

G2D ライブラリランタイムバイナリフォーマット ファイルフォーマットの説明

2009-04-01

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	7
1.1	G2Dランタイムバイナリフォーマット	7
2	G2Dランタイムバイナリフォーマット	8
2.1	G2Dランタイムバイナリの種類	8
2.2	TWL-Systemバイナリファイル規約	8
2.3	オフセット(ポインタ)表記について	8
3	NCER(セル定義情報)	9
3.1	セルバンクブロック	9
3.2	解説	12
3.2.1	NNSG2dCellDataBank. cellBankAttr	12
3.2.2	NNSG2dCellDataBank. mappingMode	12
3.2.3	NNSG2dCellDataBank. pVramTransferData	12
3.2.4	NNSG2dCellData	12
3.2.5	OAMアトリビュート情報	13
3.2.6	ユーザ拡張アトリビュート情報	13
4	NANR、NMAR(アニメーション定義情報)	14
4.1	アニメーションバンクブロック	15
4.2	解説	18
4.2.1	NNSG2dAnimBankData.playMode	18
4.2.2	NNSG2dAnimSequenceData.animType	18
4.2.3	アニメーション結果情報列	19
4.2.4	ユーザ拡張アトリビュート情報	20
5	NCGR,NCBR(キャラクタ情報)	21
5.1	キャラクター定義ブロック	21
5.2	キャラクター位置情報ブロック	22
5.3	解説	23
5.3.1	NNSG2dCharacterData.W .H	23
5.3.2	NNSG2dCharacterData.pixelFmt	23
5.3.3	NNSG2dCharacterData.characterFmt	23
5.3.4	NNSG2dCharacterData.mapingType	23
5.3.5	NNSG2dCharacterPosInfoBlock	24
6	NCLR(カラーパレット情報)	25
6.1	パレット定義ブロック	25
6.2	パレット圧縮情報ブロック	26
6.3	説明	27
6.3.1	NNSG2dPaletteCompressInfo	27

7	NMCR(マルチセル情報)	28
7.1	マルチセルバンクブロック	28
7.2	解説	31
7.2.1	NNSG2dMultiCellHierarchyData. nodeAttr	31
7.2.2	ユーザ拡張アトリビュート情報	31
8	NENR(エンティティ情報)	32
8.1	エンティティバンクブロック	32
9	NSCR(スクリーン情報)	35
9.1	スクリーン定義ブロック	35
9.2	解説	36
9.2.1	NNSG2dScreenData. ColorMode	36
9.2.2	NNSG2dScreenData. screenFormat	36
10	NFTR(フォント情報)	37
10.1	フォント情報ブロック	37
10.2	グリフィメージブロック	38
10.3	文字幅ブロック	39
10.4	文字コードマップブロック	40
10.5	解説	42
10.5.1	NNSG2dFontInformation.fontType	42
10.5.2	NNSG2dFontInformation.pGlyph / pWidth / pMap	42
10.5.3	NNSG2dFontGlyph.flags	42
10.5.4	NNSG2dFontGlyph.glyphTable	42
10.5.5	NNSG2dFontWidth.indexBegin / indexEnd / widthTable	42
10.5.6	NNSG2dFontCodeMap.ccodeBegin / ccodeEnd	43
10.5.7	NNSG2dFontCodeMap.mpppingMethod / mapInfo	43

表

表 2-1	G2Dランタイムバイナリの種類	8
表 3-1	NCERブロック構成	9
表 3-2	NNSG2dCellBankBlock	10
表 3-3	NNSG2dCellDataBank	10
表 3-4	NNSG2dCellData	10
表 3-5	NNSG2dCellBoundingRectS16	10
表 3-6	NNSG2dCellDataWithBR	11
表 3-7	NNSG2dVramTransferData	11
表 3-8	NNSG2dCellVramTransferData	11
表 3-9	NNSG2dUserExCellAttrBank	11
表 4-1	NANR NMARブロック構成	14
表 4-2	NNSG2dAnimBankDataBlock	16
表 4-3	NNSG2dAnimBankData	16
表 4-4	NNSG2dAnimSequenceData	16

表 4-5 NNSG2dAnimFrameData	16
表 4-6 NNSG2dUserExAnimAttrBank	17
表 4-7 NNSG2dUserExAnimSequenceAttr	17
表 4-8 NNSG2dUserExAnimFrameAttr	17
表 5-1 NCGR (NCBR)ブロック構成	21
表 5-2 NNSG2dCharacterDataBlock	22
表 5-3 NNSG2dCharacterData	22
表 5-4 NNSG2dCharacterPosInfo	22
表 6-1 NCLRブロック構成	25
表 6-2 NNSG2dPaletteDataBlock	25
表 6-3 NNSG2dPaletteData	26
表 6-4 NNSG2dPaletteCompressDataBlock	26
表 6-5 NNSG2dPaletteCompressInfo	27
表 7-1 NMCRブロック構成	28
表 7-2 NNSG2dMultiCellDataBankBlock構成	29
表 7-3 NNSG2dMultiCellDataBank	30
表 7-4 NNSG2dMultiCellData	30
表 7-5 NNSG2dMultiCellHierarchyData	30
表 8-1 NENRブロック構成	32
表 8-2 NNSG2dEntityDataBankBlock構成	33
表 8-3 NNSG2dEntityDataBank	33
表 8-4 NNSG2dEntityData	33
表 8-5 NNSG2dEntityAnimData	33
表 9-1 NSCRブロック構成	35
表 9-2 NNSG2dScreenDataBlock	35
表 9-3 NNSG2dScreenData	36
表 10-1 NFTRブロック構成	37
表 10-2 NNSG2dFontInformationBlock	38
表 10-3 NNSG2dFontInformation	38
表 10-4 NNSG2dCharWidths	38
表 10-5 NNSG2dFontGlyphBlock	39
表 10-6 NNSG2dFontGlyph	39
表 10-7 NNSG2dFontWidthBlock	40
表 10-8 NNSG2dFontWidth	40
表 10-9 NNSG2dFontCodeMapBlock	41
表 10-10 NNSG2dFontCodeMap	41
表 10-11 NNSG2dCMapInfoScan	41
表 10-12 NNSG2dCMapScanEntry	41



図 3-1 NNSG2dCellBankBlock構成	9
図 3-2 セルデータバンクの大まかな構成	11
図 4-1 NNSG2dCellBankBlock構成	15
図 4-2 アニメーションデータ の大まかな構成	18
図 5-1 NNSG2dCharacterDataBlock構成	21

図 5-2 NNSG2dCharacterPosInfoBlock構成.....	22
図 6-1 NNSG2dPaletteDataBlock構成.....	25
図 6-2 NNSG2dPaletteCompressDataBlock構成	26
図 7-1 NNSG2dMultiCellDataBankBlock構成	28
図 7-2 マルチセルデータのおおまかな構成.....	31
図 8-1 NNSG2dEntityDataBankBlock構成.....	32
図 8-2 Entityデータの大まかな構成.....	34
図 9-1 NNSG2dScreenDataBlock構成.....	35
図 10-1 NNSG2dFontInformationBlock構成	37
図 10-2 NNSG2dFontGlyphBlock構成	39
図 10-3 NNSG2dFontWidthBlock構成	40
図 10-4 NNSG2dFontCodeMapBlock構成.....	41
図 10-5 フォントデータの大まかな構成.....	42

改訂履歴

改訂日	改訂内容
2009-04-01	CPOS ブロックの出力条件について、記述を修正。
2008-09-12	NFTR ファイルフォーマットの更新。
2008-05-30	NITRO-System の名称変更による修正 (NITRO-System を TWL-System に変更)。 誤植の修正。
2008-04-08	改訂履歴の書式を変更。
2007-03-14	NFTR ファイルフォーマットの更新。
2005-09-01	誤植の修正。 NCGR,NCBR ファイルフォーマット変更。記述の追加。 ユーザ拡張アトリビュートに関する記述を追加。
2005-05-25	NFTR ファイルフォーマットを追加。
2005-01-31	セル情報に、OAM アトリビュート情報の説明を追加。 キャラクタデータのサイズ情報に関する注意を記述。
2005-01-31	マルチセルデータフォーマットの変更。 NNSG2dMultiCellData.numTotalOams の廃止 NNSG2dMultiCellData.numCellAnim の追加。 NNSG2dMultiCellHierarchyData.nodeAttr の内容変更。
2004-12-06	語句の統一。
2004-10-12	矩形領域情報を持つセルデータに関する記述を追加。
2004-10-04	パレット圧縮情報に関する記述を追加。
2004-09-16	ファイル識別子情報を記述。
2004-09-02	NSCR ファイルフォーマットを追加。
2004-09-02	9 月 02 日版に対応。表記の統一。
2004-08-02	8 月 02 日版に対応。コンバータ名称変更に対応。
2004-07-20	7 月 20 日版のフォーマット変更を反映
2004-06-22	6 月 22 日版のフォーマット変更を反映
2004-05-27	フォーマット変更を反映
2004-05-11	初版作成

1 はじめに

G2D ランタイムバイナリフォーマットとは G2D ランタイムが処理可能なバイナリファイル形式です。本文章は G2D ランタイムバイナリフォーマットについて説明をします。

1.1 G2Dランタイムバイナリフォーマット

G2D ランタイムバイナリを作成するためには、NITRO-CHARACTER が出力する中間バイナリファイルをバイナリコンバータ `g2dcvtr.exe` によって変換する必要があります。`g2dcvtr.exe` の詳細については、バイナリコンバータ関連ドキュメントを参照してください。(TwlSystem¥docs¥G2D¥g2dcvtr_Manual.pdf)

以降では、具体的なファイルフォーマットについて説明していきます。

2 G2Dランタイムバイナリフォーマット

本章では、G2D ランタイムバイナリフォーマットの具体的な説明を行います。

2.1 G2Dランタイムバイナリの種類

ランタイムバイナリは、格納するデータの種類に応じて、以下の表のようなファイル形式を持ちます。

表 2-1 G2D ランタイムバイナリの種類

拡張	意味	説明	ファイル識別子
ncer	<u>N</u> itro <u>C</u> ell for <u>R</u> untime	セル定義情報	NCER
nanr	<u>N</u> itro <u>A</u> nimation for <u>R</u> untime	アニメーション定義情報	NANR
nmar	<u>N</u> itro <u>M</u> ulticell <u>A</u> nimation for <u>R</u> untime	アニメーション定義情報	NMAR
ncgr	<u>N</u> itro <u>C</u> haracter <u>G</u> raphics for <u>R</u> untime	キャラクタデータ	NCGR
ncbr	<u>N</u> itro <u>C</u> haracter <u>B</u> itmap format for <u>R</u> untime	キャラクタ(ビットマップ)	NCBR
nclr	<u>N</u> itro <u>C</u> o <u>L</u> or palette for <u>R</u> untime	カラーパレット	NCLR, NCPR(古いバージョン)
nmcr	<u>N</u> itro <u>M</u> ulti <u>C</u> ell for <u>R</u> untime	マルチセル定義情報	NMCR
nenr	<u>N</u> itro <u>E</u> Ntity for <u>R</u> untime	エンティティ定義	NENR
nscr	<u>N</u> itro <u>S</u> Creen for <u>R</u> untime	スクリーン定義	NSCR

2.2 TWL-Systemバイナリファイル規約

G2D ランタイムバイナリは TWL-System バイナリファイル規約に従った、ファイル構造をとっています。

したがって、大まかなファイル構造においては TWL-System バイナリファイル規約が定めるバイナリファイルヘッダ + バイナリブロック(任意個)の形式をとっています。

詳細については、TWL-System バイナリファイル規約関連ドキュメントを参照してください。

(TwlSystem¥docs¥Readme¥DataFormatRule.pdf)

以降の章では、各ファイルフォーマットについて詳細な説明を行います。

2.3 オフセット(ポインタ)表記について

以降の説明において、オフセット(ポインタ)という表記が登場します。これは、ファイル中ではオフセットバイト値を記録しており、実行時にアドレス変換を行いポインタメンバとして使用される変数のことを表しています。

3 NCER(セル定義情報)

本章では、NCER ファイルフォーマットについて説明を行います。

NCER はセル定義情報を格納する、ファイルフォーマットです。

NITRO-CHARACTER バイナリファイル*.nce ファイルをコンバートし生成します。

NCER は、おおまかには、下表のようなブロック構成となっています。

表 3-1 NCER ブロック構成

データブロック名	識別子	値	条件	備考
セルバンクブロック	NNS_G2D_BLKSIG_CELLBANK	'CEBK'	必須	固有
ユーザ拡張情報ブ	NNS_G2D_BLKSIG_USEREXT	'UEXT'	任意	共通
名前ラベル情報	NNS_G2D_BLKSIG_NAMELABEL	'LABL'	任意	共通

NCER ファイル固有ブロックについて説明をしていきます。

3.1 セルバンクブロック

セル情報を定義するブロックです。本ブロックの大まかな構成は以下のようになっています。

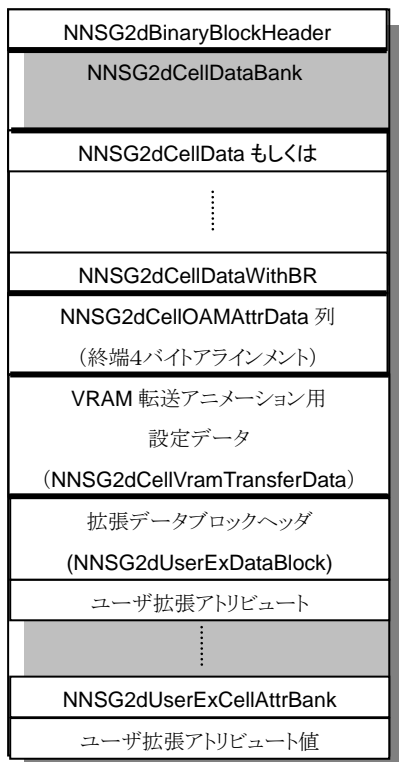


図 3-1 NNSG2dCellBankBlock 構成

各データ構造の内容は以下のようになっています。

表 3-2 NNSG2dCellBankBlock

型	パラメータ名	説明	バイト
NNSG2dBinaryBlockHeader	blockHeader	ブロックヘッダ	8
NNSG2dCellDataBank	cellDataBank	セルデータバンク	24

表 3-3 NNSG2dCellDataBank

型	パラメータ名	説明	バイト
u16	numCells	セルデータ数	2
u16	cellBankAttr	セルバンク属性	2
NNSG2dCellData*	pCellDataArrayHead	セルデータ配列へのオフセット(ポインタ)	4
NNSG2dCharacterDataMappingType	mappingMode	参照キャラクタのマッピング形式	4
NNSG2dVramTransferData*	pVramTransferData	VRAM 転送アニメに使用するデータへのポインタ(予約) 未使用時は NULL	4
void*	pStringBank	文字列バンクへのポインタ (実行時に設定されます)	4
void*	pExtendedData	ライブラリ拡張情報へのポインタ	4

表 3-4 NNSG2dCellData

型	パラメータ名	説明	バイト
u16	numOAMAttrs	セル構成 OAM アトリビュート数	2
u16	cellAttr	セル その他情報 (Flip 機能使用状況など)	2
NNSG2dCelloAMAttrData*	pOamAttrArray	セル構成 OAM アトリビュート列への オフセット(ポインタ)	4

表 3-5 NNSG2dCellBoundingRectS16

型	パラメータ名	説明	バイト
s16	maxX	セル境界最大 X 値	2
s16	maxY	セル境界最大 Y 値	2
s16	minX	セル境界最小 X 値	2
s16	minY	セル境界最小 Y 値	2

表 3-6 NNSG2dCellDataWithBR

型	パラメータ名	説明	バイト
NNSG2dCellData	cellData	セルデータ	8
NNSG2dCellDataWithBR	boundingRect	矩形領域情報	8

表 3-7 NNSG2dVramTransferData

型	パラメータ名	説明	バイト
u32	szByteMax	すべての VRAM 転送中の最大バイト数	4
NNSG2dCellVramTransferData*	pCellTransferDataArray	セル転送情報配列の先頭へのオフセット (ポインタ)	4

表 3-8 NNSG2dCellVramTransferData

型	パラメータ名	説明	バイト
u32	srcDataOffset	転送元データ先頭からのオフセット値	4
u32	szByte	転送バイトサイズ	4

表 3-9 NNSG2dUserExCellAttrBank

型	パラメータ名	説明	バイト
u16	numCells	セル数	2
u16	numAttribute	セルごとのアトリビュート数(現在は1固定)	2
NNSG2dUserExCellAttr*	pCellAttrArray	NNSG2dUserExCellAttr 配列の先頭	4

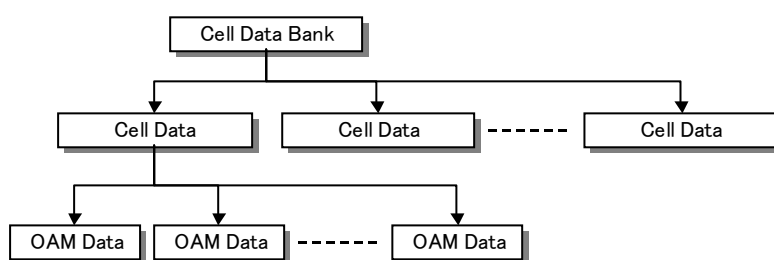


図 3-2 セルデータバンクの大まかな構成

3.2 解説

3.2.1 NNSG2dCellDataBank. cellBankAttr

cellBankAttr はセルデータバンクの持つ属性情報を格納しています。

現在格納されている情報を列挙します。格納ビット位置等は変更になる可能性がありますので、アクセサ関数を使用して情報にアクセスを行います。

- セルデータの形式(NNSG2dCellData もしくは NNSG2dCellDataWithBR)

3.2.2 NNSG2dCellDataBank. mappingMode

mappingMode は、セルが参照するキャラクタデータのフォーマット情報です。

キャラクタデータのフォーマット情報の列挙型 NNSG2dCharacterDataMappingType の宣言は以下のとおりです。

```
typedef enum NNSG2dCharacterDataMappingType
{
    NNS_G2D_CHARACTERMAPING_1D_32,
    NNS_G2D_CHARACTERMAPING_1D_64,
    NNS_G2D_CHARACTERMAPING_1D_128,
    NNS_G2D_CHARACTERMAPING_1D_256,
    NNS_G2D_CHARACTERMAPING_2D,
    NNS_G2D_CHARACTERMAPING_MAX
}NNSG2dCharacterDataMappingType;
```

3.2.3 NNSG2dCellDataBank. pVramTransferData

NNSG2dCellDataBank は VRAM 転送アニメーションを実装する際に必要となるデータへのポインタをメンバに持ちます。

VRAM 転送による描画が意図された、セル定義データは pVramTransferData に有効なデータがセットされています。(未使用時には NULL 値が設定されています)

この情報を利用して、セル番号をキーにして、セルが使用するキャラクタデータの転送元オフセット値、VRAM 転送サイズが取得可能です。

3.2.4 NNSG2dCellData

NNSG2dCellData は描画に必要な OAM アトリビュート情報以外に、その他の付加情報を cellAttr に 持っています。

現在格納されている情報を以下に列挙します。具体的な格納位置は変更になる可能性がありますので、アクセサ関数を使用して情報にアクセスを行います。

- セルを構成する OBJ のフリップ機能使用状況に関する情報。
- セルの描画最適化に使用する情報。
- セルを内包する境界球の半径。
- セルが矩形境界情報をもっているか(NNSG2dCellDataWithBR にキャスト可能か)。

3.2.5 OAMアトリビュート情報

実際に描画に使用される OAM アトリビュート情報は、NNSG2dCellData データが格納された区間の次の区間に NNSG2dCellOAMAttrData という、アフィンパラメータ部分を省略したデータ形式で格納されています。

NNSG2dCellOAMAttrData の定義を以下に示します。

```
typedef struct NNSG2dCellOAMAttrData
{
    u16      attr0;
    u16      attr1;
    u16      attr2;

}NNSG2dCellOAMAttrData;
```

ファイルデータ上では、{attr0,attr1,attr2},{attr0,attr1,attr2},...というようにアトリビュート情報が格納されています。そのため、OAM アトリビュート情報列の終端で 4 バイトアラインメントのために、適当なパディング情報をデータに挿入する必要があります。

3.2.6 ユーザ拡張アトリビュート情報

-oua オプションを指定して出力することで、ユーザ拡張アトリビュート情報が出力されます。

本データが出力された場合は、NNSG2dCellDataBank.pExtendedData メンバに拡張データへのオフセットアドレスが記録されます。拡張データ内はブロックごとに分割して管理されています。データブロックにはヘッダが存在し、ブロックサイズと識別 ID を記録しています。セルユーザ拡張ブロックアトリビュートブロックの識別 ID は以下の通りです。

```
NNS_G2D_USEREXBLK_CELLATTR      (u32) 'UCAT'
```

4 NANR、NMAR(アニメーション定義情報)

本章では、NANR(NMAR)ファイルフォーマットについて説明を行います。

NANR(NMAR)は、アニメーション定義情報を格納するファイルフォーマットです。

アニメーションは、現在セルアニメーション、マルチセルアニメーションの2種類が存在します。

どちらも同一のデータフォーマットで構成されており NNSG2dAnimSequence の animType メンバの内容が異なるだけとなっています。(ランタイムソース中では可読性を高めるために、別名を定義しています。)

それぞれ、NITRO-CHARACTER バイナリファイル*.nce, *.nmc ファイルをコンバートし生成します。

アニメーション定義情報はおおまかには、下表のようなブロック構成となっています。

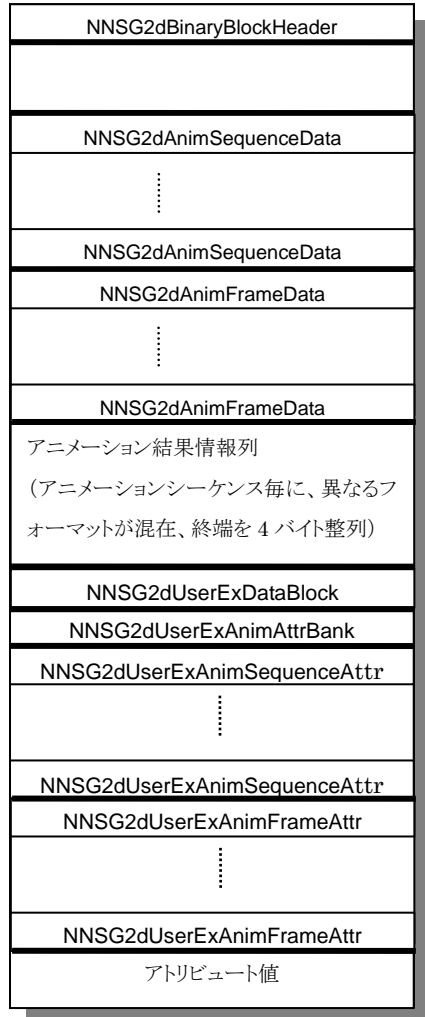
表 4-1 NANR NMAR ブロック構成

データブロック名	識別子	値	条件	備考
アニメーションバンクプロ	NNS_G2D_BLKSIG_ANIMBANK	'ABNK'	必須	固有
ユーザ拡張情報ブロック	NNS_G2D_BLKSIG_USEREXT	'UEXT'	任意	共通
名前ラベル情報	NNS_G2D_BLKSIG_NAMELABEL	'LABL'	任意	共通

アニメーション定義情報ファイル固有ブロックについて説明をしていきます。

4.1 アニメーションバンクブロック

アニメーションデータを定義するブロックです。本ブロックの大まかな構成は以下のようになっています。



a

図 4-1 NNSG2dCellBankBlock 構成

各データ構造の内容は以下のようになっています。

表 4-2 NNSG2dAnimBankDataBlock

型	パラメータ名	説明	バイト
NNSG2dBinaryBlockHeader	blockHeader	ブロックヘッダ	8
NNSG2dAnimBankData	cellDataBank	アニメーションフレーム待機フレーム数	24

表 4-3 NNSG2dAnimBankData

型	パラメータ名	説明	バイト
u16	numSequences	アニメーションシーケンス数	2
u16	numTotalFrames	アニメーションフレーム数	2
NNSG2dAnimSequenceData*	pSequenceArrayHead	アニメーションシーケンス情報配列への オフセット(ポインタ)	4
NNSG2dAnimFrameData*	pFrameArrayHead	アニメーションフレーム情報配列への オフセット(ポインタ)	4
void*	pAnimContents	アニメーション結果配列へのオフセット(ポインタ)	4
void*	pStringBank	文字列バンクへのポインタ(実行時に設定されます)	4
void*	pExtendedData	ライブラリ拡張情報へのポインタ	4

表 4-4 NNSG2dAnimSequenceData

型	パラメータ名	説明	バイト
u16	numFrames	アニメーションフレーム数	2
u16	loopStartFrameIdx	ループ再生スタートフレームインデックス	2
u32	animType	アニメーション結果種類	4
NNSG2dAnimationPlayMode	playMode	アニメーション再生方式	4
NNSG2dAnimFrameData*	pAnmFrameArray	アニメーションフレーム配列へのオフセット(ポインタ)	4

表 4-5 NNSG2dAnimFrameData

型	パラメータ名	説明	バイト
void*	pContent	アニメーション結果へのオフセット(ポインタ)	4
u16	frames	アニメーションフレーム待機フレーム数	2
u16	pad16	パディング	2

表 4-6 NNSG2dUserExAnimAttrBank

型	パラメータ名	説明	バイト
u16	numSequences	アニメーションシーケンス数	2
u16	numAttribute	アトリビュート数(現在は1固定)	2
NNSG2dUserExAnimSequenceAttr*	pAnmSeqAttrArray	シーケンスアトリビュート配列へのオフセット (ポインタ)	2

表 4-7 NNSG2dUserExAnimSequenceAttr

型	パラメータ名	説明	バイト
u16	numFrames	アニメーションフレーム数	2
u16	pad16	パディング	2
u32*	pAttr	アトリビュート値への配列	4
NNSG2dUserExAnimFrameAttr*	pAnmFrmAttrArray	フレームアトリビュート配列へのオフセット (ポインタ)	4

表 4-8 NNSG2dUserExAnimFrameAttr

型	パラメータ名	説明	バイト
u32*	pAttr	アトリビュート値への配列	4

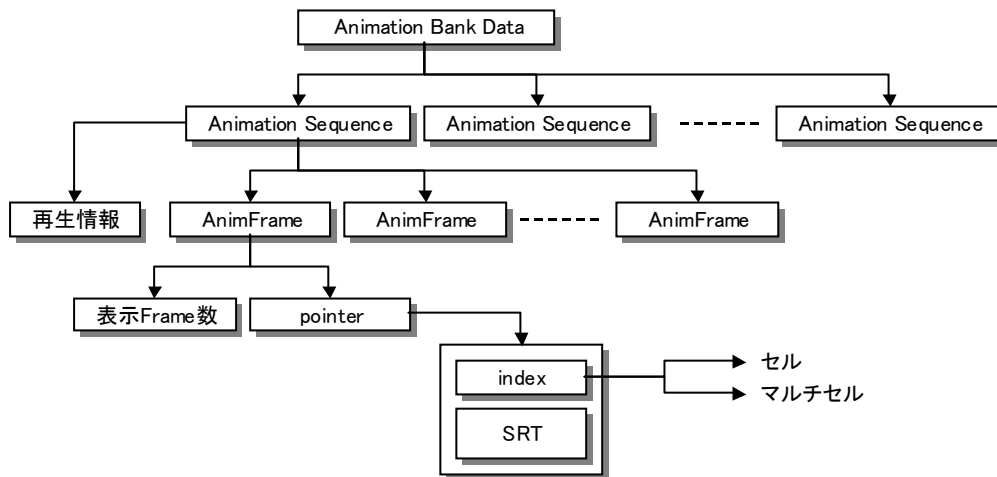


図 4-2 アニメーションデータ の大まかな構成

4.2 解説

4.2.1 NNSG2dAnimBankData.playMode

アニメーション再生方式に関する情報を格納します。

再生方法には、ワンタイム再生、リピート再生、往復再生、往復リピート再生の4種類の再生方法の内のいずれかが格納されます。

再生方法の列挙子である NNSG2dAnimationPlayMode の宣言は以下のとおりです。

```

typedef enum NNSG2dAnimationPlayMode
{
    NNS_G2D_ANIMATIONPLAYMODE_INVALID = 0x0, //無効
    NNS_G2D_ANIMATIONPLAYMODE_FORWARD, //ワンタイム再生
    NNS_G2D_ANIMATIONPLAYMODE_FORWARD_LOOP, //リピート再生
    NNS_G2D_ANIMATIONPLAYMODE_REVERSE, //往復再生
    NNS_G2D_ANIMATIONPLAYMODE_REVERSE_LOOP, //往復リピート再生
    NNS_G2D_ANIMATIONPLAYMODE_MAX
}
NNSG2dAnimationPlayMode;
  
```

4.2.2 NNSG2dAnimSequenceData.animType

NNSG2dAnimSequenceData.animType は32bit 長のデータです。

アニメーションの種類に関する情報を格納しています。

NNSG2dAnimSequenceData.animType の、下位 16bit はアニメーション結果の種類が格納されています。

アニメーション結果は複数のフォーマットを取ります。

たとえば、SRT アニメーションを使用しないシーケンスではアニメーション結果に index のみが格納されます。

これによってデータ量の削減などが実現可能となります。

アニメーション結果の型 NNSG2dAnimationElement の宣言は以下のとおりです。

```
typedef enum NNSG2dAnimationElement
{
    NNS_G2D_ANIMELEMENT_INDEX          = 0x0, // Index のみ
    NNS_G2D_ANIMELEMENT_INDEX_SRT      , // Index + SRT
    NNS_G2D_ANIMELEMENT_MAX
}
NNSG2dAnimationElement;
```

NNSG2dAnimSequenceData. animType の, 上位 16bit はアニメーションの種類が格納されています。

アニメーションの種類は、セルアニメーションかマルチセルアニメーションのどちらかが格納されます。

アニメーションの種類の列挙型 NNSG2dAnimationType の宣言は以下のとおりです。

```
typedef enum NNSG2dAnimationType
{
    NNS_G2D_ANIMATIONTYPE_INVALID      = 0x0, // 不正な種類
    NNS_G2D_ANIMATIONTYPE_CELL        , // Cell
    NNS_G2D_ANIMATIONTYPE_MULTICELLLOCATION , // MultiCell
    NNS_G2D_ANIMATIONTYPE_MAX
}
NNSG2dAnimationType;
```

(仕様変更の可能性もありますので、取得の際は専用のアクセサの利用を推奨します。)

4.2.3 アニメーション結果情報列

アニメーション結果情報列部分には、NNSG2dAnimFrameData.pContent が指し示す先の情報が格納されています。アニメーション結果のデータフォーマットは、複数のフォーマットが存在しています。アニメーションシーケンス毎に、アニメーションする情報に対応した最適なフォーマットを選択することでデータサイズの削減をおこなうことが可能です。また、アニメーション結果情報は終端をアニメーションシーケンスごとに、4 バイト整列してデータを格納しています。

具体的な、アニメーション結果のフォーマット定義は以下の通りです。

```
typedef          u16          NNSG2dAnimData; // index のみ

typedef struct NNSG2dAnimDataSRT          // index + SRT 情報
{
    u16          index;          // インデックス

    u16          rotZ;          // 回転

    fx32          sx;          // スケールX
    fx32          sy;          // スケールY

    s16          px;          // 位置X
    s16          py;          // 位置Y
}NNSG2dAnimDataSRT;
```

```
typedef struct NNSG2dAnimDataT      // index + T 情報
{
    u16      index;      // インデックス
    u16      pad_;      // 回転

    s16      px;         // 位置X
    s16      py;         // 位置Y
}NNSG2dAnimDataT;
```

4.2.4 ユーザ拡張アトリビュート情報

-oua オプションを指定して出力することで、ユーザ拡張アトリビュート情報が出力されます。

本データが出力された場合は、NNSG2dAnimBankData.pExtendedData メンバに拡張データへのオフセットアドレスが記録されます。拡張データ内はブロックごとに分割して管理されています。データブロックにはヘッダが存在し、ブロックサイズと識別 ID を記録しています。セルユーザ拡張ブロックアトリビュートブロックの識別 ID は以下の通りです。

```
NNS_G2D_USEREXBLK_ANMATTR      (u32)'UAAT'
```

5 NCGR,NCBR(キャラクタ情報)

本章では、NCGR, NCBR ファイルフォーマットについて説明を行います。

NCGR, NCBR はキャラクタ情報を格納する、ファイルフォーマットです。

NCGR は キャラクタ(8×8pixel) 毎にピクセルが整列して配置されています。

(内部メンバ characterFmt が NNS_G2D_CHARACTER_FMT_CHAR の時)

NCBR は、画素ラインごとにピクセルが整列して配置されています。

(内部メンバ characterFmt が NNS_G2D_CHARACTER_FMT_BMP の時)

どちらも、NITRO-CHARACTER バイナリファイル* .ncg ファイルをコンバートし生成します。

おおまかには、下表のようなブロック構成となっています。

表 5-1 NCGR (NCBR)ブロック構成

データブロック名	識別子	値	条件	備考
キャラクター定義ブロック	NNS_G2D_BINBLK_SIG_CHARACTERDATA	'CHAR'	必須	固有
キャラクター位置情報ブロック	NNS_G2D_BINBLK_SIG_CHAR_POSITION	'CPOS'	任意	固有
ユーザ拡張情報ブロック	NNS_G2D_BLKSIG_USEREXT	'UEXT'	任意	共通

NCGR ファイル固有ブロックについて説明をしていきます。

5.1 キャラクター定義ブロック

NNSG2dCharacterDataBlock の大まかな構成は下図のようになります。

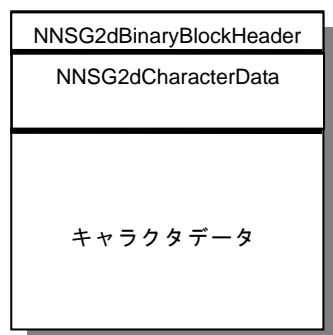


図 5-1 NNSG2dCharacterDataBlock 構成

各データ構造の内容は以下のようになっています。

表 5-2 NNSG2dCharacterDataBlock

型	パラメータ名	説明	バイト
NNSG2dBinaryBlockHeader	blockHeader	ブロックヘッダ	8
NNSG2dCharacterData	characterData	キャラクタデータ	24

表 5-3 NNSG2dCharacterData

型	パラメータ名	説明	バイト
u16	W	横方向キャラクタ数(2D マップ換算)	2
u16	H	縦方向キャラクタ数(2D マップ換算)	2
GXTexFmt	pixelFmt	ピクセルフォーマット	4
GXOBJVRamModeChar	mappingType	マッピング種類(1D or 2D)	4
u32	characterFmt	キャラクタ種類 (キャラクタ or ビットマップ、VRAM 転送キャラクタ)	4
u32	szByte	キャラクタデータのバイト数	4
void*	pRawData	キャラクタデータへのオフセットポインタ	4

5.2 キャラクター位置情報ブロック

NNSG2dCharacterPosInfoBlock の大まかな構成は下図のようになります。

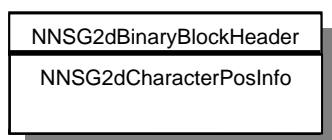


図 5-2 NNSG2dCharacterPosInfoBlock 構成

各データ構造の内容は以下のようになっています。

表 5-4 NNSG2dCharacterPosInfo

型	パラメータ名	説明	バイト
u16	srcPosX	抽出元キャラクタデータにおける X 位置(キャラクタ単位)	2
u16	srcPosY	抽出元キャラクタデータにおける Y 位置(キャラクタ単位)	2
u16	srcW	抽出元キャラクタデータの幅(キャラクタ単位)	2
u16	srcH	抽出元キャラクタデータの高さ(キャラクタ単位)	2

5.3 解説

5.3.1 NNSG2dCharacterData.W .H

NNSG2dCharacterData には、キャラクタデータの持つフォーマットに関する情報が格納されています。

W、H にはキャラクタデータのキャラクタ単位の大きさが格納されています。ただし、この情報はキャラクタが2D マッピングモードで格納されている時にのみ有効な情報です。1D マッピングモードで格納されたキャラクタデータの場合には、必ず本データには、キャラクタサイズとしては利用できない、NNS_G2D_1D_MAPPING_CHAR_SIZE が記録される点にご注意ください。

5.3.2 NNSG2dCharacterData.pixelFmt

pixelFmt はキャラクターイメージの画素フォーマットです。

G2D で有効なのは GX_TEXFMT_PLTT16、GX_TEXFMT_PLTT256 だけとなっています。

```
typedef enum
{
    GX_TEXFMT_NONE      = 0,
    GX_TEXFMT_A3I5      = 1,
    GX_TEXFMT_PLTT4      = 2,
    GX_TEXFMT_PLTT16     = 3,
    GX_TEXFMT_PLTT256    = 4,
    GX_TEXFMT_COMP4x4    = 5,
    GX_TEXFMT_A5I3       = 6,
    GX_TEXFMT_DIRECT     = 7
}
GXTexFmt;
```

5.3.3 NNSG2dCharacterData.characterFmt

characterFmt はキャラクタデータのピクセルデータの並び方種類です。上位8bit を NNSG2dCharacterFmt の格納に利用し、9bit 目を VRAM 転送用キャラクタかどうかのフラグとして利用しています。

NNSG2dCharacterFmt の値は 3D グラフィックスエンジンを使用して描画するデータでは、NNS_G2D_CHARACTER_FMT_BMP となり、2D グラフィックスエンジンを使用して描画するデータでは、NNS_G2D_CHARACTER_FMT_CHAR となります。

使用されていないビットは今後の拡張のために予約されています。

characterFmt に格納される、列挙子の詳細は以下の通りです。

```
typedef enum NNSG2dCharacterFmt
{
    NNS_G2D_CHARACTER_FMT_CHAR,
    NNS_G2D_CHARACTER_FMT_BMP,
    NNS_G2D_CHARACTER_FMT_MAX
}NNSG2dCharacterFmt;
```

5.3.4 NNSG2dCharacterData.mapingType

mapingType はキャラクタデータ格納の種類です。大きく 1D モードと2D モードに分類されます。

詳細は TWL プログラミングマニュアルを参照ください。

列挙子の定義は以下のとおりです。

```
typedef enum
{
    GX_OBJVRAMMODE_CHAR_2D      ,
    GX_OBJVRAMMODE_CHAR_1D_32K ,
    GX_OBJVRAMMODE_CHAR_1D_64K ,
    GX_OBJVRAMMODE_CHAR_1D_128K,
    GX_OBJVRAMMODE_CHAR_1D_256K
}
GXOBJVRamModeChar;
```

5.3.5 NNSG2dCharacterPosInfoBlock

キャラクタ位置情報ブロックは、2D マッピングモードのキャラクタデータで、バイナリコンバート時に **-cr** オプションが指定された場合にのみ出力されます。

6 NCLR (カラーパレット情報)

本章では、NCLR ファイルフォーマットについて説明を行います。

NCLR はパレット情報を格納する、ファイルフォーマットです。

NITRO-CHARACTER バイナリファイル* .ncl ファイルをコンバートし生成します。

NCLR は、おおまかには、下表のようなブロック構成となっています。

表 6-1 NCLR ブロック構成

データブロック名	識別子	値	条件	備考
パレット定義ブロック	NNS_G2D_BINBLK_SIG_PALETTE DATA	'PLTT'	必須	固有
パレット圧縮情報ブロック	NNS_G2D_BINBLK_SIG_PALETTE COMP INFO	'PCMP'	圧縮時	固有
ユーザ拡張情報ブロック	NNS_G2D_BLK SIG_USER EXT	'UEXT'	任意	共通

NCLR ファイル固有ブロックについて説明をしていきます。

6.1 パレット定義ブロック

NNSG2dPaletteDataBlock の大まかな構成は下図のようになります。

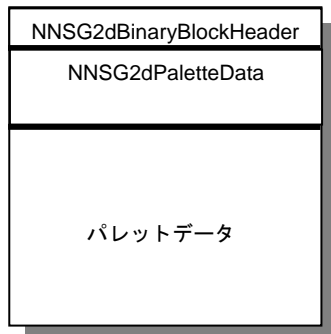


図 6-1 NNSG2dPaletteDataBlock 構成

各データ構造の内容は以下のようになっています。

表 6-2 NNSG2dPaletteDataBlock

型	パラメータ名	説明	バイト
NNSG2dBinaryBlockHeader	blockHeader	ブロックヘッダ	8
NNSG2dPaletteData	paletteData	パレットデータ	16

表 6-3 NNSG2dPaletteData

型	パラメータ名	説明	バイト
GXTexFmt	fmt	パレットフォーマット	4
BOOL	bExtendedPlt	拡張パレットを使用するか	4
u32	szByte	キャラクタデータのバイト数	4
void*	pRawData	パレットデータへのオフセットポインタ	4

6.2 パレット圧縮情報ブロック

NNSG2dPaletteCompressDataBlock の大まかな構成は下図のようになります。



図 6-2 NNSG2dPaletteCompressDataBlock 構成

各データ構造の内容は以下のようになっています。

表 6-4 NNSG2dPaletteCompressDataBlock

型	パラメータ名	説明	バイト
NNSG2dBinaryBlockHeader	blockHeader	ブロックヘッダ	8
NNSG2dPaletteCompressInfo	plttCmpInfo	パレット圧縮情報	8

表 6-5 NNSG2dPaletteCompressInfo

型	パラメータ名	説明	バイト
u16	numPalette	パレット数	2
u16	pad16	パディング	2
void*	pPlttIdxTbl	圧縮前のパレット番号配列	4

6.3 説明

6.3.1 NNSG2dPaletteCompressInfo

圧縮パレットは、使用されているパレット番号のデータのみを記憶しています。データをもとに復元するために本データは使用されます。NNSG2dPaletteCompressInfo. pPlttIdxTbl の先は配列長が圧縮後のパレット数である u16 パレット番号配列になっており、その配列をパレット番号変換テーブルとして、元のパレット番号を取り出します。

7 NMCR (マルチセル情報)

本章では、NMCR ファイルフォーマットについて説明を行います。

NMCR はマルチセル定義情報を格納する、ファイルフォーマットです。

NITRO-CHARACTER バイナリファイル* .nmc, ファイルをコンバートし生成します。

(現バージョンのコンバータでは未サポートです)

おおまかには、下表のようなブロック構成となっています。

表 7-1 NMCR ブロック構成

データブロック名	識別子	値	条件	備考
マルチセルバンクブロック	NNS_G2D_BLKSIG_MULTICELLBANK	'MCBK'	必須	固有
ユーザ拡張情報ブロック	NNS_G2D_BLKSIG_USEREXT	'UEXT'	任意	共通
名前ラベル情報	NNS_G2D_BLKSIG_NAMELABEL	'LABL'	任意	共通

NMCR ファイル固有ブロックについて説明をしていきます。

7.1 マルチセルバンクブロック

NNSG2dMultiCellDataBankBlock の大まかな構成は下図のようになります。

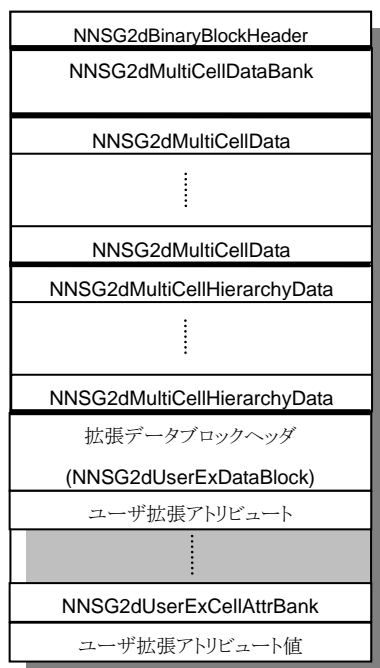


図 7-1 NNSG2dMultiCellDataBankBlock 構成

各データ構造の内容は以下のようになっています。

表 7-2 NNSG2dMultiCellDataBankBlock 構成

型	パラメータ名	説明	バイト
NNSG2dBinaryBlockHeader	blockHeader	ブロックヘッダ	2
NNSG2dMultiCellDataBank	multiCellDataBank	マルチセルデータバンク	20

表 7-3 NNSG2dMultiCellDataBank

型	パラメータ名	説明	バイト
u16	numMultiCellData	マルチセルデータ数	2
u16	pad16	パディング	2
NNSG2dMultiCellData*	pMultiCellDataArray	マルチセルデータへの オフセットポインタ	4
NNSG2dMultiCellHierarchyData*	pHierarchyDataArray	セルアニメーション要素の設定情報配列 へのオフセットポインタ	4
void*	pStringBank	文字列バンクへのポインタ (実行時に設定されます)	4
void*	pExtendedData	ライブラリ拡張情報へのポインタ (未使用)	4

表 7-4 NNSG2dMultiCellData

型	パラメータ名	説明	バイト
u16	numNodes	マルチセルを構成する セルアニメーションノード数	2
u16	numCellAnim	最低限必要なセルアニメ実体の数	2
NNSG2dMultiCellHierarchyData*	pHierDataArray	セルアニメーション要素の設定情報配 列へのオフセットポインタ	4

表 7-5 NNSG2dMultiCellHierarchyData

型	パラメータ名	説明	バイト
u16	animSequenceIdx	再生するアニメーション番号	2
s16	posX	位置 X	2
s16	posY	位置 Y	2
u16	nodeAttr	ノード属性 (NNSG2dMCAnimationPlayMode など)	2

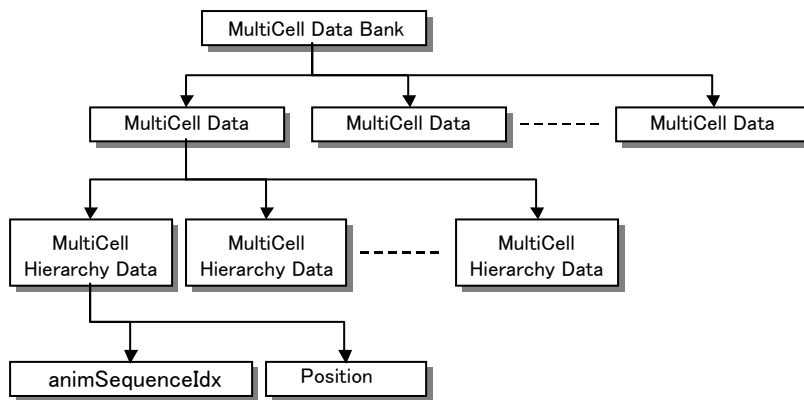


図 7-2 マルチセルデータのおおまかな構成

7.2 解説

7.2.1 NNSG2dMultiCellHierarchyData. nodeAttr

nodeAttr には、ノードにバインドされるセルアニメーションのアニメ再生方式や、可視フラグ情報が格納されています。

下位4bit には NNSG2dMCAnimationPlayMode が格納されています。

5bit 目には可視フラグが格納されています。

16-8bit は、最低限必要なセルアニメーションのセルアニメーション番号を格納しています。

たとえば、ノードが 10 個存在し、それらがすべて同一のセルアニメーションを再生するようなデータの場合には、最低限必要なセルアニメーション数は1となり、すべての nodeAttr にセルアニメーション番号 0 が格納されます。

この情報は、現在のランタイムライブラリでは使用されません。

残りの未使用ビットは今後の機能拡張のために予約されています。

NNSG2dMCAnimationPlayMode の定義を以下に示します

```

typedef enum NNSG2dMCAnimationPlayMode
{
    NNS_G2D_MCANIM_PLAYMODE_RESET = 0,
    NNS_G2D_MCANIM_PLAYMODE_CONTINUE = 1,
    NNS_G2D_MCANIM_PLAYMODE_PAUSE = 2,
    NNS_G2D_MCANIM_PLAYMODE_MAX

}NNSG2dMCAnimationPlayMode;
  
```

7.2.2 ユーザ拡張アトリビュート情報

マルチセルバンクのユーザ拡張アトリビュート情報については、セルバンクのユーザ拡張アトリビュート情報とまったく同一のデータフォーマットを使用して、格納されています。そのため、ここでの説明は割愛します。

8 NENR(エンティティ情報)

本章では、NENR ファイルフォーマットについて説明を行います。

NENR は、エンティティ定義情報を格納するファイルフォーマットです。

よりユーザプログラム寄りではあるが、ライブラリでサポートする必要があるようなデータはこの構造に格納される予定になっています。(現状では再生アニメーション番号情報のみを持っています。)

NENR は、Win32 アプリケーション BuildNENR.exe を使用して、エンティティ定義テキストをコンバートすることによって生成します。

NENR は、おおまかには、下表のようなブロック構成となっています。

表 8-1 NENR ブロック構成

データブロック名	識別子	値	条件	備考
エンティティバンクブロック	NNS_G2D_BLKSIG_ENTITYBANK	'ENBK'	必須	固有
ユーザ拡張情報ブロック	NNS_G2D_BLKSIG_USEREXT	'UEXT'	任意	共通
名前ラベル情報	NNS_G2D_BLKSIG_NAMELABEL	'LABL'	任意	共通

NENR ファイル固有ブロックについて説明をしていきます。

8.1 エンティティバンクブロック

NNSG2dEntityDataBankBlock の大まかな構成は下図のようになります。

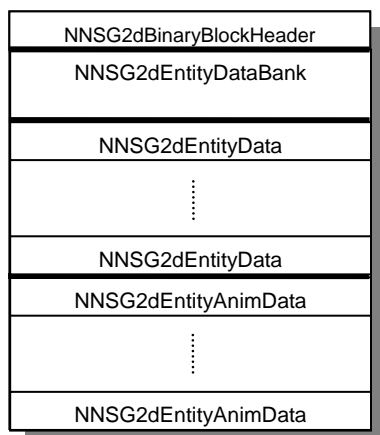


図 8-1 NNSG2dEntityDataBankBlock 構成

各データ構造の内容は以下のようになっています。

表 8-2 NNSG2dEntityDataBankBlock 構成

型	パラメータ名	説明	バイト
NNSG2dBinaryBlockHeader	blockHeader	ブロックヘッダ	2
NNSG2dEntityDataBank	entityDataBank	エンティティデータバンク	20

表 8-3 NNSG2dEntityDataBank

型	パラメータ名	説明	バイト
u16	numEntityDatas	エンティティデータ数	2
u16	pad16	パディング	2
NNSG2dEntityData*	pEntityDataArray	エンティティデータ配列へのオフセットポインタ	4
u16*	pAnimSequenceIdxArray	アニメーションシーケンス番号配列へのオフセットポインタ	4
void*	pStringBank	文字列バンクへのポインタ (実行時に設定されます)	4
void*	pExtendedData	ライブラリ拡張情報へのポインタ(未使用)	4

表 8-4 NNSG2dEntityData

型	パラメータ名	説明	バイト
NNSG2dEntityAnimData	animData	アニメーションデータ	8
NNSG2dEntityType	type	エンティティの種類(セル or マルチセル)	4

表 8-5 NNSG2dEntityAnimData

型	パラメータ名	説明	バイト
u16	numAnimSequence	アニメーションシーケンス番号数	2
u16	pad16	パディング	2
u16*	pAnimSequenceIdxArray	アニメーションシーケンス番号配列へのオフセットポインタ	4

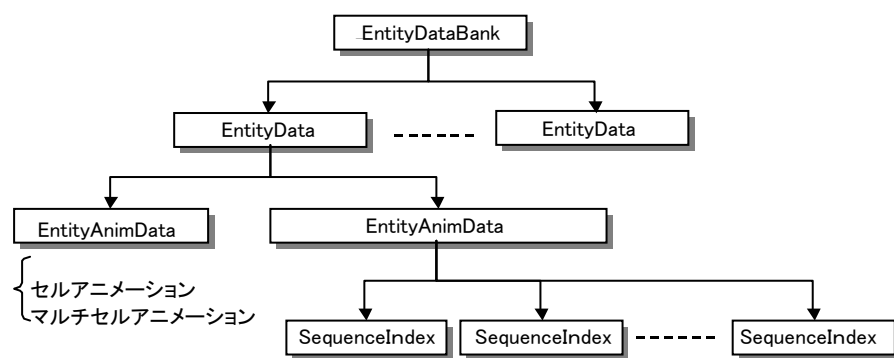


図 8-2 Entity データの大まかな構成

9 NSCR(スクリーン情報)

本章では、NSCR ファイルフォーマットについて説明を行います。

NSCR はスクリーン情報を格納する、ファイルフォーマットです。

NITRO-CHARACTER バイナリファイル *.nsc ファイルをコンバートし生成します。

おおまかには、下表のようなブロック構成となっています。

表 9-1 NSCR ブロック構成

データブロック名	識別子	値	条件	備考
スクリーン定義ブロック	NNS_G2D_BINBLK_SIG_SCRDATA	'SCRN'	必須	固有
ユーザ拡張情報ブロック	NNS_G2D_BLKSIG_USEREXT	'UEXT'	任意	共通

NSCR ファイル固有ブロックについて説明をしていきます。

9.1 スクリーン定義ブロック

NNSG2dScreenDataBlock の大まかな構成は下図のようになります。

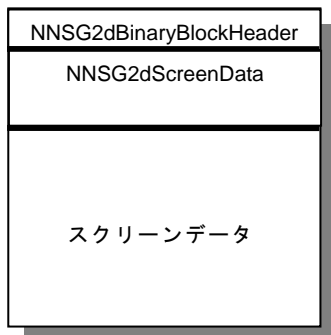


図 9-1 NNSG2dScreenDataBlock 構成

各データ構造の内容は以下のようになっています。

表 9-2 NNSG2dScreenDataBlock

型	パラメータ名	説明	バイト
NNSG2dBinaryBlockHeader	blockHeader	ブロックヘッダ	8
NNSG2dScreenData	screenData	スクリーンデータ	12

表 9-3 NNSG2dScreenData

型	パラメータ名	説明	バイト
u16	screenWidth	ピクセル単位のスクリーン幅	2
u16	screenHeight	ピクセル単位のスクリーン高さ	2
u16	colorMode	対応するパレットフォーマット	2
u16	screenFormat	BG 種類	2
u32	szByte	スクリーンデータのバイト数	4
u8[]	rawData	スクリーンデータ	szByte

9.2 解説

9.2.1 NNSG2dScreenData. ColorMode

colorMode はスクリーンデータの表示に用いるカラーパレットの種類です。

colorMode に格納される列挙子の詳細は以下のとおりです。

```
typedef enum NNSG2dColorMode
{
    NNS_G2D_SCREENCOLORMODE_16x16,
    NNS_G2D_SCREENCOLORMODE_256x1,
    NNS_G2D_SCREENCOLORMODE_256x16
}
NNSG2dColorMode1;
```

9.2.2 NNSG2dScreenData. screenFormat

screenFormat は格納しているスクリーンデータの BG タイプです。

screenFormat に格納される、列挙子の詳細は以下の通りです。

```
typedef enum NNSG2dScreenFormat
{
    NNS_G2D_SCREENFORMAT_TEXT,
    NNS_G2D_SCREENFORMAT_AFFINE,
    NNS_G2D_SCREENFORMAT_AFFINEEXT
}
NNSG2dScreenType;
```

10 NFTR(フォント情報)

本章では、NFTR ファイルフォーマットについて説明を行います。

NFTR はフォント情報を格納するファイルフォーマットです。

BMP ファイルもしくは Windows フォントを fontcvtr.exe でコンバートし生成します。

おおまかには、下表のようなブロック構成となっています。

表 10-1 NFTR ブロック構成

データブロック名	識別子	値	条件	備考
フォント情報ブロック	NNS_G2D_BINBLK_SIG_FINFDATA	'FINF'	必須	固有
グリフイメージブロック	NNS_G2D_BINBLK_SIG_CGLPDATA	'CGLP'	必須	固有
文字幅ブロック	NNS_G2D_BINBLK_SIG_CWDHDATA	'CWDH'	任意	固有
文字コードマップブロック	NNS_G2D_BINBLK_SIG_CMAPDATA	'CMAP'	任意	固有
ユーザ拡張情報ブロック	NNS_G2D_BLKSIG_USEREXT	'UEXT'	任意	共通

フォント情報ブロックとグリフイメージブロックは必ず 1 つだけ存在する必要がありますが、その他のブロックは存在しない場合も含めて数に制限はありません。

以下では NFTR ファイル固有ブロックについて説明をしていきます。

10.1 フォント情報ブロック

フォント情報ブロック(NNSG2dFontInformationBlock)はフォント全体にわたる様々な情報を保持しています。また実行時にオフセットポインタを解決する事でフォント情報ブロックを基点として他のブロックにポインタアクセスをすることができます。

NNSG2dFontInformationBlock の大まかな構成は下図のようになります。

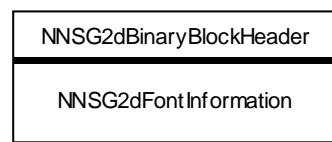


図 10-1 NNSG2dFontInformationBlock 構成

各データ構造の内容は以下のようになっています。

表 10-2 NNSG2dFontInformationBlock

型	パラメータ名	説明	バイト
NNSG2dBinaryBlockHeader	blockHeader	ブロックヘッダ	8
NNSG2dFontInformation	blockBody	フォント情報	20

表 10-3 NNSG2dFontInformation

型	パラメータ名	説明	バイト
u8	fontType	フォントの種類	1
u8	linefeed	フォントの改行幅	1
u16	alterCharIndex	代替文字のグリフインデックス	2
NNSG2dCharWidths	defaultWidth	デフォルトの文字幅情報	3
u8	encoding	対応する文字列エンコーディング	1
NNSG2dFontGlyph*	pGlyph	グリフイメージへのオフセットポインタ	4
NNSG2dFontWidth*	pWidth	文字幅データへのオフセットポインタ	4
NNSG2dFontCodeMap*	pMap	文字コードマップデータへのオフセットポインタ	4
u8	height	フォントの高さ	1
u8	width	フォント幅 (現在のライブラリでは使用されません)	1
u8	ascent	アセント	1
u8	padding_	パディング領域	1

表 10-4 NNSG2dCharWidths

型	パラメータ名	説明	バイト
s8	left	文字の左スペース幅	1
u8	glyphWidth	文字のグリフ幅	1
s8	charWidth	文字幅	1

10.2 グリフイメージブロック

グリフイメージブロック(NNSG2dFontGlyphBlock)はフォントの中心である各文字のグリフイメージとグリフイメージに関する情報を格納しています。

NNSG2dFontGlyphBlock の大まかな構成は下図のようになります。

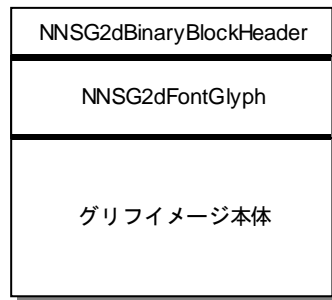


図 10-2 NNSG2dFontGlyphBlock 構成

各データ構造の内容は以下のようになっています。

表 10-5 NNSG2dFontGlyphBlock

型	パラメータ名	説明	バイト
NNSG2dBinaryBlockHeader	blockHeader	ブロックヘッダ	8
NNSG2dFontGlyph	blockBody	グリフデータ	8

表 10-6 NNSG2dFontGlyph

型	パラメータ名	説明	バイト
u8	cellWidth	セルの幅	1
u8	cellHeight	セルの高さ	1
u16	cellSize	セルのデータサイズ	2
s8	baselinePos	セル上端からのベースライン位置	1
u8	maxCharWidth	最大文字幅	1
u8	bpp	セル 1 ピクセルあたりの bit 数	1
u8	flags	フォントの特徴フラグ	1
u8[]	glyphTable	グリフデータ (セルの配列)	可変

10.3 文字幅ブロック

文字幅ブロック(NNSG2dFontWidthBlock)はグリフイメージブロックに格納されているグリフイメージの文字幅を格納しています。文字幅ブロックが 1 つのフォントリソースに複数含まれる場合、これらはリンクリストを構成しています。文字幅ブロックに対応する文字幅が格納されていないグリフには、フォント情報ブロックで定義されるデフォルトの文字幅が適用されます。

NNSG2dFontWidthBlock の大まかな構成は下図のようになります。

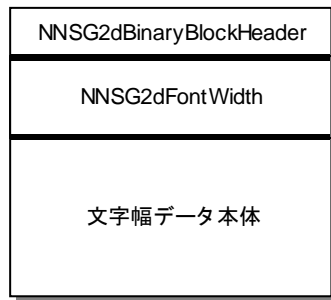


図 10-3 NNSG2dFontWidthBlock 構成

各データ構造の内容は以下のようになっています。

表 10-7 NNSG2dFontWidthBlock

型	パラメータ名	説明	バイト
NNSG2dBinaryBlockHeader	blockHeader	ブロックヘッダ	8
NNSG2dFontWidth	blockBody	文字幅データ	8

表 10-8 NNSG2dFontWidth

型	パラメータ名	説明	バイト
u16	indexBegin	最初のエントリに対応するグリフインデックス	2
u16	indexEnd	最後のエントリに対応するグリフインデックス	2
NNSG2dFontWidth*	pNext	次の NNSG2dFontWidth へのオフセットポインタ	4
NNSG2dCharWidths[]	widthTable	文字幅情報の配列	可変

10.4 文字コードマップブロック

文字コードマップブロック(NNSG2dFontCodeMapBlock)は文字コードとグリフイメージの対応付け定義を格納しています。一般に 1 つのフォントリソースには複数の文字コードマップブロックが含まれ、それらがリンクリストを構成しています。文字コードから対応するグリフインデックスを取得する場合には文字情報ブロックからリンクリストを順にたどり、最初の変換可能なブロックを用いて変換を行います。これはリンクリストの後ろに位置するブロックがより前に位置するブロックの対応する範囲を含むことを可能にし、ブロックが細切れになってしまうのを防ぎます。

NNSG2dFontCodeMapBlock の大まかな構成は下図のようになります。

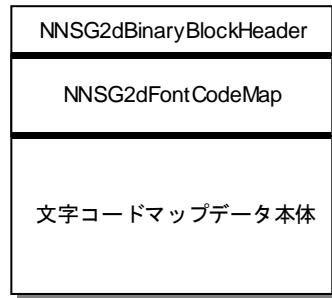


図 10-4 NNSG2dFontCodeMapBlock 構成

各データ構造の内容は以下のようになっています。

表 10-9 NNSG2dFontCodeMapBlock

型	パラメータ名	説明	バイト
NNSG2dBinaryBlockHeader	blockHeader	ブロックヘッダ	8
NNSG2dFontCodeMap	blockBody	文字コードマップデータ	12

表 10-10 NNSG2dFontCodeMap

型	パラメータ名	説明	バイト
u16	ccodeBegin	このブロックにより変換可能な文字コード始端	2
u16	ccodeEnd	このブロックにより変換可能な文字コード終端	2
u16	mappingMethod	文字コードマップ種別	2
u16	reserved	(予約)	2
NNSG2dFontCodeMap*	pNext	次の NNSG2dFontCodeMap のオフセット	4
u16	mapInfo	文字コードとグリフインデックスの対応表	可変

表 10-11 NNSG2dCMapInfoScan

型	パラメータ名	説明	バイト
u16	num	entries の要素数	2
NNSG2dCMapScanEntry[]	entries	文字コードとグリフインデックスの対応表	可変

表 10-12 NNSG2dCMapScanEntry

型	パラメータ名	説明	バイト
u16	ccode	文字コード	2
u16	index	グリフインデックス	2

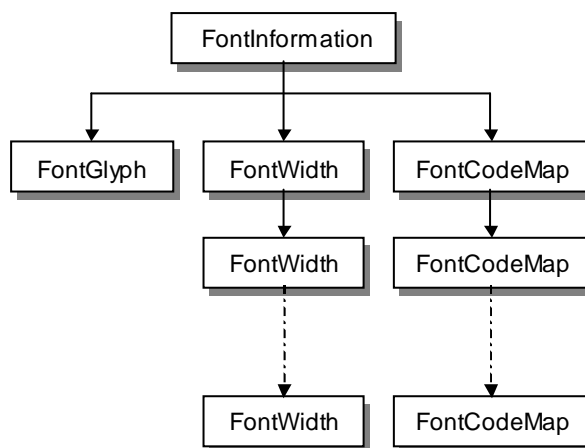


図 10-5 フォントデータの大まかな構成

10.5 解説

10.5.1 NNSG2dFontInformation.fontType

フォントリソースの持つグリフ情報の種類が格納されています。現在は `NNS_G2D_FONTTYPE_GLYPH` のみが定義されています。

10.5.2 NNSG2dFontInformation.pGlyph / pWidth / pMap

これらは各ブロック種類の最初のブロックへのオフセットポインタとなっています。それぞれのブロックは次のブロックへのオフセットポインタをもっており、単方向リンクリストを形成しています。対応するブロックが存在しない場合や、リストの終端には `NULL` が格納されています。

実行時に各ブロックのオフセットポインタを解決する事で、フォント情報ブロックからフォントリソース内の全てのブロックにポインタ経由でアクセスができるようになります。

10.5.3 NNSG2dFontGlyph.flags

フォントの特徴を表すフラグの集合です。縦書き/縦持ち用のフォントを示すフラグが格納されています。

10.5.4 NNSG2dFontGlyph.glyphTable

グリフイメージが格納されたビットマップデータの配列です。各ビットマップデータは `cellSize` バイトであり、全てのビットマップデータはアライメントされることなく連続して配置されています。なお、ビットマップデータは文字コード昇順に配置されています。

10.5.5 NNSG2dFontWidth.indexBegin / indexEnd / widthTable

`indexBegin` には `widthTable` の最初の要素に対応するグリフインデックスが、`indexEnd` には最後の要素に対応するグリフインデックスが格納されています。逆にいえばグリフインデックスが `indexBegin` であるグリフの文字幅情報が

widthTable の最初の要素として格納されています。

widthTable には indexBegin から indexEnd までの全てのグリフインデックスの文字幅情報が順に格納されていますので[グリフインデックス - indexBegin]を配列の添字として使う事で目的のグリフインデックスの文字幅情報を取得する事ができます。なお widthTable に格納されている文字幅情報の要素数は[indexEnd - indexBegin + 1]個となります。

10.5.6 NNSG2dFontCodeMap.ccodeBegin / ccodeEnd

ccodeBegin と ccodeEnd はそれぞれ当該ブロックが対象とする文字コードの最初と最後を表しています。これは ccodeBegin から ccodeEnd までの全ての文字コードに対応するグリフがフォントリソース内に格納されている事を示すものではありません。

10.5.7 NNSG2dFontCodeMap.mppingMethod / mapInfo

文字コードからどのようにしてグリフインデックスへ変換するかの方法種別が指定されます。変換方法によって mapInfo に格納されている情報は変化します。以下の 3 つの変換方法が定義されています。

- **NNS_G2D_MAPMETHOD_DIRECT**

文字コードから下に示す計算によってグリフインデックスを求めます。mapInfo には u16 型のグリフインデックスオフセットが 1 つ格納されています。

ccodeBegin から ccodeEnd までの全ての文字コードに対応するグリフが格納されています。

$$\text{グリフインデックス} = \text{文字コード} - \text{ccodeBegin} + \text{グリフインデックスオフセット}$$

- **NNS_G2D_MAPMETHOD_TABLE**

文字コードをキーとしてテーブル引きによりグリフインデックスを求めます。mapInfo には u16 型でグリフインデックスが[ccodeEnd - ccodeBegin + 1]個格納されています。取得できたグリフインデックスが 0xFFFF である場合、対応するグリフが格納されていない事を示します。

$$\text{グリフインデックス} = \text{mapInfo}[\text{文字コード} - \text{ccodeBegin}]$$

- **NNS_G2D_MAPMETHOD_SCAN**

文字コードをキーとして文字コードとグリフインデックスのペアの配列から検索します。この場合 mapInfo には NNSG2dCMapInfoScan 型のデータが格納されています。NNSG2dCMapInfoScan.entries には要素が文字コード昇順で格納されているため二分探索を利用できます。文字コードが見つからない場合は対応するグリフが格納されていない事を示します。

Microsoft Windows は米国 Microsoft Corporation 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。

© 2004-2009 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。