

# G3Dバイナリファイルフォーマット

2008-04-08

任天堂株式会社発行

このドキュメントの内容は、機密情報であるため、  
厳重な取り扱い、管理を行ってください。

## 目次

1	はじめに .....	5
2	G3D バイナリフォーマットの特徴 .....	5
3	G3D バイナリファイルフォーマットの解説 .....	6
3.1	全てのバイナリファイルで用いられるデータ構造 .....	6
3.1.1	ファイルヘッダ .....	6
3.1.2	データブロックヘッダ .....	7
3.1.3	ディクショナリ .....	7
3.1.4	アニメーションヘッダ .....	9
3.2	モデルデータファイル(.nsbmd)の構造 .....	10
3.2.1	モデルブロック .....	10
3.2.1.1	モデルの集合 .....	10
3.2.1.2	モデル .....	11
3.2.1.3	モデルに関する基本的な情報 .....	11
3.2.1.4	ノードの情報 .....	13
3.2.1.5	マテリアルの情報 .....	14
3.2.1.6	シェイプの集合とシェイプ .....	17
3.2.1.7	ノード・マテリアル・シェイプの関連付け情報 .....	18
3.2.1.8	エンベロープ計算用の行列格納領域 .....	23
3.2.2	テクスチャ・パレットブロック .....	24
3.2.2.1	テクスチャとパレットの集合 .....	24
3.3	ジョイントアニメーションデータファイル(.nsbca)の構造 .....	28
3.4	テクスチャパターンアニメーションデータファイル(.nsbtp)の構造 .....	34
3.5	マテリアルカラーアニメーションデータファイル(.nsbma)の構造 .....	36
3.6	ビジビリティアニメーションデータファイル(.nsbva)の構造 .....	38
3.7	テクスチャSRTアニメーションデータファイル(.nsbta)の構造 .....	39

## 表

表 3-1	FileHeaderのデータメンバの解説 .....	6
表 3-2	DataBlockHeaderのデータメンバの解説 .....	7
表 3-3	Dictionaryのデータメンバの解説 .....	8
表 3-4	AnmHeaderのデータメンバの解説 .....	9
表 3-5	NSBMDのデータメンバの解説 .....	10
表 3-6	ModelSetのデータメンバの解説 .....	10
表 3-7	Modelのデータメンバの解説 .....	11
表 3-8	ModelInfoのデータメンバの解説 .....	12
表 3-9	scalingRuleがとる値 .....	12
表 3-10	texMtxModeがとる値 .....	13
表 3-11	NodeDataのflagフィールドの値 .....	13

表 3-12	NodeDataのデータメンバの解説	14
表 3-13	MaterialSetのデータメンバ	14
表 3-14	Materialのデータメンバ	16
表 3-15	Materialのflagメンバの値	17
表 3-16	ShapeSetのデータメンバ	18
表 3-17	ShapeSet::Shape::flagメンバの値	18
表 3-18	SBCコマンド一覧	18
表 3-19	EvpMatricesのデータメンバ	24
表 3-20	TexPlttSetのデータメンバの解説	25
表 3-21	TexPlttSet::TexInfo::flagメンバの値	26
表 3-22	TexPlttSet::Tex4x4Info::flagメンバの値	26
表 3-23	TexPlttSet::PlttInfo::flagメンバの値	26
表 3-24	ディクショナリdictTex内に格納されているデータ	27
表 3-25	ディクショナリdictPltt内に格納されているデータ	27
表 3-26	JointAnmSetのデータメンバの解説	28
表 3-27	JointAnmのデータメンバの解説	31
表 3-28	JointAnmTransのデータメンバの解説	31
表 3-29	JointAnmRotのデータメンバの解説	31
表 3-30	JointAnmScaleのデータメンバの解説	32
表 3-31	JointAnm::TagData::flagがとる値の解説	32
表 3-32	JointAnmTrans::infoがとる値の解説	32
表 3-33	JointAnmRot::infoがとる値の解説	33
表 3-34	JointAnmScale::infoがとる値の解説	33
表 3-35	TexPatAnmSetのデータメンバの解説	34
表 3-36	TexPatAnmのデータメンバの解説	35
表 3-37	ディクショナリTexPatAnm::dictに格納されているデータ	35
表 3-38	MatColAnmSetのデータメンバの解説	36
表 3-39	MatColAnmのデータメンバの解説	36
表 3-40	MatColAnm::flagがとる値の解説	37
表 3-41	DictMatColAnmDataのデータメンバの解説	37
表 3-42	tagDiffuse/tagAmbient/tagSpecular/tagEmission/tagPolygonAlphaがとる値の解説	37
表 3-43	VisAnmSetのデータメンバの解説	38
表 3-44	VisAnmのデータメンバの解説	38
表 3-45	TexSRTAnmSetのデータメンバの解説	39
表 3-46	TexSRTAnmのデータメンバの解説	39
表 3-47	TexSRTAnm::flagがとる値の解説	40
表 3-48	DictTexSRTAnmDataのデータメンバの解説	40
表 3-49	scaleS/scaleT/rot/transS/transTがとる値の解説	41

## 改訂履歴

改訂日	改訂内容
2008-04-08	改訂履歴の書式を変更。
2005-05-11	XXX_LAST_INTERP_MASK の説明の修正。
2005-01-19	環境マップ・投影マップ用の拡張に対応。
2004-12-03	擬似構造体表記の導入他。
2004-08-19	図表番号を挿入。
2004-08-09	$\beta$ 版作成。
2004-07-23	$\alpha$ 版作成。

# 1 はじめに

G3D ライブラリでモデルを表示したりアニメーションを再生するためには、`g3dcvtr` によって NITRO 中間ファイルをバイナリファイルに変換する必要があります。このドキュメントでは `g3dcvtr` によって変換されたバイナリファイルのフォーマットについて説明いたします。なお、バイナリフォーマットの仕様は予告なく変更されたり拡張されたりする場合がありますのでご了承ください。

## 2 G3Dバイナリフォーマットの特徴

G3D で使用するバイナリフォーマットには以下のような特徴があります。

- 複数のモデルデータを1つのバイナリファイルに格納できるようになっています。アニメーションデータも同様です。
- G3D ライブラリはメモリにロードされたバイナリファイルを直接処理することができます。G3D の内部データ構造とバイナリファイルフォーマットが一致しているので、初期化時におけるバイナリファイルフォーマットからメモリ内オブジェクトへの変換作業が必要ありません。よって、ロードと初期化時のオーバーヘッドが低く、追加的なメモリアロケートを行う必要がありません。
- 内部にポインタが存在しません。バイナリフォーマット内のリンクは全てファイル内の個々のブロックの先頭を基準にしたオフセットとして表現されています。
- リソースの名前による検索を行うための辞書を保持しています。名前辞書はコンパクトかつ高速な検索が可能で、名前によるリソース検索を効率よく行うことができます。
- 描画時の計算コストを減らすことができるように、`g3dcvtr` によって計算及び整列されたデータを各種保持するようになっています。

## 3 G3D バイナリファイルフォーマットの解説

### 3.1 全てのバイナリファイルで用いられるデータ構造

ここで、`pseudo_struct` とあるのは、可変長の配列が可能だったり、条件によって構造体のデータメンバを変化させたりすることのできる、本ドキュメント用の擬似的な構造体です。擬似構造体名の隣のカッコ内には、実際に G3D で定義されている対応する構造体名が書かれています。

#### 3.1.1 ファイルヘッダ

バイナリファイルの先頭には、ファイルの種別等を表す以下のようなデータ構造が格納されています。

```
pseudo_struct FileHeader {  
    pseudo_struct HeaderInfo(NNSG3dResFileHeader) {  
        u32 signature;  
        u16 byteOrder = 0xffef;  
        u16 version;  
        u32 fileSize;  
        u16 headerSize = 16;  
        u16 dataBlocks;  
    } info;  
    u32 offset[dataBlocks];  
};
```

表 3-1 FileHeader のデータメンバの解説

名称	内容
signature	バイナリファイルの種別判定用定数(4 文字)。
byteOrder	エンディアン判定用の数。リトルエンディアンの場合 0xffef である。
version	バイナリファイルのバージョン(1.2 なら 0x0102)。
fileSize	ファイルサイズ。
headerSize	HeaderInfo のサイズ。G3D の場合は 16 で固定である。
dataBlocks	データブロックの数。nsbmd ファイルの場合は 1 か 2。その他のファイルの場合は 1 である。
offset	ファイル先頭から各ブロックへのオフセットが格納されている。

### 3.1.2 データブロックヘッダ

バイナリファイル内の各データブロックの先頭には、データブロックの種別とサイズを表す以下のようなデータが格納されています。

```
pseudo_struct DataBlockHeader(NNSG3dResDataBlockHeader) {  
    u32 kind;  
    u32 size;  
};
```

表 3-2 DataBlockHeader のデータメンバの解説

名称	内容
kind	データブロックの種別を表すシンボルが格納される
size	データブロック全体のサイズ

### 3.1.3 ディクショナリ

G3D ではテクスチャやマテリアルといった各種リソースに 16 文字までの名前をつけて、名前でアクセスすることが可能になっています。名前検索は同じデータ構造を用いて行われます。この節ではディクショナリのデータ構造について解説します。

ディクショナリを擬似構造体で表すと以下のようになります。

```
pseudo_struct Dictionary(NNSG3dResDict) {  
    u8 revision = 0;  
    u8 numEntry;  
    u16 sizeDictBlk;  
    PADDING(2 bytes);  
    u16 ofsEntry;  
  
    pseudo_struct PtreeNode(NNSG3dResDictTreeNode) {  
        u8 refBit;  
        u8 idxLeft;  
        u8 idxRight;  
        u8 idxEntry;  
    } node[numEntry + 1];  
  
    pseudo_struct DictEntry(NNSG3dResDictEntryHeader) {  
        u16 sizeUnit;  
        u16 ofsName;  
        u8 data[numEntry][sizeUnit];  
    } entry;  
  
    pseudo_struct DictName(NNSG3dResName) {  
        u8 name[16];  
    } names[numEntry];  
};
```

表 3-3 Dictionary のデータメンバの解説

名称	内容	
revision	ディクショナリ構造のバージョン(0 のみ)	
numOfEntry	ディクショナリに登録されているエントリの数	
sizeDictBlk	ディクショナリのサイズ(バイト単位)	
ofsEntry	Dictionary の先頭から DictEntry へのオフセット	
PtreeNode	refBit	入力文字列の先頭から rebBit 目のビットが参照される
	idxLeft	参照されたビットが 0 のとき次に参照するノードのインデックス
	idxRight	参照されたビットが 1 のとき次に参照するノードのインデックス
	idxEntry	ノードに対応する DictEntry と DictName のインデックス
DictEntry	sizeUnit	データエントリ1つあたりのサイズ。4 バイト以上 4 バイト単位
	ofsName	DictEntry の先頭から DictName へのオフセット
	data	データ格納部
DictName	name	リソースの名前文字列。名前は 16 文字以内で使用されていない領域には 0 が入っていないてはならない。16 文字を全て使い切った場合には C 文字列として扱うことはできない。

データ格納部(DictEntry::data)には通常リソースへのオフセットが格納されていますが、データが直に入っていることもあります。オフセットが格納されている場合は、ディクショナリを利用した関数が正しい参照先のポインタを求める必要があります。

名前による検索は、DictName を線形検索することによっても可能ですが、パトリシアというアルゴリズムを使用するための木(node[])が用意されています。パトリシアの利用によりエントリ数が増加した場合の検索時間の増大を抑えることが可能になっています。このアルゴリズムの詳細については、R.セジウィック「アルゴリズム C++」等を参照して下さい。なお、インデックスによる参照も DictEntry を走査することにより可能です。

パトリシア木は基数探索木の一種で、木の各ノードには「探しているキーのどのビットを調べるか」「調べたビットが ON の時進むノード(右)と OFF の時進むノード(左)へのポインタ」「ノードの持っているキー」という情報が格納されており、これらの情報を使って探索を行います。

パトリシア木を探索するには、まずノードに書かれている「どのビットを調べるか」の情報を、与えられたキー(この場合具体的にはリソース名そのもの)に当てはめて調べ、左右どちらのノードへ進むかを決定します。進んだノードが通常の子ノードならまた同じ事を繰り返し、進んだノードがさっきそこから来たノードよりも「上流の(要するにリンク関係上

より根に近いところにある)」ノードだったら、そこでノードの移動を打ち切ってノードの持っているキーとのの比較に移ります。

比較の結果同一のキーなら発見、そうでなければ失敗、として処理を終えます。

このように、このアルゴリズムはキー全体の比較を行うのは最後の一度だけで、各ノードでの比較は 1bit だけで済むため、高速な検索が可能です。



### 3.1.4 アニメーションヘッダ

各アニメーションリソースは、アニメーションの種別を分類するためのヘッダを保持しています。以下のようなデータ構造になります。

```
pseudo_struct AnmHeader(NNSG3dResAnmHeader) {  
    u8 category0;  
    u8 revision;  
    u16 category1;  
};
```

表 3-4 AnmHeader のデータメンバの解説

名称	内容
category0	アニメーションのカテゴリを指定します。 ‘M’ ならばマテリアルアニメーション ‘J’ ならばジョイントアニメーション ‘V’ ならばビジビリティアニメーション
revision	アニメーションファイルフォーマットのリビジョン。
category1	アニメーションの種類を指定します。 ‘CA’ ならばジョイントアニメーション ‘VA’ ならばビジビリティアニメーション ‘MA’ ならばマテリアルカラーアニメーション ‘TP’ ならばテクスチャパターンアニメーション ‘TA’ ならばテクスチャSRTアニメーション

## 3.2 モデルデータファイル(.nsbmd)の構造

.nsbmd ファイルは大きくモデル部とテクスチャ・パレット部に分けることができます。モデルブロックには、モデルの集合が格納されていて、モデル毎にジョイント構造・マテリアル・シェイプが格納されています。一方、テクスチャ・パレットブロックにはテクスチャとパレットの集合が格納されていて、実行時に VRAM にロード・各モデルに関連付けすることができます。各モデルとテクスチャやパレットは 16 文字以内の名前によって関連付けられます。

以下に.nsbmd ファイルを擬似構造体を用いて示します。

```
pseudo_struct NSBMD {
    FileHeader fileHeader = {
        dataBlocks = 1 or 2,
        signature = '0DMB'
    };
    ModelSet modelSet;
    IF (there are textures) {
        TexPlttSet texPlttSet;
    }
};
```

表 3-5 NSBMD のデータメンバの解説

名称	内容
fileHeader	ファイルのヘッダ領域
modelSet	モデルブロック
texPlttSet	テクスチャ・パレットブロック

### 3.2.1 モデルブロック

モデルブロックには複数のモデルを格納することができ、各モデルに対しては 16 文字以内の名前によってアクセスすることができます。

#### 3.2.1.1 モデルの集合

モデルの集合を擬似構造体で示すと以下のようになります。

```
pseudo_struct ModelSet(NNSG3dResMdlSet) {
    DataBlockHeader header = {
        kind = '0LDM',
        size = SIZEOF(ModelSet)
    };
    Dictionary dict = {sizeUnit = 4 bytes};
    Model models[# of models];
};
```

ディクショナリのデータ部は 32bit で ModelSet の先頭部からのオフセット(バイト単位)が格納されています。

表 3-6 ModelSet のデータメンバの解説

名称	内容
header	モデルブロックのヘッダ領域
dict	各モデルへアクセスするための辞書領域
models	モデルブロック内のモデルの集合

### 3.2.1.2 モデル

1つのモデルを擬似構造体で示すと以下のようになります。

```
pseudo_struct Model(NNSG3dResMdl) {  
    u32 size = SIZE_OF(Model);  
    u32 ofsSbc;  
    u32 ofsMat;  
    u32 ofsShp;  
    u32 ofsEvpMtx;  
    ModelInfo info;  
    NodeSet nodes;  
    u8 sbc[ofsMat - ofsSbc];  
    MaterialSet materials;  
    ShapeSet shapes;  
    EvpMatrices evpMatrices;  
};
```

ここで、各オフセットは **Model** の先頭部からのバイト数として格納されています。1つのモデルは、モデルに関する基本的な情報・各ノードの情報・各マテリアルの情報・各シェイプの情報・ノード/マテリアル/シェイプの関連付け情報・エンベロープ計算用の行列格納領域に分けられます。

**表 3-7 Model のデータメンバの解説**

名称	内容
size	モデルのサイズ
ofsSbc	Model の先頭から SBC 列へのオフセット
ofsMat	Model の先頭からマテリアルの集合へのオフセット
ofsShape	Model の先頭からシェイプの集合へのオフセット
ofsEvpMtx	Model の先頭からエンベロープ行列計算用の行列格納領域へのオフセット
info	モデルに関する基本的な情報
nodes	各ノード位置・姿勢等の情報
sbc	ノード/マテリアル/シェイプの関連付け情報
materials	各マテリアルの情報
shapes	各シェイプの情報
evpMatrices	エンベロープ計算用の行列格納領域

### 3.2.1.3 モデルに関する基本的な情報

モデルに関する基本的な情報を擬似構造体で示すと以下のようになります。

```

pseudo_struct ModelInfo(NNSG3dResMdlInfo) {
    u8 sbcType;
    u8 scalingRule;
    u8 texMtxMode;
    u8 numNode;
    u8 numMat;
    u8 numShp;
    u8 firstUnusedMtxStackID;
    PADDING(1 byte);

    fx32 posScale;
    fx32 invPosScale;
    u16 numVertex;
    u16 numPolygon;
    u16 numTriangle;
    u16 numQuad;

    fx16 boxX, boxY, boxZ;
    fx16 boxW, boxH, boxD;
    fx32 boxPosScale;
    fx32 boxInvPosScale;
};

```

表 3-8 ModellInfo のデータメンバの解説

名称	内容
sbcType	SBC のタイプ。0 が入る。
scalingRule	スケーリングの計算方法
texMtxMode	テクスチャ行列の計算方法
numNode	ジョイント数
numMat	マテリアル数
numShp	シェイプ数
firstUnusedMtxStackID	行列スタック中で空いている領域の先頭(スタックのインデックス)
posScale, invPosScale	頂点位置座標にかけるスケール値とその逆数
vertexSize	頂点数
polygonSize	ポリゴン数
triangleSize	polygonSize にカウントされるポリゴンのうち、三角形ポリゴンの数
quadSize	polygonSize にカウントされるポリゴンのうち、四角形ポリゴンの数
boxX, boxY, boxZ	ボックステスト用 (G3_BoxTest に渡すべきパラメータ)
boxW, boxH, boxD	同上
boxPosScale, boxInvPosScale	ボックステスト前にかけるスケール値とその逆数

表 3-9 scalingRule がとる値

値	内容
0	通常のモデルである場合
1	Maya の segment scale compensate が適用されているジョイントがあるモデルの場合
2	Softimage 3D, Softimage XSI の classic scale off が指定されているモデルの場合

表 3-10 texMtxMode がとる値

値	内容
0	Maya に対応したテクスチャ行列の計算を用いる
1	Softimage 3D に対応したテクスチャ行列の計算を用いる
2	3dsmax に対応したテクスチャ行列の計算を用いる
3	Softimage XSI に対応したテクスチャ行列の計算を用いる

### 3.2.1.4 ノードの情報

各ノード位置・姿勢等の情報の集合を擬似構造体で示すと以下のようになります。

```
pseudo_struct NodeSet(NNSG3dResNodeInfo) {
    Dictionary dict = {sizeUnit = 4 bytes};
    pseudo_struct NodeData(NNSG3dResNodeData) {
        u16 flag;
        u16 _00;
        IF (!(flag & NNS_G3D_SRT_FLAG_TRANS_ZERO)) {
            fx32 Tx, Ty, Tz;
        }
        IF (!(flag & NNS_G3D_SRT_FLAG_ROT_ZERO) &&
            !(flag & NNS_G3D_SRT_FLAG_PIVOT_EXIST)) {
            fx16 _01, _02;
            fx16 _10, _11, _12;
            fx16 _20, _21, _22;
        }
        IF (!(flag & NNS_G3D_SRT_FLAG_ROT_ZERO) &&
            (flag & NNS_G3D_SRT_FLAG_PIVOT_EXIST)) {
            fx16 A, B;
        }
        IF (!(flag & NNS_G3D_SCALE_ONE)) {
            fx32 Sx, Sy, Sz;
            fx32 InvSx, InvSy, InvSz;
        }
    } data[# of nodes];
};
```

ディクショナリのデータ部は 32bit で NodeInfo の先頭部からのオフセット(バイト単位)が格納されています。NodeData のサイズは flag の値によって異なります。単位行列やゼロベクトルといった場合には、データが省略されるようになっています。

表 3-11 NodeData の flag フィールドの値

定義名	値	説明
NNS_G3D_SRTFLAG_TRANS_ZERO	0x0001	このビットが ON なら平行移動成分は 0
NNS_G3D_SRTFLAG_ROT_ZERO	0x0002	このビットが ON なら回転行列は単位行列
NNS_G3D_SRTFLAG_SCALE_ONE	0x0004	このビットが ON ならスケールは 1
NNS_G3D_SRTFLAG_PIVOT_EXIST	0x0008	このビットが ON なら回転行列は圧縮形
NNS_G3D_SRTFLAG_PIVOT_MASK	0x00f0	回転行列が圧縮形である場合、ピボット要素(絶対値が 1 である要素)の位置(0-8)を示す。
NNS_G3D_SRTFLAG_PIVOT_MINUS	0x0100	このビットが ON ならピボット要素は負(つまり-1)
NNS_G3D_SRTFLAG_SIGN_REVC	0x0200	このビットが ON なら C は B の反対の符号を持つ
NNS_G3D_SRTFLAG_SIGN_REVD	0x0400	このビットが ON なら D は A の反対の符号を持つ
NNS_G3D_SRTFLAG_IDXPIVOT_MASK	0x00f0	この値と info の論理積でピボット要素の位置を指定

表 3-12 NodeData のデータメンバの解説

名称	内容
flag	フラグ。表 3-11 NodeDataのflagフィールドの値 を参照のこと。
Tx, Ty, Tz	ノードに設定されている平行移動成分。全て 0 の場合は省略されます。
_00, _01, _02, _10, _11, _12 _20, _21, _22	ノードに設定されている回転行列。単位行列である場合は省略されています。また、いずれかの要素が 1 か-1 である場合には、下段のデータが回転行列として使用されます。
A, B	ノードに設定されている回転行列が単位行列でなくて、いずれかの要素が 1 か-1 である場合用いられる回転行列のデータ形式です。詳しくは欄外に説明されています。
Sx, Sy, Sz, InvSx, InvSy, InvSz	ノードに設定されているスケール値とその逆数。全て 1 の場合は省略されます。

表中 ABCD とあるのは、行列中で、ピボット要素を含む行・列を消去した結果の4つの要素(ピボット要素に関する小行列中の要素)を指しています。具体的に元の行列上でのピボット要素と ABCD の位置関係を書くと次のようになります。

$$\text{ピボットが 4} \rightarrow \begin{pmatrix} A & 0 & B \\ 0 & 1 & 0 \\ C & 0 & D \end{pmatrix}, \text{ピボットが 0} \rightarrow \begin{pmatrix} 1 & 0 & 0 \\ 0 & A & B \\ 0 & C & D \end{pmatrix}, \text{ピボットが 8} \rightarrow \begin{pmatrix} A & B & 0 \\ C & D & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

ここで、回転行列は直交行列なので、ピボット要素を含む行・列の要素でピボット要素以外の要素は 0 であり、C は+B か・B, D は+A か・A になります。

### 3.2.1.5 マテリアルの情報

マテリアルの集合をまとめる部分を擬似構造体で示すと以下のようになります。

```
pseudo_struct MaterialSet(NNSG3dResMat) {
    u16 ofsDictTexToMatList;
    u16 ofsDictPlttToMatList;
    Dictionary dict = {sizeUnit = 4 bytes};
    Dictionary dictTexToMatList = {sizeUnit = 4 bytes};
    Dictionary dictPlttToMatList = {sizeUnit = 4 bytes};
    u8 matIdxData[];
    PADDING(4 bytes alignment);
    Material materials[# of materials];
};
```

表 3-13 MaterialSet のデータメンバ

名称	内容
ofsDictTexToMatList	MaterialSet の先頭から dictTexToMatList へのオフセット
ofsDictPlttToMatList	MaterialSet の先頭から dictPlttToMatList へのオフセット
dict	マテリアル名やマテリアル ID から各マテリアルを参照する辞書
dictTexToMatList	テクスチャ名からマテリアル ID のリストを参照する辞書
dictPlttToMatList	パレット名からマテリアル ID のリストを参照する辞書
matIdxData	マテリアル ID の並び。dictTexToMatList と dictPlttToMatList からアクセスされる。
materials	各マテリアルの並び

**MaterialSet** はディクショナリを 3 種類もっています。ディクショナリ **dict** はマテリアル名から各マテリアルを参照するための辞書で、ディクショナリ **dictTexToMatList** とディクショナリ **dictPlttToMatList** はテクスチャ名やパレット名からそのテクスチャやパレットを利用しているマテリアル ID のリストを得るための辞書です。この 2 つの辞書はテクスチャやパレットとモデルを関連付ける際に利用することができます。

ディクショナリ **dict** のデータ部は、32bit で **MaterialSet** の先頭部からのオフセットが格納されています。また、ディクショナリ **dictTexToMatList**, **dictPlttToMatList** のデータ部は 32bit で、下位 16bit に **MaterialSet** の先頭部からのオフセットが格納されていて、**matIdxData** 内に格納されているマテリアル ID の並びの先頭を参照しています。16-23 ビットには、マテリアル ID の数が格納されています。24-31 ビットは、テクスチャがバインドされている場合は 1 で、そうでない場合は 0 が入ることになっています。

各マテリアルのデータ構造を擬似構造体で表すと以下ようになります。

```
pseudo_struct Material(NNSG3dResMatData) {
    u16 itemTag = 0;
    u16 size;
    u32 diffAmb, specEmi;
    u32 polyAttr, polyAttrMask;
    u32 texImageParam, texImageParamMask;
    u16 texPlttBase;
    u16 flag;
    u16 origWidth, origHeight;
    fx32 magW, magH;
    IF (!(flag & NNS_G3D_MATFLAG_TEXMTX_SCALEONE)) {
        fx32 scaleS, scaleT;
    }
    IF (!(flag & NNS_G3D_MATFLAG_TEXMTX_ROTZERO)) {
        fx32 rotSin, rotCos;
    }
    IF (!(flag & NNS_G3D_MATFLAG_TEXMTX_TRANSZERO)) {
        fx32 transS, transT;
    }
    IF (flag & NNS_G3D_MATFLAG_EFFECTMTX) {
        fx32 effectMtx[16];
    }
};
```

表 3-14 Material のデータメンバ

名称	内容
itemTag	マテリアルデータの種別を表す(現在のところ 0 のみ)。
size	当該マテリアルのサイズ。
diffAmb	ディフューズとアンビエントの指定。ジオメトリコマンドの MaterialColor0 コマンドのパラメータと同じビットパターンである。
specEmi	スペキュラとエミッションの指定。ジオメトリコマンドの MaterialColor1 コマンドのパラメータと同じビットパターンである。
polygonAttr	ポリゴン属性値の指定。ジオメトリコマンドの PolygonAttr コマンドのパラメータと同じビットパターンである。
polygonAttrMask	polygonAttr のうちデータとして有効なビットを 1 としたマスク。無効とされたビットに関しては、デフォルトの設定と合成される等して設定されることになる。
texImageParam	テクスチャイメージパラメータの設定。ジオメトリコマンドの TexImageParam コマンドのパラメータと同じビットパターンである。テクスチャの VRAM 先頭アドレス、テクスチャサイズ、テクスチャフォーマット、パレットのカラー0設定値イネーブルフラグは設定されておらず、バインド時にバインドされるテクスチャの設定を利用します。
texImageParamMask	TexImageParam のうちデータとして有効なビットを 1 としたマスク。無効とされたビットに関しては、デフォルトの設定と合成される等して設定されることになる。
texPlttBase	テクスチャパレットのベースアドレスの設定。ジオメトリコマンドの TexPlttBase コマンドのパラメータの下位 16 ビットと同じビットパターンである。
flag	テクスチャに関する各種フラグ(後述)。
origWidth, origHeight	ツール作成時にマテリアルに割り当てられていたテクスチャの幅と高さ。
magW, magH	実行時にバインドされたテクスチャの幅と高さを origWidth, origHeight で割ったものを格納する領域。
scaleS, scaleT	テクスチャのスケール成分。
rotSin, rotCos	テクスチャの回転角の正弦と余弦。
transS, transT	テクスチャの平行移動成分。



表 3-15 Material の flag メンバの値

名称	値	説明
NNS_G3D_MATFLAG_TEXMTX_USE	0x0001	テクスチャ行列を使用するかどうか
NNS_G3D_MATFLAG_TEXMTX_SCALEONE	0x0002	テクスチャのスケール成分が全て 1.0 なら ON
NNS_G3D_MATFLAG_TEXMTX_ROTZERO	0x0004	テクスチャが回転しないなら ON
NNS_G3D_MATFLAG_TEXMTX_TRANSZERO	0x0008	テクスチャが平行移動しないなら ON
NNS_G3D_MATFLAG_ORIGWH_SAME	0x0010	テクスチャの Width/Height がシステムと同じ場合 セットされる(このビットは実行の際、初期化時にセ ットされる)
NNS_G3D_MATFLAG_WIREFRAME	0x0020	ワイヤーフレーム表示なら ON
NNS_G3D_MATFLAG_DIFFUSE	0x0040	マテリアルで diffuse を指定するなら ON
NNS_G3D_MATFLAG_AMBIENT	0x0080	マテリアルで ambient を指定するなら ON
NNS_G3D_MATFLAG_VTXCOLOR	0x0100	マテリアルで vtxcolor フラグを指定するなら ON
NNS_G3D_MATFLAG_SPECULAR	0x0200	マテリアルで specular を指定するなら ON
NNS_G3D_MATFLAG_EMISSION	0x0400	マテリアルで emission を指定するなら ON
NNS_G3D_MATFLAG_SHININESS	0x0800	マテリアルで shininess フラグを指定するなら ON
NNS_G3D_MATFLAG_TEXPLTTBASE	0x1000	テクスチャパレットベースアドレスを指定するなら ON
NNS_G3D_MATFLAG_EFFECTMTX	0x2000	環境マップ・投影マップで使用するエフェクト行列 が存在する場合は ON

Material はマテリアルカラー、ポリゴンアトリビュート、テクスチャ関連のパラメータを保持しています。  
TexImageParam, texPlttBase, magW, magH は、テクスチャのバインド時にバインドされるテクスチャの設定を反映  
する必要があります。

### 3.2.1.6 シェイプの集合とシェイプ

シェイプの集合とシェイプを擬似構造体で示すと以下のようになります。

```

pseudo_struct ShapeSet(NNSG3dResShp) {
    Dictionary dict = {sizeUnit = 4 bytes};

    pseudo_struct Shape(NNSG3dResShpData) {
        u16 itemTag = 0;
        u16 size;
        u32 flag;
        u32 ofsDL;
        u32 sizeDL;
    } shape[# of shapes];

    u32 DL[SUM(Shape::sizeDL)];
};

```

表 3-16 ShapeSet のデータメンバ

名称	内容	
dict	各シェイプへアクセスするための辞書領域	
shape	itemTag	シェイプデータの種類を表す(現在のところ 0 のみ)
	size	当該シェイプのサイズ(Shape のサイズ)
	flag	ディスプレイリストの特徴を表すフラグ。 表 3-17を参照のこと。
	ofsDL	Shape の先頭からディスプレイリストへのオフセット
	sizeDL	ディスプレイリストのサイズ
DL	ShapeSet に属するシェイプのディスプレイリストを格納する配列	

表 3-17 ShapeSet::Shape::flag メンバの値

名称	値	内容
NNS_G3D_SHPFLAG_USE_NORMAL	0x00000001	ON ならディスプレイリスト内に Normal コマンドが存在
NNS_G3D_SHPFLAG_USE_COLOR	0x00000002	ON ならディスプレイリスト内に Color コマンドが存在
NNS_G3D_SHPFLAG_USE_TEXCOORD	0x00000004	ON ならディスプレイリスト内に TexCoord コマンドが存在
NNS_G3D_SHPFLAG_USE_RESTOREMTX	0x00000008	ON ならディスプレイリスト内に RestoreMtx コマンドが存在

g3devtr で.imd ファイルを変換する場合、ディスプレイリスト内の最初の RestoreMtx コマンドは後述の SBC 内にエンコードされます。従って、エンベロープを用いない場合は、ディスプレイリスト内に RestoreMtx コマンドは存在しません。

### 3.2.1.7 ノード・マテリアル・シェイプの関連付け情報

ノード・マテリアル・シェイプを互いに関連付ける情報は、可変長のバイトコードとしてエンコードされています。このバイトコードを SBC(Structure Byte Code)と呼ぶことにします。

SBC はノード間の親子関係・ノードと行列スタックのインデックスとの関連付け・マテリアルとシェイプの組み合わせの指定と、ノードへの関連付け・ビルボード変換・モデリング行列のブレンドといった情報を格納しています。それぞれの情報は別々のコマンドとして定義されていて、順番に処理すればモデルが描画できるように配置されています。

表 3-18 SBC コマンド一覧

コマンド名(シンボル)	NOP(NNS_G3D_SBC_NOP)
エンコーディング	<div>7<span style="float: right;">0</span></div> <div>0 0 0 0 0 0 0 0</div>
オペランド	無し
処理内容	何もしない。

コマンド名 (シンボル)	RET(NNS_G3D_SBC_RET)
エンコーディング	<div>7 0</div> <div>0 0 0 0 0 0 0 1</div>
オペランド	無し
処理内容	SBC 列の最後に存在します。

コマンド名 (シンボル)	NODE(NNS_G3D_SBC_NODE)
エンコーディング	<div>7 0</div> <div>0 0 0 0 0 0 1 0</div> <div>NodeID</div> <div>--- --- --- --- --- --- --- V</div>
オペランド	NodeID: ノード ID に対応するノードを指定する V: NodeID に属するシェイプが可視である場合は 1、可視でない場合は 0
処理内容	次の NODE コマンドが出現するまでの全ての MAT コマンドと SHP コマンドは、NODE コマンドで指定された NodeID を持つノードに所属するものとみなされる。

コマンド名 (シンボル)	MTX(NNS_G3D_SBC_MTX)
エンコーディング	<div>7 0</div> <div>0 0 0 0 0 0 1 1</div> <div>0 0 0 Idx</div>
オペランド	Idx: 行列スタックのインデックス
処理内容	RestoreMtx コマンドを発行して、位置座標行列の行列スタックの指定位置からカレント行列に行列を読み出します。

コマンド名 (シンボル)	MAT(NNS_G3D_SBC_MAT)
エンコーディング	<div>7 0</div> <div>OPT 0 0 1 0 0</div> <div>MatID</div>
オペランド	MatID: マテリアルのID
処理内容	指定したマテリアルの設定をジオメトリエンジンに設定します。OPT の値は動作の高速化のためのヒントとして使用されます。  OPT=000 の場合: オペランドで指定された MatID はこの SBC 内で唯一のものである場合。 OPT=001 の場合: オペランドで指定された MatID は以降の MAT コマンドで指定される可能性がある。 OPT=010 の場合: オペランドで指定された MatID は以前の MAT コマンドで指定されたことがあるが、今後指定されることはない。

コマンド名 (シンボル)	SHP(NNS_G3D_SBC_SHP)
エンコーディング	<div>70</div> <div><div>00000101</div></div> <div>ShpID</div>
オペランド	ShpID: シェイプの ID
処理内容	指定されたシェイプを描画します。

コマンド名 (シンボル)	NODEDESC(NNS_G3D_SBC_NODEDESC)																						
エンコーディング	<div><div>70</div><table><tr><td>OPT</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr></table><div>NodeID</div><div>ParentNodeID</div><table><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>P</td><td>S</td></tr></table><div>OPT=001,011 の場合に存在</div><table><tr><td>0</td><td>0</td><td>0</td><td>DestIdx</td></tr></table><div>OPT=010,011 の場合に存在</div><table><tr><td>0</td><td>0</td><td>0</td><td>SrcIdx</td></tr></table></div>	OPT	0	0	1	1	0	0	0	0	0	0	0	P	S	0	0	0	DestIdx	0	0	0	SrcIdx
OPT	0	0	1	1	0																		
0	0	0	0	0	0	P	S																
0	0	0	DestIdx																				
0	0	0	SrcIdx																				
オペランド	<div>NodeID: モデリング行列を求めるノード ID を指定します。</div> <div>ParentNodeID: 親ノードの ID を指定します。</div> <div>S: このノードに Maya の Segment Scale Compensate が掛かります。</div> <div>P: このノードは Maya の Segment Scale Compensate が掛かるノードの親ノードです。</div> <div>DestIdx: 計算結果を行列スタックにストアする場合に行列スタックのインデックスが指定されます。 計算結果を行列スタックにストアする必要がある場合に指定されます。</div> <div>SrcIdx: 計算前に行列スタックから行列をリストアする場合に行列スタックのインデックスが指定されます。親ノードに対応する行列を行列スタックから取り出す場合に指定されます。</div>																						
処理内容	ノード ID に対応するモデリング行列を計算します。																						

コマンド名 (シンボル)	BB(NNS_G3D_SBC_BB)
エンコーディング	<div><div>70</div><div><div>00000111</div></div><div><div>NodeID</div></div><div>OPT=001,011 の場合に存在</div><div><div>000DestIdx</div></div><div>OPT=010,011 の場合に存在</div><div><div>000SrcIdx</div></div></div>
オペランド	<div><div>NodeID:</div><div>ビルボード変換を適用する行列のノードIDです。</div><div>DestIdx:</div><div>計算結果を行列スタックにストアする場合に行列スタックのインデックスが指定されます。</div><div>SrcIdx:</div><div>計算前に行列スタックから行列をリストアする場合に行列スタックのインデックスが指定されます。</div></div>
処理内容	行列にビルボード変換を適用します。

コマンド名 (シンボル)	BBY(NNS_G3D_SBC_BBY)
エンコーディング	<div><div>70</div><div><div>00001000</div></div><div><div>NodeID</div></div><div>OPT=001,011 の場合に存在</div><div><div>000DestIdx</div></div><div>OPT=010,011 の場合に存在</div><div><div>000SrcIdx</div></div></div>
オペランド	<div>NodeID: Y 軸ビルボード変換を適用する行列のノードIDです。</div> <div>DestIdx: 計算結果を行列スタックにストアする場合に行列スタックのインデックスが指定されます。</div> <div>SrcIdx: 計算前に行列スタックから行列をリストアする場合に行列スタックのインデックスが指定されます。</div>
処理内容	行列に Y 軸ビルボード変換を適用します。

コマンド名 (シンボル)	NODEMIX(NNS_G3D_SBC_NODEMIX)
エンコーディング	<div><div>70</div><div><div>00001001</div></div><div>DestIdx</div><div>NumMtx</div><div>以下の値を NumMtx 個繰り返す</div><div>SrcIdx_N</div><div>NodeID_N</div><div>Ratio_N</div><div>命令長は 3+3*NumMtx になる。</div></div>
オペランド	<div>DestIdx: 計算結果の行列が格納される行列スタックのインデックス</div> <div>NumMtx: ブレンドされる行列の数</div> <div>SrcIdx_N, NodeID_N, Ratio_N は順番に NumMtx 回繰り返される。</div> <div>SrcIdx_N: ブレンドされる行列が格納されている行列スタックのインデックス</div> <div>NodeID_N: ブレンドされる行列のノード ID</div> <div>Ratio_N: 行列のブレンド比率、符号なしの小数部 8 ビットの固定小数になっている。</div>
処理内容	位置座標行列を指定された比率でブレンドして、ウェイトドエンベロープ用の行列を計算します。計算の際には、evpMatrices 内に格納されているモデリング変換行列の逆行列(モデル全体の座標系から各ジョイントの座標系に変換するため)を使用します。

コマンド名 (シンボル)	CALLDL(NNS_G3D_SBC_CALLDL)
エンコーディング	<div><div>70</div><div><div>00001010</div></div><div>310</div><div><div>RelAddr</div></div><div>310</div><div><div>Size</div></div></div>
オペランド	<div>RelAddr: CALLDL 命令の先頭アドレスからディスプレイリストへの相対アドレス</div> <div>Size: ディスプレイリストのサイズ(バイト単位)</div>
処理内容	オペランドで指定されたディスプレイリストをジオメトリエンジンに送信します。

コマンド名 (シンボル)	POSSCALE(NNS_G3D_SBC_POSSCALE)
エンコーディング	<div>7 0</div> <div>OPT 0 1 0 1 1</div>
オペランド	無し
処理内容	OPT=000 の場合は、カレント行列にモデルデータ毎に設定されているスケーリング行列 (ModelInfoの posScale, invPosScaleを参照)をかけます。 OPT=001 の場合は、その逆行列をかけます。

コマンド名 (シンボル)	ENVMAP(NNS_G3D_SBC_ENVMAP)
エンコーディング	<div>7 0</div> <div>OPT=0 0 1 1 0 0</div> <div>MatID</div> <div>Flag</div>
オペランド	MatID: マテリアルのID Flag: 拡張用フラグ(現在のところ常に 0)
処理内容	環境マップ用のテクスチャ行列を計算します。MAT コマンドの直後に配置され、OPT の値は常に0です。

コマンド名 (シンボル)	PRJMAP(NNS_G3D_SBC_PRJMAP)
エンコーディング	<div>7 0</div> <div>OPT=0 0 1 1 0 1</div> <div>MatID</div> <div>Flag</div>
オペランド	MatID: マテリアルのID Flag: 拡張用フラグ(現在のところ常に 0)
処理内容	投影マップ用のテクスチャ行列を計算します。MAT コマンドの直後に配置され、OPT の値は常に 0 です。

### 3.2.1.8 エンベロープ計算用の行列格納領域

ウェイト付エンベロープ付のモデルの場合のみ、この領域は存在します。静止ポーズにおける、個々のジョイント座標系からオブジェクト座標系に変換する位置座標行列・方向ベクトル行列の逆行列(つまり、オブジェクト座標系から個々のジョイント座標系への変換行列)が格納されています。SBCの NODEMIXコマンドの処理において、逆行列の計算を省略するために用いることができます。

```
pseudo_struct EvpMatrices {  
    IF (Weighted envelope model) {  
        pseudo_struct {  
            MtxFx43 invM;  
            MtxFx33 invN;  
        } m[# of nodes];  
    }  
};
```

**表 3-19 EvpMatrices のデータメンバ**

名称	内容
invM	位置座標行列の逆行列
invN	方向ベクトル行列の逆行列

## 3.2.2 テクスチャ・パレットブロック

テクスチャ・パレットブロックは、複数のテクスチャと複数のパレットを格納することができ、各テクスチャ・パレットに対しては、16 文字以内の名前によってアクセスすることができます。

### 3.2.2.1 テクスチャとパレットの集合

テクスチャとパレットの集合を擬似構造体で示すと以下のようになります。



```

pseudo_struct TexPlttSet(NNSG3dResTex) {
    DataBlockHeader header = {
        kind = '0XET',
        size = SIZE_OF(TexPlttSet)
    };
    pseudo_struct TexInfo(NNSG3dResTexInfo) {
        u32 vramKey;
        u16 sizeTex;
        u16 ofsDict;
        u16 flag;
        PADDING(2 bytes);
        u32 ofsTex;
    } texInfo;

    pseudo_struct Tex4x4Info(NNSG3dResTex4x4Info) {
        u32 vramKey;
        u16 sizeTex;
        u16 ofsDict;
        u16 flag;
        PADDING(2 bytes);
        u32 ofsTex;
        u32 ofsTexPlttIdx;
    } tex4x4Info;

    pseudo_struct PlttInfo(NNSG3dResPlttInfo) {
        u32 vramKey;
        u16 sizePltt;
        u16 flag;
        u16 ofsDict;
        PADDING(2 bytes);
        u32 ofsPlttData;
    } plttInfo;

    Dictionary dictTex = {sizeUnit = 8 bytes};
    Dictionary dictPltt = {sizeUnit = 4 bytes};
    u8 texData[texInfo.sizeTex << 3];
    u8 tex4x4Data[tex4x4Info.sizeTex << 3];
    u8 tex4x4IdxData[tex4x4Info.sizeTex << 2];
    u8 plttData[plttInfo.sizePltt << 3];
};

```

表 3-20 TexPlttSet のデータメンバの解説

名称	内容	
header	テクスチャ・パレットブロックのヘッダ領域	
texInfo	vramKey	4x4 以外のテクスチャデータに対する VRAM キーの格納場所
	sizeTex	texData のサイズを右に 3bit シフトした数値
	ofsDict	TexPlttSet の先頭から dictTex へのオフセット
	flag	4x4 以外のテクスチャデータに関するフラグ

	ofsTex	TexPlttSet の先頭から texData へのオフセット
tex4x4Info	vramKey	4x4 テクセル圧縮のテクスチャデータに対する VRAM キーの格納場所
	sizeTex	tex4x4Data のサイズを右に 3bit シフトした数値
	ofsDict	TexPlttSet の先頭から dictTex へのオフセット
	flag	4x4 テクセル圧縮テクスチャデータに関するフラグ
	ofsTex	TexPlttSet の先頭から tex4x4Data へのオフセット
	ofsTexPlttIdx	TexPlttSet の先頭から tex4x4IdxData へのオフセット
plttInfo	vramKey	パレットに対する VRAM キーの格納場所
	sizePltt	plttData のサイズを右に 3bit シフトした数値
	flag	パレットデータに関するフラグ
	ofsDict	TexPlttSet の先頭から dictPltt へのオフセット
	ofsPlttData	TexPlttSet の先頭から plttData へのオフセット
dictTex		テクスチャ名から各テクスチャの属性値にアクセスする辞書
dictPltt		パレット名から各パレットの属性値にアクセスする辞書
texData		4x4 テクセル圧縮テクスチャ以外のテクスチャデータがまとめられた配列
tex4x4Data		4x4 テクセル圧縮テクスチャのテクスチャデータがまとめられた配列
tex4x4IdxData		4x4 テクセル圧縮テクスチャ用のパレットインデックスデータがまとめられた配列
plttData		パレットデータがまとめられた配列

表 3-21 TexPlttSet::TexInfo::flag メンバの値

名称	値	説明
NNS_G3D_RESTEX_LOADED	0x0001	4x4 以外のテクスチャデータが VRAM にロードされている場合にセットされます。

表 3-22 TexPlttSet::Tex4x4Info::flag メンバの値

名称	値	説明
NNS_G3D_RESTEX4x4_LOADED	0x0001	4x4 テクセル圧縮テクスチャデータが VRAM にロードされている場合にセットされます。

表 3-23 TexPlttSet::PlttInfo::flag メンバの値

名称	値	説明
NNS_G3D_RESPLTT_LOADED	0x0001	パレットデータが VRAM にロードされている場合にセットされます。
NNS_G3D_RESPLTT_USEPLTT4	0x8000	4 色パレットが存在する場合にセットされています。

辞書 dictTex 内のデータはデータへのオフセットではなく、以下の擬似構造体で示されるデータが格納されています。

```
pseudo_struct DictTexData(NNSG3dResDictTexData) {
    u32 texImageParam;
    u32 extraParam;
};
```

表 3-24 ディクショナリ dictTex 内に格納されているデータ

名称	内容							
texImageParam	31 30 29 28 26 25 23 22 20 19 16 15 0							
	<table><tr><td></td><td>P</td><td>Fmt</td><td>T</td><td>S</td><td></td><td>OFS</td></tr></table>		P	Fmt	T	S		OFS
		P	Fmt	T	S		OFS	
	ジオメトリコマンド <code>TexImageParam</code> のパラメータと同様のレイアウトになっています。ただし、フリップ・リピート及びテクスチャ座標変換モードのビットは設定されていません。							
	OFS:							
	<code>TexPlttSet::texData</code> 又は <code>TexPlttSet::tex4x4Data</code> に対するオフセットを 3bit 右にシフトしたもの。							
	S:							
	テクスチャの幅							
	T:							
	テクスチャの高さ							
Fmt:								
テクスチャフォーマット								
P:								
パレットカラー設定値イネーブルフラグ								
extraParam	31 30 22 21 11 10 0							
	<table><tr><td>S</td><td></td><td>OrigH</td><td>OrigW</td></tr></table>	S		OrigH	OrigW			
	S		OrigH	OrigW				
	OrigW:							
	ツール上でのテクスチャの幅							
	OrigH:							
	ツール上でのテクスチャの高さ							
	S:							
	<code>TexImageParam</code> で指定されている幅と高さと同じ場合は 1 になる。							

また、辞書 `dictPltt` 内のデータもデータへのオフセットではなく、以下の擬似構造体で示されるデータが格納されています。

```
pseudo_struct DictPlttData(NNSG3dResDictPlttData) {
    u16 offset;
    u16 flag;
};
```

表 3-25 ディクショナリ dictPltt 内に格納されているデータ

名称	内容
offset	<code>TexPlttSet::plttData</code> に対するオフセットを 3bit 右にシフトしたもの
flag	1 ならば 4 色パレット。0 ならばそれ以外。

### 3.3 ジョイントアニメーションデータファイル(.nsbca)の構造

.nsbca ファイルはジョイントアニメーションの集合を格納しています。各ジョイントアニメーションは.ica ファイルのファイル名から拡張子を取り除いた文字列の先頭 16 文字(あまった場合は NULL 文字で埋まる)によって関連付けられます。以下に.nsbca ファイルのフォーマットについて擬似構造体を用いて解説します。

```
pseudo_struct NSBCA {
    FileHeader file_header = {
        dataBlocks = 1,
        signature = '0ACB'
    };
    JointAnmSet jntAnmSet;
};

pseudo_struct JointAnmSet(NNSG3dResJntAnmSet) {
    DataBlockHeader header = {
        kind = '0TNJ',
        size = sizeof(JointAnmSet)
    };
    Dictionary dict = {sizeUnit = 4 bytes};
    JointAnm jntAnm[dict.numEntry];
    pseudo_struct JointAnmRot3 {
        ul6 info;
        fx16 A;
        fx16 B;
    } rot3[];
    PADDING(4 bytes alignment);

    pseudo_struct JointAnmRot5 {
        fx16 data[5];
    } rot5[];
    PADDING(4 bytes alignment);

    ul6 rotIdx[];
    PADDING(4 bytes alignment);

    pseudo_struct JointAnmScaleFx16 {
        fx16 scale, invScale;
    } scaleFx16[];
    PADDING(4 bytes alignment);

    fx16 transFx16[];
    PADDING(4 bytes alignment);

    pseudo_struct JointAnmScaleFx32 {
        fx32 scale, invScale;
    } scaleFx32[];

    fx32 transFx32[];
};
```

表 3-26 JointAnmSet のデータメンバの解説

名称	内容
header	ジョイントアニメーションブロックのヘッダ領域
dict	各ジョイントアニメーションへアクセスするための辞書領域

jntAnm	各ジョイントアニメーションの本体データ										
rot3	<p>回転行列データ(6 bytes)の配列。rotIdx からインデックス参照される。</p> <div><div>1565430</div><table><tr><td>—</td><td>SD</td><td>SC</td><td>M</td><td>IdxPivot</td></tr></table><div>A</div><div>B</div><p>データの意味は NodeDataのものと同様です。</p><p>idxPivot:</p><p>回転行列のピボット要素(絶対値が1である要素)の位置(0-8)を示す。</p><p>M:</p><p>このビットが ON の場合、ピボット要素は負(つまり-1)</p><p>SC:</p><p>このビットが ON なら C は B の反対の符号を持つ</p><p>SD:</p><p>このビットが ON なら D は A の反対の符号を持つ</p><p>A, B:</p><p>回転行列の要素</p></div>	—	SD	SC	M	IdxPivot					
—	SD	SC	M	IdxPivot							
rot5	<p>回転行列データ(10 bytes)の配列。rotIdx からインデックス参照される。それぞれの要素の値の絶対値は 1 より小さくなります。</p> <div><div>1520</div><table><tr><td>_00</td><td>_12(9-11)</td></tr><tr><td>_01</td><td>_12(6-8)</td></tr><tr><td>_02</td><td>_12(3-5)</td></tr><tr><td>_10</td><td>_12(0-2)</td></tr><tr><td>_11</td><td>_12(sign)</td></tr></table><p>_00, _01, _02, _10, _11:</p><p>回転行列の要素です。右に 3bit 算術シフトをして、fx16 型の値として利用します。</p><p>_12(...):</p><p>回転行列の 2 行目 3 列目の要素は、分割されて格納されています。</p><p>3 行目は外積を用いて求めることができます。</p></div>	_00	_12(9-11)	_01	_12(6-8)	_02	_12(3-5)	_10	_12(0-2)	_11	_12(sign)
_00	_12(9-11)										
_01	_12(6-8)										
_02	_12(3-5)										
_10	_12(0-2)										
_11	_12(sign)										
rotIdx	<p>回転行列データへのインデックスを格納した配列。下位 15bit がインデックスで、最上位 1bit で rot3 か rot5 を選択するようになっている。1 の場合は rot3 を参照し、0 の場合は rot5 を参照する。jntAnm からオフセットで参照される。</p>										
scaleFx16	<p>スケール成分(fx16)を格納したデータ列。jntAnm からオフセットで参照される。</p>										
transFx16	<p>平行移動成分(fx16)を格納した配列。jntAnm からオフセットで参照される。</p>										
scaleFx32	<p>スケール成分(fx32)を格納したデータ列。jntAnm からオフセットで参照される。</p>										
transFx32	<p>平行移動成分(fx32)を格納した配列。jntAnm からオフセットで参照される。</p>										

個々のジョイントアニメーションのフォーマットは擬似構造体で表現すると以下のようになります。内部で実データ列に対するオフセット等をスケール・回転・平行移動成分ごとに持っています。

```
pseudo_struct JointAnm(NNSG3dResJntAnm) {
    AnmHeader anmHeader = {
        category0 = 'J',
        category1 = 'CA'
    };
    u16 numFrame;
    u16 numNode;
    u32 annFlag;
    u32 ofsRot3;
    u32 ofsRot5;
    u16 ofsTag[numNode];
    PADDING(4 bytes alignment);
    pseudo_struct TagData(NNSG3dJntAnmSRTTag) {
        u32 flag;
        IF (!(flag & NNS_G3D_JNTANM_SRTINFO_IDENTITY)) {
            IF (!(flag & NNS_G3D_JNTANM_SRTINFO_IDENTITY_T) &&
                !(flag & NNS_G3D_JNTANM_SRTINFO_BASE_T)) {
                JointAnmTrans<flag & NNS_G3D_JNTANM_SRTINFO_CONST_TX> tx;
                JointAnmTrans<flag & NNS_G3D_JNTANM_SRTINFO_CONST_TY> ty;
                JointAnmTrans<flag & NNS_G3D_JNTANM_SRTINFO_CONST_TZ> tz;
            }
            IF (!(flag & NNS_G3D_JNTANM_SRTINFO_IDENTITY_R) &&
                !(flag & NNS_G3D_JNTANM_SRTINFO_BASE_R)) {
                JointAnmRot<flag & NNS_G3D_JNTANM_SRTINFO_CONST_R> r;
            }
            IF (!(flag & NNS_G3D_JNTANM_SRTINFO_IDENTITY_S) &&
                !(flag & NNS_G3D_JNTANM_SRTINFO_BASE_S)) {
                JointAnmScale<flag & NNS_G3D_JNTANM_SRTINFO_CONST_SX> sx;
                JointAnmScale<flag & NNS_G3D_JNTANM_SRTINFO_CONST_SY> sy;
                JointAnmScale<flag & NNS_G3D_JNTANM_SRTINFO_CONST_SZ> sz;
            }
        }
    } tagData[numNode];
};

pseudo_struct JointAnmTrans<isConst> {
    IF (isConst) {
        fx32 const_trans;
    } ELSE {
        u32 info;
        u32 offset;
    }
};

pseudo_struct JointAnmRot<isConst> {
    IF (isConst) {
        u32 const_offset;
    } ELSE {
        u32 info;
        u32 offset;
    }
};
```

```

pseudo_struct JointAnmScale<isConst> {
    IF (isConst) {
        fx32 const_scale;
        fx32 const_invScale;
    } ELSE {
        u32 info;
        u32 offset;
    }
};

```

表 3-27 JointAnm のデータメンバの解説

名称	内容	
anmHeader	アニメーションヘッダ	
numFrame	アニメーションのフレーム数	
numNode	ジョイントアニメーションの対象になるモデルのノード数	
anmFlag	ジョイントアニメーションのオプションを指定するフラグ	
ofsRot3	JointAnm の先頭から回転行列データ(6 bytes)の配列へオフセット	
ofsRot5	JointAnm の先頭から回転行列データ(10 bytes)の配列へのオフセット	
ofsTag	JointAnm の先頭からノードに対応する tagData の要素へのオフセット	
tagData	flag	個々の tagData に入っているデータを決定するフラグ群
	tx	ジョイントの平行移動ベクトルのx成分に関するデータ
	ty	ジョイントの平行移動ベクトルのy成分に関するデータ
	tz	ジョイントの平行移動ベクトルのz成分に関するデータ
	r	ジョイントの回転行列に関するデータ
	sx	ジョイントの x 方向のスケールに関するデータ
	sy	ジョイントの y 方向のスケールに関するデータ
	sz	ジョイントの z 方向のスケールに関するデータ

表 3-28 JointAnmTrans のデータメンバの解説

名称	内容
const_trans	定数の平行移動成分の値
info	Translation データ列の特徴を記述したフラグ
offset	JointAnm の先頭から Translation データ列へのオフセット

表 3-29 JointAnmRot のデータメンバの解説

名称	内容
const_offset	定数の回転行列へのインデックス値
info	Rotation データ列の特徴を記述したフラグ
offset	JointAnm の先頭から Rotation データインデックス列へのオフセット

表 3-30 JointAnmScale のデータメンバの解説

名称	内容
const_scale	定数のスケール値
const_invScale	定数のスケール値の逆数
info	Scale データ列の特徴を記述したフラグ
offset	JointAnm の先頭から Scale データ列へのオフセット

表 3-31 JointAnm::TagData::flag がとる値の解説

名称	値	内容
NNS_G3D_JNTANM_SRTINFO_IDENTITY	0x00000001	SRT に何も変更がないとき ON
NNS_G3D_JNTANM_SRTINFO_IDENTITY_T	0x00000002	平行移動しないとき ON
NNS_G3D_JNTANM_SRTINFO_BASE_T	0x00000004	Trans にモデルの値を使うとき ON
NNS_G3D_JNTANM_SRTINFO_CONST_TX	0x00000008	Tx が定数の場合 ON
NNS_G3D_JNTANM_SRTINFO_CONST_TY	0x00000010	Ty が定数の場合 ON
NNS_G3D_JNTANM_SRTINFO_CONST_TZ	0x00000020	Tz が定数の場合 ON
NNS_G3D_JNTANM_SRTINFO_IDENTITY_R	0x00000040	回転がない場合 ON
NNS_G3D_JNTANM_SRTINFO_BASE_R	0x00000080	Rot にモデルの値を使うとき ON
NNS_G3D_JNTANM_SRTINFO_CONST_R	0x00000100	Rot が定数の場合 ON
NNS_G3D_JNTANM_SRTINFO_IDENTITY_S	0x00000200	Scale がかからない場合 ON
NNS_G3D_JNTANM_SRTINFO_BASE_S	0x00000400	Scale にモデルの値を使う場合 ON
NNS_G3D_JNTANM_SRTINFO_CONST_SX	0x00000800	Sx が定数の場合 ON
NNS_G3D_JNTANM_SRTINFO_CONST_SY	0x00001000	Sy が定数の場合 ON
NNS_G3D_JNTANM_SRTINFO_CONST_SZ	0x00002000	Sz が定数の場合 ON
NNS_G3D_JNTANM_SRTINFO_NODE_MASK	0xFF000000	アニメーション対象のノードIDが入る場所のマスク

表 3-32 JointAnmTrans::info がとる値の解説

名称	値	内容
NNS_G3D_JNTANM_TINFO_STEP_1	0x00000000	データが毎フレームあるとき
NNS_G3D_JNTANM_TINFO_STEP_2	0x40000000	フレームステップが 2 のときに ON
NNS_G3D_JNTANM_TINFO_STEP_4	0x80000000	フレームステップが 4 のときに ON
NNS_G3D_JNTANM_TINFO_FX16ARRAY	0x20000000	アニメーションデータが fx16 の配列の場合 ON
NNS_G3D_JNTANM_TINFO_LAST_INTERP_MASK	0x1FFF0000	frameStep=2,4 のとき: (numFrame-1) & ~(frameStep-1) frameStep=1 のとき: numFrame



表 3-33 JointAnmRot::info がとる値の解説

名称	値	内容
NNS_G3D_JNTANM_RINFO_STEP_1	0x00000000	データが毎フレームあるとき
NNS_G3D_JNTANM_RINFO_STEP_2	0x40000000	フレームステップが 2 のときにON
NNS_G3D_JNTANM_RINFO_STEP_4	0x80000000	フレームステップが 4 のときにON
NNS_G3D_JNTANM_RINFO_LAST_INTERP_MASK	0x1FFF0000	frameStep=2,4 のとき: (numFrame-1) & ~(frameStep - 1)  frameStep=1 のとき: numFrame

表 3-34 JointAnmScale::info がとる値の解説

名称	値	内容
NNS_G3D_JNTANM_SINFO_STEP_1	0x00000000	データが毎フレームあるとき
NNS_G3D_JNTANM_SINFO_STEP_2	0x40000000	フレームステップが 2 のときにON
NNS_G3D_JNTANM_SINFO_STEP_4	0x80000000	フレームステップが 4 のときにON
NNS_G3D_JNTANM_SINFO_FX16ARRAY	0x20000000	アニメーションデータが fx16 の配列の場合 ON
NNS_G3D_JNTANM_SINFO_LAST_INTERP_MASK	0x1FFF0000	frameStep=2,4 のとき: (numFrame-1) & ~(frameStep - 1)  frameStep=1 のとき: numFrame

### 3.4 テクスチャパターンアニメーションデータファイル(.nsbtp)の構造

.nsbtp ファイルはテクスチャパターンアニメーションの集合を格納しています。各テクスチャアニメーションは.itp ファイルのファイル名から拡張子を取り除いた文字列の先頭 16 文字(あまった場合は NULL 文字で埋まる)によって関連付けられます。以下に.nsbtp ファイルのフォーマットについて擬似構造体を用いて解説します。

```
pseudo_struct NSBTP {
    FileHeader file_header = {
        dataBlocks = 1,
        signature = '0PTB'
    };
    TexPatAnmSet texPatAnmSet;
};

pseudo_struct TexPatAnmSet(NNSG3dResTexPatAnmSet) {
    DataBlockHeader header = {
        kind = '0TAP',
        size = SIZEOF(TexPatAnmSet)
    };
    Dictionary dict = {sizeUnit = 4 bytes};
    TexPatAnm texPatAnm[dict.numEntry];
};
```

表 3-35 TexPatAnmSet のデータメンバの解説

名称	内容
header	テクスチャパターンアニメーションブロックのヘッダ領域
dict	各テクスチャパターンアニメーションへアクセスするための辞書領域
texPatAnm	各テクスチャパターンアニメーションの本体データ

```
pseudo_struct TexPatAnm(NNSG3dResTexPatAnm) {
    AnmHeader anmHeader = {
        category0 = 'M',
        category1 = 'TP'
    };
    u16 numFrame;
    u8 numTex;
    u8 numPltt;
    u16 ofsTexName;
    u16 ofsPlttName;
    Dictionary dict(sizeUnit = 8 bytes);

    pseudo_struct TexPatFV(NNSG3dResTexPatAnmFV) {
        u16 idxFrame;
        u8 idTex;
        u8 idPltt;
    } texPatFV[];

    pseudo_struct DictName(NNSG3dResName) {
        u8 name[16];
    } texName[numTex];

    pseudo_struct DictName(NNSG3dResName) {
        u8 name[16];
    } plttName[numPltt];
};
```

表 3-36 TexPatAnm のデータメンバの解説

名称	内容	
anmHeader	アニメーションヘッダ	
numFrame	アニメーションのフレーム数	
numTex	アニメーションするテクスチャの総数	
numPltt	アニメーションするパレットの総数	
ofsTexName	TexPatAnm の先頭から texName へのオフセット	
ofsPlttName	TexPatAnm の先頭から plttName へのオフセット	
dict	マテリアル名からアニメーションデータにアクセスするための辞書	
texPatFV	idxFrame	下記のテクスチャ・パレットに切り替わるフレーム番号
	idTex	texName に対するインデックス
	idPltt	plttName に対するインデックス
texName	アニメーションで表示されるテクスチャ名の配列	
plttName	アニメーションで表示されるパレット名の配列	

```

pseudo_struct DictTexPatAnmData(NNSG3dResDictTexPatAnmData) {
    u16 numFV;
    u16 flag;
    fx16 ratioDataFrame;
    u16 offset;
};

```

表 3-37 ディクショナリ TexPatAnm::dict に格納されているデータ

名称	内容
numFV	FV データの数
flag	<div>15<span style="float: right;">0</span></div> <div style="border: 1px solid black; padding: 2px; display: inline-block; width: 100%;">---</div> <div style="border: 1px solid black; padding: 2px; display: inline-block; width: 100%;">P</div> <p>P: 1 ならば、パレットのアニメーションは存在しません。</p>
ratioDataFrame	FV データの数をフレーム数で割った数です。現在のフレームから FV データを検索するためのヒントとして利用できます。
offset	TexPatAnm を起点とした FV データの並び(TexPatAnm::texPatFV 内)へのオフセットです。

テクスチャパターンアニメーションはマテリアルアニメーションの一種で、指定したマテリアルに属するテクスチャやパレットを記録されている FV(Frame-Value)データによって切り替えます。.nsbtp ファイルには、実際のテクスチャデータは存在せず、テクスチャやパレットの名前とその切り替えタイミングについての情報のみが記録されています。

### 3.5 マテリアルカラーアニメーションデータファイル(.nsbma)の構造

.nsbma ファイルはマテリアルカラーアニメーションの集合を格納しています。各マテリアルカラーアニメーションは.ima ファイルのファイル名から拡張子を取り除いた文字列の先頭 16 文字(あまった場合は NULL 文字で埋まる)によって関連付けられます。以下に.nsbma ファイルのフォーマットについて擬似構造体を用いて解説します。

```
pseudo_struct NSBMA {
    FileHeader file_header = {
        dataBlocks = 1,
        signature = '0AMB'
    };
    MatColAnmSet matColAnmSet;
};

pseudo_struct MatColAnmSet(NNSG3dResMatCAnmSet) {
    DataBlockHeader header = {
        kind = '0TAM',
        size = SIZEOF(MatColAnmSet)
    };
    Dictionary dict = {sizeUnit = 4 bytes};
    MatColAnm matColAnm[dict.numEntry];
};
```

表 3-38 MatColAnmSet のデータメンバの解説

名称	内容
header	マテリアルカラーアニメーションブロックのヘッダ領域
dict	各マテリアルカラーアニメーションへアクセスするための辞書領域
matColAnm	各マテリアルカラーアニメーションの本体データ

```
pseudo_struct MatColAnm(NNSG3dResMatCAnm) {
    AnmHeader anmHeader = {
        category0 = 'M',
        category1 = 'MA'
    };
    u16 numFrame;
    u16 flag;
    Dictionary dict(sizeUnit = 20 bytes);
    u16 rgbData[];
    PADDING(4 bytes alignment);
    u8 alphaData[];
    PADDING(4 bytes alignment);
};
```

表 3-39 MatColAnm のデータメンバの解説

名称	内容
anmHeader	アニメーションヘッダ
numFrame	アニメーションのフレーム数
flag	マテリアルカラーアニメーションのオプションを指定するフラグ
dict	マテリアル名からアニメーションデータにアクセスするための辞書
rgbData	RGB のアニメーションデータ
alphaData	Alpha のアニメーションデータ

表 3-40 MatColAnm::flag がとる値の解説

名称	値	内容
NNS_G3D_MATCANM_OPTION_INTERPOLATION	0x0001	色の補間をする
NNS_G3D_MATCANM_OPTION_END_TO_START_INTERPOLATION	0x0002	終了フレームから開始フレームへの色補間をする(ループ用)

DictMatColAnmData擬似構造体は、カラーやアルファのアニメーションに関する管理情報を保持していて、MatColAnm内のディクショナリのエントリとして格納されています。以下に定義を示し、各データメンバがとる値を表 3-41で説明します。

```
pseudo_struct DictMatColAnmData(NNSG3dResDictMatColAnmData) {
    u32 tagDiffuse;
    u32 tagAmbient;
    u32 tagSpecular;
    u32 tagEmission;
    u32 tagPolygonAlpha;
};
```

表 3-41 DictMatColAnmData のデータメンバの解説

名称	内容
tagDiffuse	Diffuse のアニメーションに関する管理情報
tagAmbient	Ambient のアニメーションに関する管理情報
tagSpecular	Specular のアニメーションに関する管理情報
tagEmission	Emission のアニメーションに関する管理情報
tagPolygonAlpha	Polygon Alpha のアニメーションに関する管理情報

表 3-42 tagDiffuse/tagAmbient/tagSpecular/tagEmission/tagPolygonAlpha がとる値の解説

名称	値	内容
NNS_G3D_MATCANM_ELEM_STEP_1	0x00000000	データが毎フレームあるとき
NNS_G3D_MATCANM_ELEM_STEP_2	0x40000000	フレームステップが 2 のときに ON
NNS_G3D_MATCANM_ELEM_STEP_4	0x80000000	フレームステップが 4 のときに ON
NNS_G3D_MATCANM_ELEM_CONST	0x20000000	ON ならば下位 16 ビットは定数データとして扱われる
NNS_G3D_MATCANM_ELEM_LAST_INTERP_MASK	0x1FFF0000	frameStep=2,4 のとき: (numFrame-1) & ~(frameStep-1) frameStep=1 のとき: numFrame
NNS_G3D_MATCANM_ELEM_OFFSET_CONSTANT_MASK	0x0000FFFF	MatColAnm を起点とするデータ列へのオフセットが定数値

### 3.6 ビジビリティアニメーションデータファイル(.nsbva)の構造

.nsbva ファイルはビジビリティアニメーションの集合を格納しています。各ビジビリティアニメーションは.iva ファイルのファイル名から拡張子を取り除いた文字列の先頭 16 文字(あまった場合は NULL 文字で埋まる)によって関連付けられます。以下に.nsbva ファイルのフォーマットについて擬似構造体を用いて解説します。

```
pseudo_struct NSBVA {
    FileHeader file_header = {
        dataBlocks = 1,
        signature = '0AVB'
    };
    VisAnmSet visAnmSet;
};

pseudo_struct VisAnmSet(NNSG3dResVisAnmSet) {
    DataBlockHeader header = {
        kind = '0SIV',
        size = SIZEOF(MatColAnmSet)
    };
    Dictionary dict = {sizeUnit = 4 bytes};
    VisAnm visAnm[dict.numEntry];
};
```

表 3-43 VisAnmSet のデータメンバの解説

名称	内容
header	ビジビリティアニメーションブロックのヘッダ領域
dict	各ビジビリティアニメーションへアクセスするための辞書領域
visAnm	各ビジビリティアニメーションの本体データ

```
pseudo_struct VisAnm(NNSG3dResVisAnm) {
    AnmHeader anmHeader = {
        category0 = 'V',
        category1 = 'VA'
    };
    u16 numFrame;
    u16 numNode;
    u16 size = SIZEOF(VisAnm);
    PADDING(2 bytes);
    u32 visData[1 + (numFrame * numNode >> 5)];
};
```

表 3-44 VisAnm のデータメンバの解説

名称	内容
anmHeader	アニメーションヘッダ
numFrame	アニメーションのフレーム数
numNode	アニメーションさせるモデルのノード数
size	VisAnm 擬似構造体のサイズ
visData	ビジビリティアニメーションのデータ。各 1 ビットがノードが見える・見えないという情報に対応している。リトルエンディアンの場合、CurFrame * numNode + nodeID ビット目に CurFrame フレームの nodeID ノードのビジビリティ情報が格納されている。

### 3.7 テクスチャSRTアニメーションデータファイル(.nsbta)の構造

.nsbta ファイルはテクスチャ SRT アニメーションの集合を格納しています。各テクスチャSRTアニメーションは.ita ファイルのファイル名から拡張子を取り除いた文字列の先頭 16 文字(あまった場合は NULL 文字で埋まる)によって関連付けられます。以下に.nsbta ファイルのフォーマットについて擬似構造体を用いて解説します。

```
pseudo_struct NSBTA {
    FileHeader file_header = {
        dataBlocks = 1,
        signature = '0ATB'
    };
    TexSRTAnmSet texSRTAnmSet;
};

pseudo_struct TexSRTAnmSet(NNSG3dResTexSRTAnmSet) {
    DataBlockHeader header = {
        kind = '0TRS',
        size = SIZEOF(TexSRTAnmSet)
    };
    Dictionary dict = {sizeUnit = 4 bytes};
    TexSRTAnm texSRTAnm[dict.numEntry];
};
```

表 3-45 TexSRTAnmSet のデータメンバの解説

名称	内容
header	ビジュアルアニメーションブロックのヘッダ領域
dict	各テクスチャ SRT アニメーションへアクセスするための辞書領域
texSRTAnm	各テクスチャ SRT アニメーションの本体データ

```
pseudo_struct TexSRTAnm(NNSG3dResTexSRTAnm) {
    AnmHeader anmHeader = {
        category0 = 'M',
        category1 = 'TA'
    };
    u16 numFrame;
    u8 flag;
    u8 texMtxMode;
    Dictionary dict(sizeUnit = 40 bytes);
    u32 anmData[];
};
```

表 3-46 TexSRTAnm のデータメンバの解説

名称	内容
anmHeader	アニメーションヘッダ
numFrame	アニメーションのフレーム数
flag	テクスチャ SRT アニメーションのオプションを指定するフラグ
texMtxNode	テクスチャ行列の計算法。表 3-10 を参照
dict	マテリアル名からアニメーションデータにアクセスするための辞書
anmData	DictTexSRTAnmDataから参照される各種数値データ列

表 3-47 TexSRTAnm::flag がとる値の解説

名称	値	内容
NNS_G3D_TEXSRTANM_OPTION_INTERPOLATION	0x0001	補間再生の指定
NNS_G3D_TEXSRTANM_OPTION_END_TO_START_INTERPOLATION	0x0002	終了フレームから開始フレームへの補間をする(ループ用)

```

pseudo_struct DictTexSRTAnmData(NNSG3dResDictTexSRTAnmData) {
    u32 scaleS;
    u32 scaleSEx;
    u32 scaleT;
    u32 scaleTEx;
    u32 rot;
    u32 rotEx;
    u32 transS;
    u32 transSEx;
    u32 transT;
    u32 transTEx;
};

```

表 3-48 DictTexSRTAnmData のデータメンバの解説

名称	内容
scaleS	テクスチャのスケールアニメーション(縦方向)に関する管理情報
scaleSEx	scaleS に NNS_G3D_TEXSRTANM_ELEM_CONST が設定されているときは定数値(即値)、それ以外の場合は TexSRTAnm の先頭からデータ列へのオフセット
scaleT	テクスチャのスケールアニメーション(縦方向)に関する管理情報
scaleTEx	scaleT に NNS_G3D_TEXSRTANM_ELEM_CONST が設定されているときは定数値(即値)、それ以外の場合は TexSRTAnm の先頭からデータ列へのオフセット
rot	テクスチャの回転アニメーションに関する管理情報
rotEx	rot に NNS_G3D_TEXSRTANM_ELEM_CONST が設定されているときは sin と cos をパックした定数値(即値)になる(上位 16bit が fx16 の cos 値、下位 16bit が fx16 の sin 値)、それ以外の場合は TexSRTAnm の先頭からデータ列へのオフセット
transS	テクスチャの平行移動アニメーション(横方向)に関する管理情報
transSEx	transS に NNS_G3D_TEXSRTANM_ELEM_CONST が設定されているときは定数値(即値)、それ以外の場合は TexSRTAnm の先頭からデータ列へのオフセット
transT	テクスチャの平行移動アニメーション(縦方向)に関する管理情報
transTEx	transT に NNS_G3D_TEXSRTANM_ELEM_CONST が設定されているときは定数値(即値)、それ以外の場合は TexSRTAnm の先頭からデータ列へのオフセット



表 3-49 scaleS/scaleT/rot/transS/transT がとる値の解説

名称	値	内容
NNS_G3D_TEXSRTANM_ELEM_STEP_1	0x00000000	データが毎フレームあるとき
NNS_G3D_TEXSRTANM_ELEM_STEP_2	0x40000000	フレームステップが 2 のときにON
NNS_G3D_TEXSRTANM_ELEM_STEP_4	0x80000000	フレームステップが 4 のときにON
NNS_G3D_TEXSRTANM_ELEM_CONST	0x20000000	ON ならば下位 16 ビットは定数データとして扱われる
NNS_G3D_TEXSRTANM_ELEM_FX16	0x10000000	ON ならばデータを fx16 の配列でもつ
NNS_G3D_TEXSRTANM_ELEM_LAST_INTERP_MASK	0x0000FFFF	frameStep=2,4 のとき: (numFrame-1) & ~(frameStep - 1) frameStep=1 のとき: numFrame

Softimage、SOFTIMAGE|3D、SOFTIMAGE|XSI は米国 Avid Technology,Inc. の登録商標または商標です。

3ds max、Maya は Autodesk,Inc./Autodesk Canada,Inc. の米国およびその他の国における登録商標または商標です。

その他、記載されている会社名、製品名等は、各社の登録商標または商標です。

©2004–2008 Nintendo

任天堂株式会社の許諾を得ることなく、本書に記載されている内容の一部あるいは全部を無断で複製・複写・転写・頒布・貸与することを禁じます。