 :

```
[ChunkHead(31-16) | longsize(15-0)]
[IndexOffset(31-16) | nbIndices(15-0)] [Data]
  ChunkHead:
    NJD_CV_VNX_D8
  longsize:
    次のチャンクまでのオフセットです。
  IndexOffset:
    頂点中間バッファの開始位置を与えます。
  nbIndices:
    頂点の数を与えます。
  Data:
    x,y,z,nxyz32,D8888,...
```

説明 :

```
#define NJD_CV_VNX_D8 (NJD_VERTOFF+17)
```

頂点リストを与えます。32bit 頂点法線あり。頂点法線データを削減し、データ量を少なくします。x,y,z にそれぞれ 10bit を割り振ります。残り 2bit はリザーブされます。分解能は 1024 です。この分解能で求められた頂点法線で、グロー計算が行われます。頂点カラーあり。頂点カラーは Sint32 にパックされます。

ChunkName : 'NJD_CV_VNX_UF'

(Chunk Vertex VertexNormal 32bits(X) and UserFlags32)

概要 :

vlist において頂点リストを定義します。32bit 頂点法線あり。ユーザフラグ領域あり。

形式 :

```
[ChunkHead(31-16) | longsize(15-0)]
[IndexOffset(31-16) | nbIndices(15-0)] [Data]
  ChunkHead:
    NJD_CV_VNX_UF
  longsize:
    次のチャンクまでのオフセットです。
  IndexOffset:
    頂点中間バッファの開始位置を与えます。
  nbIndices:
    頂点の数を与えます。
  Data:
    x,y,z,nxyz32,UserFlags32,...
```

説明 :

```
#define NJD_CV_VNX_UF (NJD_VERTOFF+18)
```

頂点リストを与えます。32bit 頂点法線あり。頂点法線データを削減し、データ量を少なくします。x,y,z にそれぞれ 10bit を割り振ります。残り 2bit はリザーブされます。分解能は 1024 です。この分解能で求められた頂点法線で、グロー計算が行われます。ユーザフラグ領域を 32bit 持ちます。

3.9 Chunk Volume

Chunk Volume は、コリジョン、モディファイアボリューム用に用意されます。直接ライブラリで描画に使用されません。マテリアルデータは持ちません。現在 Chunk Volume は三種類用意されます。NJD_CO_P3 は独立三角形データから、また NJD_CO_P4 は独立四角形から構成されています。3D Studio MAX では、すべてデータが三角形データとして出力されますが NinjaExport では、オプションで面間角度が 0.1 度以下の面同士を接続して四角形を復元し、独立四角形として出力できます。これにより、3D Studio MAX においても四角形によるコリジョンを生成できます。また元データが三角形/四角形/N角形混在データの場合に、コンバータオプションですべてを三角形に分割し、再び独立三角形、四角形に再構成することができます。この場合、データには N 角形がなくなり、plist には NJD_CO_P3 と NJD_CO_P4 が並びます。NJD_CO_ST は、トライアングルストリップによる Chunk Volume です。Chunk Volume は Chunk Strip(後述)のものと、等価なポリゴンに対するユーザフラグ領域 (16, 32, 48bit) を持つことができます。またモデラーで設定されたマテリアルカラーを、各ポリゴンのユーザフラグ領域に出力することができます。

NJD_CO_P3 の独立三角形 Chunk Volume は、モディファイアボリュームにそのまま利用できます (ただしモディファイアボリュームに使用するには、3D 空間的に閉じたボリュームにする必要があります。ハード仕様。モディファイアボリュームは独立三角形しか使用できないことに注意してください)。ChunkHead の上位 8bit の部分 (headbits) は、Chunk Volume に関して未使用です。

```
[ChunkHead(15-0)][shortsize(15-0)]
[UserOffset(15-14)|nbPolygon(13-0)][Data]
```

ユーザフラグ領域は 16bit 単位 (plist が Sint16 の一次元配列のため) で扱われ、16、32、48bit の三つが選べます。この領域サイズの指定は、ポリゴン数を与える nbPolygon の最上位 2bit に割り振られた UserOffset で行われます。

UserFlags Offset:

```
#define NJD_UFO_SHIFT      14
#define NJD_UFO_0          (0<<NJD_UFO_SHIFT)
#define NJD_UFO_1          (1<<NJD_UFO_SHIFT)
#define NJD_UFO_2          (2<<NJD_UFO_SHIFT)
#define NJD_UFO_3          (3<<NJD_UFO_SHIFT)
#define NJD_UFO_MASK       (0x3<<NJD_UFO_SHIFT) /* 0 - 3 */
```

UserOffset=NJD_UFO_0 の時 UserFlags サイズは 0bit。
UserOffset=NJD_UFO_1 の時 UserFlags サイズは 16bit。
UserOffset=NJD_UFO_2 の時 UserFlags サイズは 32bit。
UserOffset=NJD_UFO_3 の時 UserFlags サイズは 48bit。

ChunkName : 'NJD_CO_P3'

(Chunk volume Polygon3)

概要 :

plist において volume 用ポリゴンリストを定義します。直接描画には使用されず、コリジョン、モディファイアボリュームに使用されます。マテリアルは持ちませんが、ユーザフラグ領域にポリゴンカラーを出力できます。

形式 :

```
[ChunkHead(15-0)][shortsize(15-0)]
  [UserOffset(15-14)|nbPolygon(13-0)][Data]
ChunkHead:
  NJD_CO_P3
shortsize:
  次のチャンクまでのオフセットです。
UserOffset:
  ユーザフラグ領域のサイズを与えます。
nbPolygon:
  ポリゴン数を与えます。
Data:
  index0, index1, index2, UserflagPoly0(*N),
  index3, index4, index5, UserflagPoly1(*N), ...
```

説明 :

```
#define NJD_CO_P3 (NJD_VOLOFF+0)
```

コリジョン、モディファイアボリューム用の独立三角形データです。マテリアルは持ちませんが、ユーザフラグ領域にポリゴンカラーを出力できます。このポリゴンカラーは、マテリアルで設定されたものです。ポリゴンインデックスの後ろにユーザフラグ領域を持ちます。このサイズは、UserOffset の値(NJD_UFO_0 ではなし、NJD_UFO_1 では 16bit、NJD_UFO_2 では 32bit、NJD_UFO_3 では 48bit)で決まります。ライブラリは、ユーザフラグ領域に関して何も関与せずスキップします。元データが三角形/四角形/N角形の混在する場合でも、コンバータオプションですべてを独立三角形に分割し、NJD_CO_P3 に出力することができます。

ChunkName : 'NJD_CO_P4'
(Chunk volume Polygon4)

概要 :

plist において volume 用ポリゴンリストを定義します。直接描画には使われずコリジョンに使用します。マテリアルは持ちませんが、ユーザフラグ領域にポリゴンカラーを出力できます。

形式 :

```
[ChunkHead(15-0)][shortsize(15-0)]
  [UserOffset(15-14)|nbPolygon(13-0)][Data]
ChunkHead:
  NJD_CO_P4
shortsize:
  次のチャンクまでのオフセットです。
UserOffset:
  ユーザフラグ領域のサイズを与えます。
nbPolygon:
  ポリゴン数を与えます。
Data:
  index0, index1, index2, index3, UserflagPoly0(*N),
  index4, index5, index6, index7, UserflagPoly1(*N), ...
```

説明 :

```
#define NJD_CO_P4 (NJD_VOLOFF+1)
```

コリジョン用の独立四角形データです。マテリアルは持ちませんが、ユーザフラグ領域にポリゴンカラーを出力できます。このポリゴンカラーはマテリアルで設定されたものです。3D Studio MAX ではオブジェクトカラーが利用されます。ポリゴンインデックスの後ろにユーザフラグ領域を持ちます。このサイズは、UserOffset の値(NJD_UFO_0 ではなし、NJD_UFO_1 では 16bit、NJD_UFO_2 では 32bit、NJD_UFO_3 では 48bit)で決まります。ライブラリは、ユーザフラグ領域に関して何も関与せずスキップします。3D Studio MAX の場合すべてのデータが独立三角形で出力されますが、コンバータオプションで面間角度が 0.1 度以下の場合二つの三角形を連結して四角形を再現し、これにより独立四角形データを生成できます。

ChunkName : 'NJD_CO_ST'
(Chunk vOlume Triangle Strip)

概要 :

plist において volume 用ポリゴンリストを定義します。直接描画には使用されず、コリジョンに使用されます。マテリアルは持ちませんが、ユーザフラグ領域にポリゴンカラーを出力できます。

形式 :

```
[ChunkHead(15-0)][shortsize(15-0)]
  [UserOffset(15-14)|nbPolygon(13-0)][Data]
ChunkHead:
  NJD_CO_ST
shortsize:
  次のチャンクまでのオフセットです。
UserOffset:
  ユーザフラグ領域のサイズを与えます。
nbPolygon:
  ポリゴン数を与えます。
Data:
  [flag(15)|len(14-0), i0, i1, i2, Userflag2(*N), i3, Userflag3(*N), ...]
```

説明 :

```
#define NJD_CO_ST (NJD_VOLOFF+2)
```

コリジョンとして使用されます。NJD_CO_P3 及び NJD_CO_P4 よりも、コリジョンデータサイズを小さくすることができます。ただしストリップでは、三角形が連結される方向はより効率良い方向に進むため、必ずしもユーザの意図した方向へは進まないのに注意してください。flag は、ストリップ先頭の三角形の回転方向(右回り/左回り)を与えます。ChunkModel は、符号により右回り/左回りを切り替えます。マイナス値の場合は右回りです。len はストリップに含まれる頂点数の数です。i?はポリゴン頂点インデックスを意味します。マテリアルは持ちませんが、ユーザフラグ領域にポリゴンカラーを出力できます。このポリゴンカラーは、マテリアルで設定されたものです。ポリゴン頂点インデックスの後ろにユーザフラグ領域を持ちます。このサイズは、UserOffset の値(NJD_UFO_0 ではなく、NJD_UFO_1 では 16bit、NJD_UFO_2 では 32bit、NJD_UFO_3 では 48bit)で決まります。ライブラリは、ユーザフラグ領域に関して何も関与せずスキップします。元データが三角形/四角形/N角形の混在する場合でも、コンバータは自動的にすべてを独立三角形に分割し、NJD_CO_ST に出力します。

3.10 Chunk Strip

Chunk Strip は、Chunk Vertex の頂点リストからライブラリ内に展開された頂点中間バッファの頂点エントリ番号指定により、トライアングルストリップを生成します。頂点カラー、頂点法線、ポリゴン単位の利用者フラグ領域を持つことができます。ポリゴン側に頂点カラーを持つことにより、同一頂点でもポリゴン単位に頂点カラーを設定できます。ポリゴン側に頂点法線を持つことで、ポリゴン間のエッジを立たせることができます。softimage の Discontinuity に対応でき、softimage で設定された頂点法線をそのまま出力することができます。利用者フラグ領域には、マテリアルから設定されたポリゴンカラーを出力できます。Chunk Vertex と Chunk Strip の両方に頂点カラーを持つことはできません。Chunk Strip 側に頂点カラー出力が指定された場合、Chunk Vertex 側の頂点カラーは出力されません。また Chunk Vertex と Chunk Strip の両方に頂点法線を持つことはできません。Chunk Strip 側に頂点法線出力が指定された場合、Chunk Vertex 側の頂点法線は出力されません。

ChunkHead の上位 8bit の部分 (headbits) には、マテリアルから設定されるアトリビュートフラグ (ChunkFlags) が設定されます。

略号の意味は次のとおりです。

IL: Ignore light
IS: Ignore specular
IA: Ignore ambient
UA: Use alpha
DB: Double side
FL: Flat shading
ENV: Environment mapping

Flag STRip :

```
#define NJD_FST_SHIFT      8
#define NJD_FST_IL        (0x01<<NJD_FST_SHIFT)
#define NJD_FST_IS        (0x02<<NJD_FST_SHIFT)
#define NJD_FST_IA        (0x04<<NJD_FST_SHIFT)
#define NJD_FST_UA        (0x08<<NJD_FST_SHIFT)
#define NJD_FST_DB        (0x10<<NJD_FST_SHIFT)
#define NJD_FST_FL        (0x20<<NJD_FST_SHIFT)
#define NJD_FST_ENV        (0x40<<NJD_FST_SHIFT)
#define NJD_FST_MASK      (0xFF<<NJD_FST_SHIFT)
```

次に Chunk Strip のフォーマットを示します。ポリゴン単位の利用者フラグが index2 以降に一つずつ挿入されていることに注意してください。これは三点目で始めて一つ目の三角形が生成されることによります。四点目以降は一点ごとに三角形が一つ生成されるため、一頂点ごとに頂点の後ろに利用者フラグが配置されています。

ポリゴン頂点法線なし、頂点カラーなし、テクスチャなしの場合

```
[ChunkFlags(15-8)|ChunkHead(7-0)]
[shortsize(15-0)][UserOffset(15-14)|nbStrip(13-0)]
[flag(15)|len(14-0), index0(15-0),
                                index1(15-0),
                                index2, UserFlag2(*N), ...]
```

ポリゴン頂点法線なし、頂点カラーなし、テクスチャありの場合

```
[ChunkFlags(15-8)|ChunkHead(7-0)]
[shortsize(15-0)][UserOffset(15-14)|nbStrip(13-0)]
[flag(15)|len(14-0), index0(15-0), U0(15-0), V0(15-0),
                                index1, U1, V1,
                                index2, U2, V2, UserFlag2(*N), ... ]
```

ポリゴン頂点法線あり、頂点カラーなし、テクスチャなしの場合

```
[ChunkFlags(15-8)|ChunkHead(7-0)]
[shortsize(15-0)][UserOffset(15-14)|nbStrip(13-0)]
[flag(15)|len(14-0), index0(15-0), vnx0(15-0), vny0(15-0), vnz0(15-0),
                                index1, vnx1, vny1, vnz1,
                                index2, vnx2, vny2, vnz2, UserFlag2(*N),
                                index3, vnx2, vny2, vnz2, UserFlag3(*N), ... ]
```

ポリゴン頂点法線あり、頂点カラーなし、テクスチャありの場合

```
[ChunkFlags(15-8)|ChunkHead(7-0)]
[shortsize(15-0)][UserOffset(15-14)|nbStrip(13-0)]
[flag(15)|len(14-0),
    index0(15-0), U0(15-0), V0(15-0),vnx0(15-0), vny0(15-0), vnz0(15-0),
    index1, U1, V1, vnx1, vny1, vnz1,
    index2, U2, V2, vnx2, vny2, vnz2, UserFlag2(*N),
    index3, U3, V3, vnx3, vny3, vnz3, UserFlag3(*N), ... ]
```

ポリゴン頂点法線なし、頂点カラーあり、テクスチャなしの場合

```
[ChunkFlags(15-8)|ChunkHead(7-0)]
[shortsize(15-0)][UserOffset(15-14)|nbStrip(13-0)]
[flag(15)|len(14-0),
    index0(15-0), AR0(15-0), GB0(15-0),
    index1, AR1, GB1,
    index2, AR2, GB2, UserFlag2(*N),
    index3, AR3, GB3, UserFlag3(*N), ... ]
```

ポリゴン頂点法線なし、頂点カラーあり、テクスチャありの場合

```
[ChunkFlags(15-8)|ChunkHead(7-0)]
[shortsize(15-0)][UserOffset(15-14)|nbStrip(13-0)]
[flag(15)|len(14-0),
    index0(16), U0(16), V0(16), AR0(16), GB0(16),
    index1, U1, V1, AR1, GB1,
    index2, U2, V2, AR2, GB2, UserFlag2(*N), ... ]
```

二種類の UV 値表現を持ちます。0-255 による UVN、分解能を高めた 0-1023 による UVH です。UVN は、Basic Model でも利用していた従来の表現です。256 を超えるテクスチャでは、1 ピクセル単位で指定できません。UVH はハイレゾモード、1024x1024 のテクスチャで、1 ピクセル単位での指定が可能です。ただし UVH は UVN よりも分解能を上げた分だけ、テクスチャのリピートの表現回数が減少します(UVN が 128 回に対し UVH では 32 回)。コンバートオプションによりモデルツリー全体に対しまして材料ネームにより各モデル単位で UVN、UVH を切り替えることができます。材料ネームからの指定では単一のモデルに使われる複数の材料のどれか一つに設定することでモデルに対する UV 表現が変更されます。デフォルトは UVN。

UVN : Normal type Uv (0-255)

UVH : Hiresolution type Uv (0-1023)

Chunk Volume と同様に、Chunk Strip でもユーザフラグ領域は nbStrip の最上位 2bit に割り付けられた UserOffset で与えられます。16bit 単位 (plist が Sint16 の一次元配列のため) で扱われ、16、32、48bit の三つが選べます。

UserFlags Offset:

```
#define NJD_UFO_SHIFT      14
#define NJD_UFO_0          (0<<NJD_UFO_SHIFT)
#define NJD_UFO_1          (1<<NJD_UFO_SHIFT)
#define NJD_UFO_2          (2<<NJD_UFO_SHIFT)
#define NJD_UFO_3          (3<<NJD_UFO_SHIFT)
#define NJD_UFO_MASK       (0x3<<NJD_UFO_SHIFT) /* 0 - 3 */
```

UserOffset=NJD_UFO_0 の時 UserFlags サイズは 0bit です。

UserOffset=NJD_UFO_1 の時 UserFlags サイズは 16bit です。

UserOffset=NJD_UFO_2 の時 UserFlags サイズは 32bit です。

UserOffset=NJD_UFO_3 の時 UserFlags サイズは 48bit です。

ChunkName : 'NJD_CS'

(Chunk Strip)

概要 :

plist においてポリゴンリストを定義します。ポリゴン頂点法線なし、頂点カラーなし、テクスチャなし。

形式 :

```
[ChunkFlags(15-8) | ChunkHead(7-0)]
[shortsize(15-0) | UserOffset(15-14) | nbStrip(13-0)] [Data]
ChunkFlags:
    NJD_FST_IL (光源無視)、NJD_FST_IS (スペキュラ無視)、
    NJD_FST_IA (アンビエント無視)、NJD_FST_UA (Use Alpha)、
    NJD_FST_DB (両面)、NJD_FST_FL (フラットシェーディング)、
    NJD_FST_ENV (環境マッピング)。
ChunkHead:
    NJD_CS
shortsize:
    次のチャンクまでのオフセットです。
UserOffset:
    ユーザフラグ領域のサイズを与えます。
nbStrip:
    ストリップの頂点数です。
Data:
    [flag(15) | len(14-0), index0(15-0),
                                     index1(15-0),
                                     index2, UserFlag2(*N), ...]
```

説明 :

```
#define NJD_CS (NJD_STRIPOFF+0)
```

flag は、ストリップ先頭の三角形の回転方向(右回り/左回り)を与えます。ChunkModel は、符号により右回り/左回りを切り替えます。マイナス値の場合は右回りです。len はストリップに含まれる頂点数の数です。index?はポリゴン頂点インデックスを意味します。ポリゴン頂点インデックスの後ろにユーザフラグ領域を持ちます。このサイズは、UserOffset の値(NJD_UFO_0 ではなし、NJD_UFO_1 では 16bit、NJD_UFO_2 では 32bit、NJD_UFO_3 では 48bit)で決まります。ライブラリは、ユーザフラグ領域に関して何も関与せずスキップします。三角形/四角形/N角形を自動的に独立三角形に分割し、ストリップに変換して出力します。

ChunkName : 'NJD_CS_UVN'

(Chunk Strip UVN)

概要 :

plist においてポリゴンリストを定義します。ポリゴン頂点法線なし。頂点カラーなし。テクスチャあり。UV 値は UVN(0-255)で与えられます。

形式 :

```
[ChunkFlags(15-8) | ChunkHead(7-0)]
[shortsize(15-0)][UserOffset(15-14) | nbStrip(13-0)][Data]
ChunkFlags:
    NJD_FST_IL(光源無視)、NJD_FST_IS(スペキュラ無視)、
    NJD_FST_IA(アンビエント無視)、NJD_FST_UA(Use Alpha)、
    NJD_FST_DB(両面)、NJD_FST_FL(フラットシェーディング)、
    NJD_FST_ENV(環境マッピング)。
ChunkHead:
    NJD_CS_UVN
shortsize:
    次のチャンクまでのオフセットです。
UserOffset:
    ユーザフラグ領域のサイズを与えます。
nbStrip:
    ストリップの頂点数です。
Data:
    [flag(15) | len(14-0), index0(15-0), U0(15-0), V0(15-0),
      index1, U1, V1,
      index2, U2, V2, UserFlag2(*N), ... ]
```

説明 :

```
#define NJD_CS_UVN (NJD_STRIPOFF+1)
```

UV 値には UVN (0-255) が使用されます。flag は、ストリップ先頭の三角形の回転方向(右回り/左回り)を与えます。ChunkModel は、符号により右回り/左回りを切り替えます。マイナス値の場合は右回りです。len はストリップに含まれる頂点数の数。index?はポリゴン頂点インデックスを意味します。ポリゴン頂点インデックスの後ろにユーザフラグ領域を持ちます。このサイズは、UserOffset の値(NJD_UFO_0 ではなし、NJD_UFO_1 では 16bit、NJD_UFO_2 では 32bit、NJD_UFO_3 では 48bit)で決まります。ライブラリは、ユーザフラグ領域に関して何も関与せずスキップします。三角形/四角形/N角形を自動的に独立三角形に分割し、ストリップに変換して出力します。

ChunkName : 'NJD_CS_UVH'

(Chunk Strip UVH)

概要 :

plist においてポリゴンリストを定義します。ポリゴン頂点法線なし。頂点カラーなし。テクスチャあり。UV 値は UVH(0-1023)で与えられます。

形式 :

```
[ChunkFlags(15-8) | ChunkHead(7-0)]
[shortsize(15-0)][UserOffset(15-14) | nbStrip(13-0)][Data]
ChunkFlags:
    NJD_FST_IL(光源無視)、NJD_FST_IS(スペキュラ無視)、
    NJD_FST_IA(アンビエント無視)、NJD_FST_UA(Use Alpha)、
    NJD_FST_DB(両面)、NJD_FST_FL(フラットシェーディング)、
    NJD_FST_ENV(環境マッピング)。
ChunkHead:
    NJD_CS_UVH
shortsize:
    次のチャンクまでのオフセットです。
UserOffset:
    ユーザフラグ領域のサイズを与えます。
nbStrip:
    ストリップの頂点数です。
Data:
    [flag(15) | len(14-0), index0(15-0), U0(15-0), V0(15-0),
      index1, U1, V1,
      index2, U2, V2, UserFlag2(*N), ... ]
```

説明 :

```
#define NJD_CS_UVH (NJD_STRIPOFF+2)
```

UV 値には UVH (0-1023) が使用されます。UVN に比べリピートの上限値が低くなっている (32 回) ことに注意してください。flag は、ストリップ先頭の三角形の回転方向(右回り/左回り)を与えます。ChunkModel は、符号により右回り/左回りを切り替える。マイナス値の場合は右回りです。len は、ストリップに含まれる頂点数の数です。index?はポリゴン頂点インデックスを意味します。ポリゴン頂点インデックスの後ろにユーザフラグ領域を持ちます。このサイズは、UserOffset の値(NJD_UFO_0 ではなし、NJD_UFO_1 では 16bit、NJD_UFO_2 では 32bit、NJD_UFO_3 では 48bit)で決まります。ライブラリは、ユーザフラグ領域に関して何も関与せずスキップします。三角形/四角形/N角形を自動的に独立三角形に分割し、ストリップに変換して出力します。

ChunkName : 'NJD_CS_VN'
(Chunk Strip VertexNormal)

概要 :

plist においてポリゴンリストを定義します。ポリゴン頂点法線あり。頂点カラーなし。テクスチャなし。

形式 :

```
[ChunkFlags(15-8) | ChunkHead(7-0)]
[shortsize(15-0)][UserOffset(15-14) | nbStrip(13-0)][Data]
  ChunkFlags:
    NJD_FST_IL(光源無視)、NJD_FST_IS(スペキュラ無視)、
    NJD_FST_IA(アンビエント無視)、NJD_FST_UA(Use Alpha)、
    NJD_FST_DB(両面)、NJD_FST_FL(フラットシェーディング)、
    NJD_FST_ENV(環境マッピング)。
  ChunkHead:
    NJD_CS_VN
  shortsize:
    次のチャンクまでのオフセットです。
  UserOffset:
    ユーザフラグ領域のサイズを与えます。
  nbStrip:
    ストリップの頂点数です。
  Data:
    [flag(15) | len(14-0), index0(15-0), vnx0(15-0), vny0(15-0), vnz0(15-0),
      index1, vnx1, vny1, vnz1,
      index2, vnx2, vny2, vnz2, UserFlag2(*N),
      index3, vnx2, vny2, vnz2, UserFlag3(*N), ... ]
```

説明 :

```
#define NJD_CS_VN (NJD_STRIPOFF+3)
```

flag はストリップ先頭の三角形の回転方向(右回り/左回り)を与えます。ChunkModel は符号により右回り/左回りを切り替えます。マイナス値の場合は右回りです。len はストリップに含まれる頂点数の数です。index?はポリゴン頂点インデックスを意味します。ポリゴン頂点インデックスの後ろにユーザフラグ領域を持ちます。このサイズは、UserOffset の値(NJD_UFO_0 ではなく、NJD_UFO_1 では 16bit、NJD_UFO_2 では 32bit、NJD_UFO_3 では 48bit)で決まります。ライブラリは、ユーザフラグ領域に関して何も関与せずスキップします。三角形/四角形/N角形を自動的に独立三角形に分割し、ストリップに変換して出力します。

ChunkName : 'NJD_CS_UVN_VN'

(Chunk Strip UVN VertexNormal)

概要 :

plist においてポリゴンリストを定義します。ポリゴン頂点法線あり。頂点カラーなし。テクスチャあり。UV 値は UVN(0-255)で与えられます。

形式 :

```
[ChunkFlags(15-8)|ChunkHead(7-0)]
[shortsize(15-0)][UserOffset(15-14)|nbStrip(13-0)][Data]
ChunkFlags:
    NJD_FST_IL(光源無視)、NJD_FST_IS(スペキュラ無視)、
    NJD_FST_IA(アンビエント無視)、NJD_FST_UA(Use Alpha)、
    NJD_FST_DB(両面)、NJD_FST_FL(フラットシェーディング)、
    NJD_FST_ENV(環境マッピング)。
ChunkHead:
    NJD_CS_UVN_VN
shortsize:
    次のチャンクまでのオフセットです。
UserOffset:
    ユーザフラグ領域のサイズを与えます。
nbStrip:
    ストリップの頂点数です。
Data:
    [flag(15)|len(14-0),
    index0(15-0), U0(15-0), V0(15-0), vnx0(15-0), vny0(15-0), vnz0(15-0),
    index1, U1, V1, vnx1, vny1, vnz1,
    index2, U2, V2, vnx2, vny2, vnz2, UserFlag2(*N),
    index3, U3, V3, vnx3, vny3, vnz3, UserFlag3(*N), ... ]
```

説明 :

```
#define NJD_CS_UVN_VN (NJD_STRIPOFF+4)
```

UV 値には UVN (0-255) が使用されます。flag はストリップ先頭の三角形の回転方向(右回り/左回り)を与えます。ChunkModel は符号により右回り/左回りを切り替えます。マイナス値の場合は右回りです。len はストリップに含まれる頂点数の数です。index?はポリゴン頂点インデックスを意味します。ポリゴン頂点インデックスの後ろにユーザフラグ領域を持ちます。このサイズは、UserOffset の値(NJD_UFO_0 ではなく、NJD_UFO_1 では 16bit、NJD_UFO_2 では 32bit、NJD_UFO_3 では 48bit)で決まります。ライブラリは、ユーザフラグ領域に関して何も関与せずスキップします。三角形/四角形/N角形を自動的に独立三角形に分割し、ストリップに変換して出力します。

ChunkName : 'NJD_CS_UVH_VN'

(Chunk Strip UVH VertexNormal)

概要 :

plist においてポリゴンリストを定義します。ポリゴン頂点法線あり。頂点カラーなし。テクスチャあり。UV 値は UVH(0-1023)で与えられます。

形式 :

```
[ChunkFlags(15-8)|ChunkHead(7-0)]
[shortsize(15-0)][UserOffset(15-14)|nbStrip(13-0)][Data]
ChunkFlags:
    NJD_FST_IL(光源無視)、NJD_FST_IS(スペキュラ無視)、
    NJD_FST_IA(アンビエント無視)、NJD_FST_UA(Use Alpha)、
    NJD_FST_DB(両面)、NJD_FST_FL(フラットシェーディング)、
    NJD_FST_ENV(環境マッピング)。
ChunkHead:
    NJD_CS_UVH_VN
shortsize:
    次のチャンクまでのオフセットです。
UserOffset:
    ユーザフラグ領域のサイズを与えます。
nbStrip:
    ストリップの頂点数です。
Data:
    [flag(15)|len(14-0),
    index0(15-0), U0(15-0), V0(15-0),vnx0(15-0), vny0(15-0), vnz0(15-0),
    index1, U1, V1, vnx1, vny1, vnz1,
    index2, U2, V2, vnx2, vny2, vnz2, UserFlag2(*N),
    index3, U3, V3, vnx3, vny3, vnz3, UserFlag3(*N), ... ]
```

説明 :

```
#define NJD_CS_UVH_VN (NJD_STRIPOFF+5)
```

UV 値には UVH (0-1023) が使用されます。UVN に比べリピートの上限値が低くなっている (32 回) ことに注意してください。flag は、ストリップ先頭の三角形の回転方向(右回り/左回り)を与えます。ChunkModel は符号により右回り/左回りを切り替えます。マイナス値の場合は右回りです。len はストリップに含まれる頂点数の数です。index?はポリゴン頂点インデックスを意味します。ポリゴン頂点インデックスの後ろにユーザフラグ領域を持ちます。このサイズは、UserOffset の値(NJD_UFO_0 ではなし、NJD_UFO_1 では 16bit、NJD_UFO_2 では 32bit、NJD_UFO_3 では 48bit)で決まります。ライブラリは、ユーザフラグ領域に関して何も関与せずスキップします。三角形/四角形/N角形を自動的に独立三角形に分割し、ストリップに変換して出力します。

ChunkName : 'NJD_CS_D8'
(Chunk Strip Diffuse ARGB8888)

概要 :

plist においてポリゴンリストを定義します。ポリゴン頂点法線なし。頂点カラーあり。テクスチャなし。

形式 :

```
[ChunkFlags(15-8) | ChunkHead(7-0)]  
[shortsize(15-0)][UserOffset(15-14) | nbStrip(13-0)][Data]  
  ChunkFlags:  
    NJD_FST_IL(光源無視)、NJD_FST_IS(スペキュラ無視)、  
    NJD_FST_IA(アンビエント無視)、NJD_FST_UA(Use Alpha)、  
    NJD_FST_DB(両面)、NJD_FST_FL(フラットシェーディング)、  
    NJD_FST_ENV(環境マッピング)。  
  ChunkHead:  
    NJD_CS_D8  
  shortsize:  
    次のチャンクまでのオフセットです。  
  UserOffset:  
    ユーザフラグ領域のサイズを与えます。  
  nbStrip:  
    ストリップの頂点数です。  
  Data:  
    [flag(15) | len(14-0),  
    index0(15-0), AR0(15-0), GB0(15-0),  
    index1, AR1, GB1,  
    index2, AR2, GB2, UserFlag2(*N),  
    index3, AR3, GB3, UserFlag3(*N), ... ]
```

説明 :

```
#define NJD_CS_D8 (NJD_STRIPOFF+6)
```

flag は、ストリップ先頭の三角形の回転方向(右回り/左回り)を与えます。ChunkModel は、符号により右回り/左回りを切り替える。マイナス値の場合は右回りです。len はストリップに含まれる頂点数の数です。index?はポリゴン頂点インデックスを意味します。ポリゴン頂点インデックスの後ろにユーザフラグ領域を持ちます。このサイズは、UserOffset の値(NJD_UFO_0 ではなし、NJD_UFO_1 では 16bit、NJD_UFO_2 では 32bit、NJD_UFO_3 では 48bit)で決まります。ライブラリは、ユーザフラグ領域に関して何も関与せずスキップします。三角形/四角形/N角形を自動的に独立三角形に分割し、ストリップに変換して出力します。

ChunkName : 'NJD_CS_UVN_D8'
(Chunk Strip UVN Diffuse ARGB8888)

概要 :

plist においてポリゴンリストを定義します。ポリゴン頂点法線なし。頂点カラーあり。テクスチャあり。UV 値は UVN(0-255)で与えられます。

形式 :

```
[ChunkFlags(15-8)|ChunkHead(7-0)]
[shortsize(15-0)][UserOffset(15-14)|nbStrip(13-0)][Data]
ChunkFlags:
    NJD_FST_IL(光源無視)、NJD_FST_IS(スペキュラ無視)、
    NJD_FST_IA(アンビエント無視)、NJD_FST_UA(Use Alpha)、
    NJD_FST_DB(両面)、NJD_FST_FL(フラットシェーディング)、
    NJD_FST_ENV(環境マッピング)。
ChunkHead:
    NJD_CS_UVN_D8
shortsize:
    次のチャンクまでのオフセットです。
UserOffset:
    ユーザフラグ領域のサイズを与えます。
nbStrip:
    ストリップの頂点数です。
Data:
[ChunkFlags(15-8)|ChunkHead(7-0)]
[shortsize(15-0)][UserOffset(15-14)|nbStrip(13-0)]
[flag(15)|len(14-0),
 index0(16), U0(16), V0(16), AR0(16), GB0(16),
 index1, U1, V1, AR1, GB1,
 index2, U2, V2, AR2, GB2, UserFlag2(*N), ... ]
```

説明 :

```
#define NJD_CS_UVN_D8 (NJD_STRIPOFF+7)
```

UV 値には UVN (0-255) が使用されます。flag は、ストリップ先頭の三角形の回転方向(右回り/左回り)を与えます。ChunkModel は、符号により右回り/左回りを切り替えます。マイナス値の場合は右回りです。len はストリップに含まれる頂点数の数です。index?はポリゴン頂点インデックスを意味します。ポリゴン頂点インデックスの後ろにユーザフラグ領域を持ちます。このサイズは、UserOffset の値(NJD_UFO_0 ではなく、NJD_UFO_1 では 16bit、NJD_UFO_2 では 32bit、NJD_UFO_3 では 48bit)で決まります。ライブラリはユーザフラグ領域に関して何も関与せずスキップします。三角形/四角形/N角形を自動的に独立三角形に分割し、ストリップに変換して出力します。

ChunkName : 'NJD_CS_UVH_D8'
(Chunk Strip UVH Diffuse ARGB8888)

概要 :

plist においてポリゴンリストを定義します。ポリゴン頂点法線なし。頂点カラーあり。テクスチャあり。UV 値は UVH(0-1023)で与えられます。

形式 :

```
[ChunkFlags(15-8)|ChunkHead(7-0)]
[shortsize(15-0)][UserOffset(15-14)|nbStrip(13-0)][Data]
ChunkFlags:
    NJD_FST_IL(光源無視)、NJD_FST_IS(スペキュラ無視)、
    NJD_FST_IA(アンビエント無視)、NJD_FST_UA(Use Alpha)、
    NJD_FST_DB(両面)、NJD_FST_FL(フラットシェーディング)、
    NJD_FST_ENV(環境マッピング)。
ChunkHead:
    NJD_CS_UVH_D8
shortsize:
    次のチャンクまでのオフセットです。
UserOffset:
    ユーザフラグ領域のサイズを与えます。
nbStrip:
    ストリップの頂点数です。
Data:
    [ChunkFlags(15-8)|ChunkHead(7-0)]
    [shortsize(15-0)][UserOffset(15-14)|nbStrip(13-0)]
    [flag(15)|len(14-0),
     index0(16), U0(16), V0(16), AR0(16), GB0(16),
     index1, U1, V1, AR1, GB1,
     index2, U2, V2, AR2, GB2, UserFlag2(*N), ... ]
```

説明 :

```
#define NJD_CS_UVH_D8 (NJD_STRIPOFF+8)
```

UV 値には UVH (0-1023) が使用されます。UVN に比べリピートの上限値が低くなっている (32 回) ことに注意してください。flag は、ストリップ先頭の三角形の回転方向(右回り/左回り)を与えます。ChunkModel は、符号により右回り/左回りを切り替えます。マイナス値の場合は右回りです。len はストリップに含まれる頂点数の数です。index?はポリゴン頂点インデックスを意味します。ポリゴン頂点インデックスの後ろにユーザフラグ領域を持ちます。このサイズは、UserOffset の値(NJD_UFO_0 ではなし、NJD_UFO_1 では 16bit、NJD_UFO_2 では 32bit、NJD_UFO_3 では 48bit)で決まります。ライブラリは、ユーザフラグ領域に関して何も関与せずスキップします。三角形/四角形/N角形を自動的に独立三角形に分割し、ストリップに変換して出力します。

4. アスキー出力の注意事項

Chunk Model .nja ファイル出力に対する注意事項をまとめます。

vlist は Sint32 の配列のため、この中に float 値の頂点、頂点法線をそのまま入力できません。そのためアスキー出力では、ヘキサ表現になっています。もし float として値を確認したい場合は、コンバータオプションによりコメントとして float 値を出力できます。

Chunk Model では、フラグもすべて文字列で出力します。なるべく短くかつ一般に使われない文字列を心がけました。**NJD をはずした文字列がそのまま nja ファイルに利用されています。**詳細は NjDef.h を参照してください。