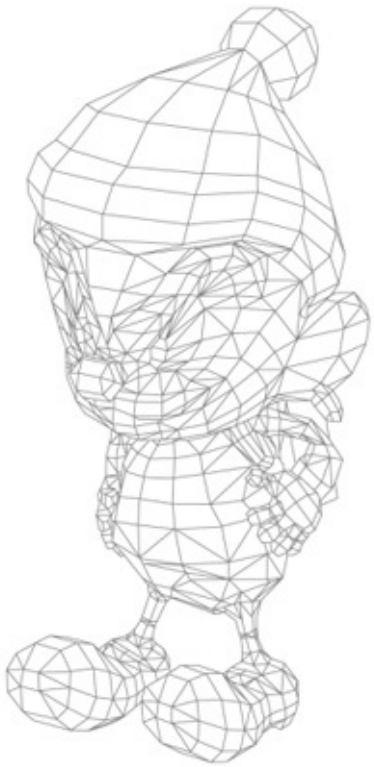
MASCOT CAPSULE®
REAL-TIME 3DRENDERING ENGINE

BAC File Ver.6.0
Data Format Specification

MascotCapsuleV3

日本語版



Ver.2.1

HI CORP.

－更新履歴－

2003 年 10 月 8 日
・Ver.1.0　新規作成

2005 年 6 月 15 日
・Ver.1.0

更新内容
- 文章の一部およびレイアウトの修正

2007 年 3 月 20 日
・Ver.2.0

更新内容
- 文章およびレイアウトの修正
- サンプルコードを追加

2007 年 5 月 28 日
・Ver.2.1

更新内容
- 文章およびレイアウトの修正
- ボーン座標を定義するチャンクの説明を一部改変しました。

― 目 次 ―

1. はじめに

本稿は MascotCapsuleV2/V3（以下、V2/V3 と記す）の中間ファイルである BAC ファイル（以下、BAC と記す）について解説しています。

BAC ファイルとは、V2/V3 で利用するモデル情報を持つデータファイルのことで、バージョン 5.0/6.0（以下、BAC5/BAC6 と記す）があります。BAC5 は、V2 で利用するモデル情報を持つデータファイルです。それに対して BAC6 は、V3 で利用するモデル情報を持つデータファイルになります。

本稿では、この BAC6 のデータフォーマットについて詳細を説明します。

1.1. BAC6 とは

BAC6 とはテキストフォーマット形式（BAC5 はバイナリフォーマット形式）のファイルで、V3 で利用するモデルデータに適用されているマテリアルの属性や、ボーン情報などを文字列で記述していきます。

アニメーション情報に関しては TRA4 ファイルに定義します。別紙「TRA File Ver.4.0 Data Format Specification」を参照してください。BAC6 と TRA4 は V3 で利用する対となる中間ファイルです。

BAC6 のファイル拡張子は「.bac」です。テキスト形式で扱う文字規約は「ASCII コード」の範囲でサポートしています。

2. マクロについて

本稿では BAC6 の説明及び記述の簡略化のためにマクロ表現を行うことがあります。そのため、以下では各マクロについて説明します。

2.1. マクロ詳細

本稿を参照する上で BAC6 のパラメータ部を以下のマクロで定義しています。

2.1.1. [STRING]

[STRING]マクロはヌル終端処理された文字列を意味します。
文字列はダブルクォーテーション(“”)で囲い、255 文字(byte)以内に収めてください。

2.1.2. [A|B]

[A|B]マクロは A あるいは B を意味します。
A 又は B はヌル終端文字列で 255 文字(byte)以内に収めてください。

2.1.3. [INT]

[INT]マクロは整数値を表す文字列を意味します。
文字列(数値)の範囲は int 型のビット範囲に収めてください。

2.1.4. [FLOAT]

[FLOAT]マクロは浮動小数値を表す文字列を意味します。
文字列(数値)の範囲は float 型のビット範囲に収めてください。

2.1.5. ...

…マクロは任意の繰り返しを意味します。

2.1.6. ;（セミコロン）

；マクロはセミコロンから行末までがコメント文字列であること意味します。

3. ファイル構造

ここでは **BAC6** の基本となる構造と規約について説明します。

3.1. BAC6 ファイル構造

BAC6 は、ファイル識別情報としてファイルの先頭に定義する「;BAC」と、（Head）チャンクと（Figure）チャンクの２つのチャンクから構成されています。ヘッダーを意味する（Head）チャンクを定義した後に、モデル情報を定義する（Figure）チャンクも定義します。これら定義の順序は変更できません。

（Head）と（Figure）チャンクはそれぞれに属する子チャンクを持ち、その子チャンクはさらに子チャンクを複数持っています。

これらを踏まえた上で図 1「BAC6 ファイル構造」を参照してください。

図 1「BAC6 ファイル構造」

```
;BAC

( Head
        ( bacVersion [FLOAT] )
)
( Figure
        ( name [STRING] )

        ( Textures
                ( i2 [INT] [INT] )
                …
        )

        ( Colors
                ( f3 [FLOAT] [FLOAT] [FLOAT] )
                …
        )

        ( Materials
                ( material
                        ( blendMode       [normal | add | sub | half] )
                        ( doubleFace      [true | false] )
                        ( transparent     [true | false] )
                        ( lighting        [true | false] )
                        ( textureIndex    [INT] )
                        ( colorIndex      [INT] )
                        ( specular        [FLOAT] )
                        ( alpha           [FLOAT] )
                        ( shininess       [FLOAT] )
                )
                …
        )

        ( Vertices
                ( coords
                        ( pnt [FLOAT] [FLOAT] [FLOAT] )
                        …
                )
                ( normals
                        ( vct [FLOAT] [FLOAT] [FLOAT] )
                        …
                )
        )
```

```
    ( Bones
          ( bone
                        ( name           [STRING] )
                        ( hasChild       [true | false] )
                        ( hasBrother     [true | false] )
                        ( translate      [FLOAT] [FLOAT] [FLOAT] )
                        ( rotate         [FLOAT] [FLOAT] [FLOAT] )
                        ( handle         [FLOAT] [FLOAT] [FLOAT] )
                        ( vertexIndices
                                [INT] [INT] …
                        )
           )
           …
    )

    ( TextureCoords
           ( f2 [FLOAT] [FLOAT] )
           …
    )

    ( Polygons
           ( face [INT] ( i3 [INT] [INT] [INT] )( i3 [INT] [INT] [INT] ))
           ( face [INT] ( i4 [INT] [INT] [INT] [INT] )( i4 [INT] [INT] [INT] [INT] ))
           …
    )
    ( DynamicPolygons
        ( group
           ( name [STRING] )
           ( face [INT] ( i3 [INT] [INT] [INT] )( i3 [INT] [INT] [INT] ))
           ( face [INT] ( i4 [INT] [INT] [INT] [INT] )( i4 [INT] [INT] [INT] [INT] ))
           …
        )
        …
    )
)
```

4. 各チャンクについて

ここでは BAC6 に記述する各チャンクについて詳細を説明します。

4.1. （Head）チャンク

（bacVersion [FLOAT]）チャンクには BAC のバージョンを記述し、ファイルヘッダーを定義します。BAC6 はバージョン 6.0 なので、 [FLOAT]には 6.0 と文字列で定義します。
【参照】図 2「Head チャンク定義例」

```
( Head
        ( bacVersion 6.0 )
)
```

図 2「Head チャンク定義例」

4.2. （Figure）チャンク

（Figure）チャンクはモデルデータ（ポリゴンメッシュ）の情報を定義します。
（Figure）チャンクはモデルデータの各情報を定義するために、子チャンクを幾つか持ちます。
【参照】図 3「Figure チャンク定義例」

```
( Figure
        ( name … )
        ( Textures … )
        ( Materials … )
        ( Vertices … )
        ( Bones … )
        ( Polygons … )
        ( DynamicPolygons … )
)
```

図 3「Figure チャンク定義例」

モデルデータにおける各情報の詳細の定義については、（Figure）チャンク内の子チャンクで行ないます。
以下、子チャンクに関する詳細を説明します。

4.2.1. （name）チャンク

（Figure）チャンク内の（name [STRING]）チャンクはモデルデータの名前を文字列 255 文字（byte）以内で定義します。

モデルの名前は必ずしも定義する必要はありませんが、定義する場合は（name "" ）のようにダブルクォーテーションで囲む必要があります。

【参照】図 4「name チャンク定義例」

```
( name "figure" )
```

図 4「name チャンク定義例」

【関連】4.2.6.1.1.（name）チャンク
4.2.9.1.1.（name）チャンク

4.2.2. （Textures）チャンク

（Textures）チャンクはモデルデータに使用されているテクスチャ画像のピクセルサイズ（幅、高さ）を定義します。

テクスチャ情報がないモデルデータ（ポリゴンメッシュ）には、（Figure）チャンク内にこのチャンクを定義する必要はありません。

このチャンクは、定義する場合は必ず（Figure）チャンク内に定義してください。

【関連】4.2.7.（TextureCoords）チャンク

4.2.2.1 .（i2 [INT] [INT]）チャンク

テクスチャ画像のピクセルサイズ（幅、高さ）を定義します。

このチャンクには、それぞれ上から順に 0 から始まるインデックス ID が自動的に割り当てられ、インデックス ID は（material）チャンク内の（textureIndex [INT]）チャンクによって参照されます。

また（i2）チャンクは（Textures）チャンク内でのみ定義することができ、他のチャンク内では使用できません。

【参照】図 5「Textures チャンク」

```
( Textures
        ( i2 256 256 ) ; index 0
        ( i2 128 128 ) ; index 1
        …
)
```

図 5「Textures チャンク定義例」

【関連】4.2.4.1.5.（textureIndex）チャンク

4.2.3. （Colors）チャンク

（Colors）チャンクはモデルデータに使用されているカラーポリゴン用のカラーテーブルを定義します。
モデルデータにカラーポリゴンがない場合は、このチャンクを（Figure）チャンクに定義する必要はありません。
このチャンクは、定義する場合は必ず（Figure）チャンク内に定義してください。

4.2.3.1. （f3 [FLOAT] [FLOAT] [FLOAT]）チャンク

カラーポリゴン用のカラーテーブルを定義します。このチャンクにはそれぞれ上から順に 0 から始まるインデックス ID が自動的に割り当てられ、（material）チャンク内の（colorIndex [INT] ）チャンクによって参照されます。
カラーの有効範囲は RGB（0.0, .0.0, 0.0）を黒色として、RGB（1.0, 1.0, 1.0）を白色とした範囲でカラー情報を記述します。
【参照】図 6「Colors チャンク定義例」

```
( Colors
        ( f3 0.000 0.000 0.000 ) ; index 0 【黒】
        ( f3 1.000 1.000 1.000 ) ; index 1 【白】
        …
)
```

図 6「Colors チャンク定義例」

【関連】4.2.4.1.6.（colorIndex）チャンク

4.2.4. （Materials）チャンク

（Materials）チャンクはモデルデータのポリゴンに設定されている任意の表示属性タイプを定義します。このチャンクは、定義する場合は必ず（Figure）チャンク内に定義してください。
以下では（Materials）チャンク内で定義する各チャンクについて説明します。
【参照】図 7「Materials チャンク定義例」

```
( Materials
        ( material
                ( blendMode   normal )
                ( doubleFace  true )
                ( transparent false )
                ( lighting    true )
                ( textureIndex -1 )
                ( colorIndex    0 )
                ( specular      0.000 )
                ( alpha         0.000 )
                ( shininess     0.000 )
        )
        …
)
```

図 7「Materials チャンク定義例」

4.2.4.1.（material）チャンク

（material）チャンクはポリゴンに設定されている属性を定義します。（Materials）チャンク内にこのチャンクを必ず最低でも１つは登録しなければいけません。

また、（material）チャンクはそれぞれ上から順に 0 から始まるインデックス ID が自動的に割り当てられ、このインデックス ID は（Polygons）チャンク内の（face）チャンクによって参照されます。

【関連】4.2.8.1.（face）チャンク

以下では（material）チャンク内で定義する各チャンク(4.2.4.1.1～4.2.4.1.9)について説明します。（material）チャンク内で定義する各チャンクの記述順序に規定はありません。

4.2.4.1.1. （blendMode [normal | add | sub | half]）チャンク

（blendMode [normal | add | sub | half]）チャンクにはポリゴンが表示されるフラグメント処理を以下の属性からいずれか１つ定義します。

※ d_c = デスティネーションカラー、 s_c = ソースカラー

normal	標準のフラグメント処理をします(置換)	$d_c = s_c$
add	ポリゴンのカラーを塗り重ねます(加算)	$d_c = d_c + s_c$
sub	ポリゴンのカラー値を減算します(減算)	$d_c = d_c - s_c$
half	ポリゴンカラーの半分の値で重ねます(半透過)	$d_c = 0.5\ d_c + 0.5\ s_c$

4.2.4.1.2. （doubleFace [true | false]）チャンク

（doubleFace [true | false]）チャンクはポリゴンの表面と裏面の描画について定義します。

true	ポリゴンの両面描画を有効にします
false	ポリゴンの表面描画を有効にし、裏面は描画しません

4.2.4.1.3. （transparent [true | false]）チャンク

（transparent [true | false]）チャンクはテクスチャの BMP カラーパレット 0 番の色を透過させるかどうかを定義します。

true	透過効果を有効にします
false	透過効果を無効にします

4.2.4.1.4. （lighting [true | false]）チャンク

（lighting [true | false]）チャンクはライティング効果について定義します。

true	ライティング効果を有効にします
false	ライティング効果を無効にします

4.2.4.1.5. （textureIndex [INT]）チャンク

（textureIndex [INT]）は（Textures）チャンクで自動的に割り当てられたインデックス ID を定義します。テクスチャポリゴンとして処理しない場合は -1 を定義します。

【関連】4.2.2.（Textures）チャンク

4.2.4.1.6. （colorIndex [INT]）チャンク

（colorIndex [INT]）チャンクは（Colors）チャンクで登録したカラーID を定義します。

定義する登録 ID は必ず(Colors)チャンク内で定義する（f3）チャンクに自動的に割り当てられるインデックス ID でなければいけません。カラーポリゴンとして処理しない場合は -1 を定義します。

【関連】4.2.3.（Colors）チャンク

4.2.4.1.7. （specular [FLOAT]）チャンク

（specular [FLOAT]）チャンクは環境マッピングを設定する鏡面反射フラグ値です。値の有効範囲は 0.00～1.00 で、値は **0.25** 以上で環境マッピングによる鏡面反射処理を有効にします。

4.2.4.1.8. （alpha [FLOAT]）チャンク

(alpha [FLOAT])チャンクはテクスチャの半透過アルファ値を定義します。値の有効範囲は 0.0 ～ 1.0 になります。

この機能は V3 ではサポートしていません。従って必要のない場合は(material)チャンク内にこのチャンクを定義する必要はありません。

4.2.4.1.9. （shininess [FLOAT]）チャンク

(shininess [FLOAT])チャンクは環境マッピングに設定する鏡面反射係数を定義します。値の有効範囲は 0.0 ～ 1.0 になります。

この機能は V3 ではサポートしていません。従って必要のない場合は(material)チャンク内にこのチャンクを定義する必要はありません。

4.2.5. （Vertices）チャンク

（Vertices）チャンクはモデルデータを構成する頂点位置ならびに、その頂点に対する法線ベクトルを定義します。

【参照】図 8「Vertices チャンク定義例」

```
( Vertices
        ( coords
                ( pnt -1.000 -1.000 0.000 ) ; index 0
                ( pnt 1.000 -1.000 0.000 )  ; index 1
                ( pnt 1.000 1.000 0.000 )   ; index 2
                ( pnt -1.000 0.000 0.000 )  ; index 3
                …
        )
        ( normals
                ( vct 0.000 0.000 1.000 )  ; index 0
                ( vct 0.000 0.000 1.000 )  ; index 1
                ( vct 0.000 0.000 1.000 )  ; index 2
                ( vct 0.000 0.000 1.000 )  ; index 3
                …
        )
)
```

(pnt)チャンク 0 番のインデックスが定義する頂点座標は、(vct)チャンクのインデックス 0 番が定義する法線ベクトル値と対の関係性を保持して定義しなくてはいけません。

このように定義する頂点座標と法線ベクトル値は、頂点数だけ関係性を維持する必要があります。

図 8「Vertices チャンク定義例」

4.2.5.1.（coords）チャンク

（coords）チャンクはモデルデータを形成するポリゴンの頂点座標を定義します。
このチャンクは必ず（Vertices）チャンク内に定義しなければなりません。

4.2.5.1.1. （pnt [FLOAT] [FLOAT] [FLOAT]）チャンク

（pnt）チャンクはポリゴンを形成する三次元頂点位置（XYZ）をモデル座標系で定義します。
このチャンクは上から順に 0 から始まるインデックス ID が自動的に割り当てられ、このインデックス ID は（normals）チャンク内で列挙される（vct）チャンクのインデックス ID と一致していなければなりません。
またインデックス ID は（Polygons）チャンク等で使用される（face）チャンクによって参照されます。

【関連】4.2.8.（Polygons）チャンク

4.2.5.2.（normals）チャンク

（normals）チャンクはモデルデータを形成するポリゴンの法線ベクトルを定義します。
このチャンクは必ず（Vertices）チャンク内に定義しなければなりません。

4.2.5.2.1. （vct [FLOAT] [FLOAT] [FLOAT]）チャンク

(vct)チャンクはポリゴン形成する頂点の法線ベクトル(XYZ)をモデル座標系で定義します。
このチャンクは上から順に 0 から始まるインデックス ID が自動的に割り当てられ、このインデックス ID は(coords)チャンク内に列挙される(pnt)チャンクのインデックス ID と一致していなければなりません。
またインデックス ID は（Polygons）チャンク等で使用される（face）チャンクによって参照されます。

【関連】4.2.8.（Polygons）チャンク

4.2.6. （Bones）チャンク

（Bones）チャンクはモデルデータに割り当てられているボーンの階層構造及び基本となる各ボーン座標系を定義します。

モデルデータを構成する上で、（Figure）チャンク内には最低でも１つのボーンを定義する必要があります。また（Bones）チャンク内で最初に定義する（bone）チャンクは、ボーンの階層構造における最上位に位置するルートボーンを表し、このルートボーンは必ず１つにする必要があります。

以下では（Bones）チャンクに含まれる（bone）チャンクの説明を行います。

【参照】図 8「ボーン定義例」図 9「ボーン構造例」

```
( Bones
    ( bone
        ( name            "parent" )
        ( hasChild        true )
        ( hasBrother      false )
        ( translate       0.000 2.000 0.000 )
        ( rotate          0.000 2.000 1.000 )
        ( handle          0.000 3.000 0.000 )
        ( vertexIndices
            0 1 2 3
        )
    )
    ( bone
        ( name            "child_1" )
        ( hasChild        false )
        ( hasBrother      true )
        ( translate       2.000 0.500 0.000 )
        ( rotate          2.000 0.500 1.000 )
        ( handle          2.000 2.000 0.000 )
        ( vertexIndices
            4 5 6 7
        )
    )
    ( bone
        ( name            "child_2" )
        ( hasChild        false )
        ( hasBrother      false )
        ( translate       1.000 0.500 0.000 )
        ( rotate          1.000 0.500 1.000 )
        ( handle          1.500 1.500 0.000 )
        ( vertexIndices
            8 9 10 11
        )
    )
)
```

図 8「ボーン定義例」

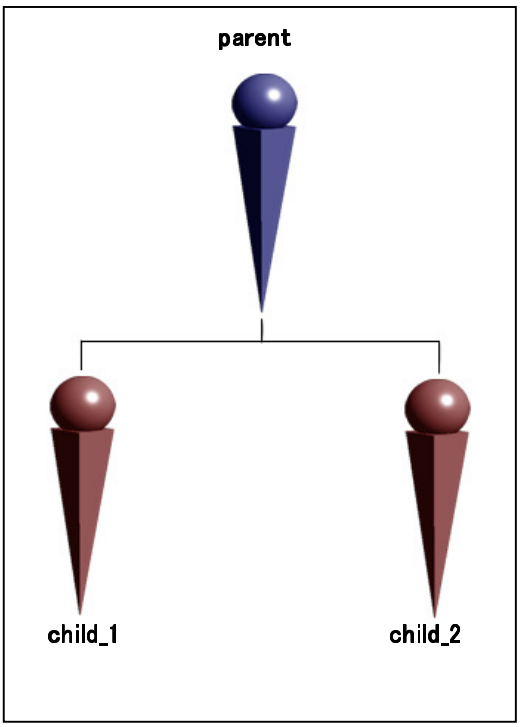


図 9「ボーン構造例」

※ボーンアニメーションは、TRA ファイルにおいて各ボーン座標系からの移動として定義されます。

4.2.6.1.（bone）チャンク

（bone）チャンクは１つのボーン情報を定義する各チャンクの親チャンクになり、モデルデータに割り当てられているボーンの数だけ（bone）チャンクを（Bones）チャンク内に定義する必要があります。

（bone）チャンクは自身と他のボーンとの階層構造などの関係性を定義する子チャンクを持ちます。

以下では、ボーンの詳細情報を定義する各チャンク（4.2.6.1.1～4.2.6.1.7）について説明します。

4.2.6.1.1. （name [STRING]）チャンク

（name）チャンクは、対象ボーンの名前を定義します。

対象ボーンの名前は必ずしも定義する必要はありませんが、定義する場合は（name ""）のように必ずダブルクォーテーションで囲んでください。

【関連】4.2.1（name）チャンク

4.2.6.1.2. （hasChild [true | false]）チャンク

（hasChild）チャンクは、対象ボーンに子ボーンが存在するか否かを定義します。

true	対象ボーンに子ボーンを定義する
false	対象ボーンに子ボーンを定義しない

TRUE が定義されている場合は、次に定義されている（bone）チャンクがそのボーンの子として定義されていることになります。

【参照】図 10「親子ボーン関係図」

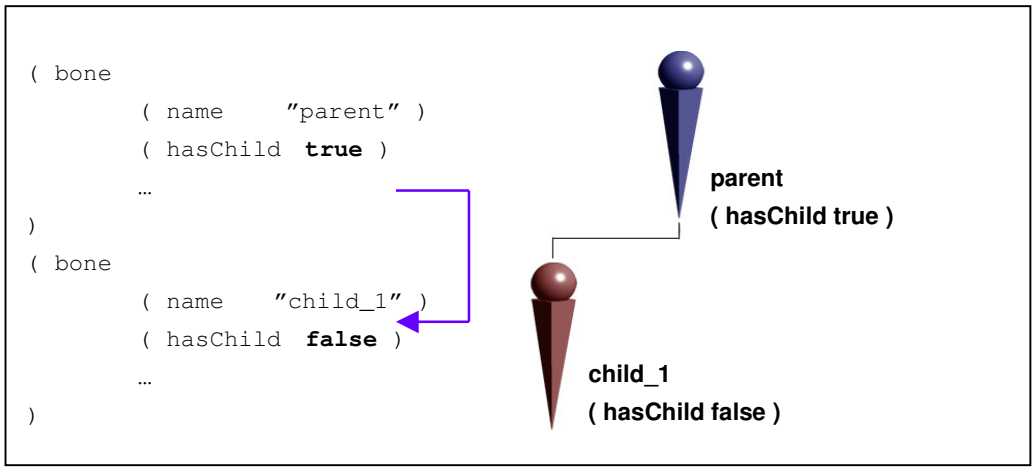


図 10「親子ボーン関係図」

【関連】4.2.6.2 ボーン構造の補足

4.2.6.1.3. （hasBrother [true | false]）チャンク

（hasBrother [true | false]）チャンクは、対象ボーンの同じ階層に他のボーンが存在するか否かを定義します。

true	このボーンに同階層ボーンは存在する
false	このボーンに同階層ボーンは存在しない

TRUE が定義されている場合は、対象ボーンと同じ階層のボーンが存在することを定義することになります。同じ階層における最後のボーンを定義する際は、必ず（hasBrother）チャンクには **FALSE** を定義します。

【参照】図 11「同一階層ボーン図」

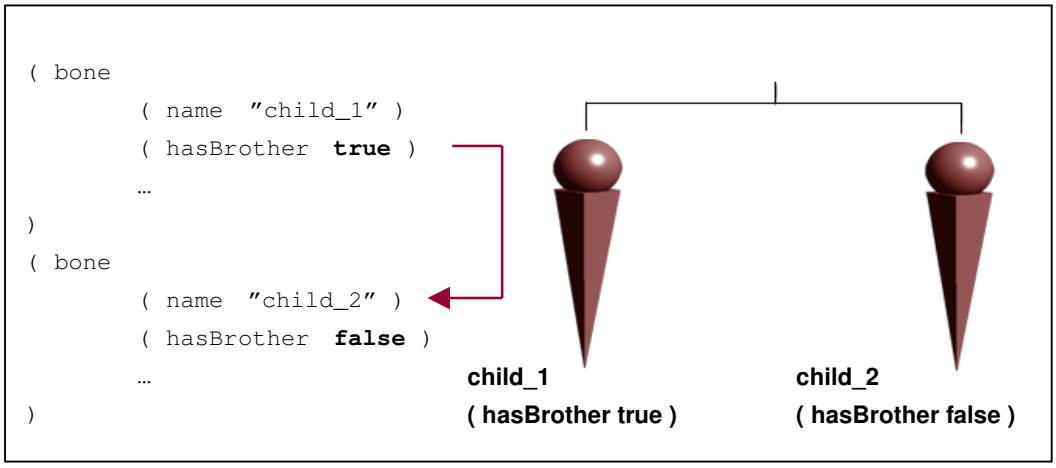


図 11「同一階層ボーン図」

【関連】4.2.6.2 ボーン構造の補足

4.2.6.1.4. （translate [FLOAT] [FLOAT] [FLOAT]）チャンク

(translate)チャンクは、対象ボーン座標のモデル座標系における原点位置を定義します。

ボーン座標系の原点位置がモデル座標系の原点位置と一致する場合は、（translate 0.0 0.0 0.0）と定義することができます。

【参照】図 12「ボーン座標系関係図」

【関連】4.2.6.1.5.（handle チャンク）、4.2.6.1.6（rotate）チャンク

4.2.6.1.5. （handle [FLOAT] [FLOAT] [FLOAT]）チャンク

（handle）チャンクは、対象ボーン座標系のモデル座標系における+Y 軸の位置を（translate）チャンクで定義した原点位置と共に定義します。

+Y 軸のモデル座標系における方向ベクトルは

$$\overrightarrow{(\text{handle})} - \overrightarrow{(\text{translate})}$$

となります。従って、+Y 軸の方向ベクトル

$$\overrightarrow{(\text{handle})} - \overrightarrow{(\text{translate})}$$

は、Z 軸に方向ベクトル

$$\overrightarrow{(\text{rotate})} - \overrightarrow{(\text{translate})}$$

と直交関係にある必要があります。

（translate）チャンクが（translate 0.0 0.0 0.0）のとき、ボーン座標系の Y 軸とモデル座標系の Y 軸を一致させるには、（handle 0.0 1.0 0.0）と定義することができます。

【参照】図 12「ボーン座標系関係図」

【関連】4.2.6.1.4.（translate チャンク）、4.2.6.1.6（rotate）チャンク

4.2.6.1.6. （rotate [FLOAT] [FLOAT] [FLOAT]）チャンク

（rotate）チャンクは、対象ボーン座標系のモデル座標系における+Z 軸の位置を（translate）チャンクで定義した原点位置と共に定義します。

+Z 軸のモデル座標系における方向ベクトルは

$$\overrightarrow{(\text{rotate})} - \overrightarrow{(\text{translate})}$$

となります。従って、+Z 軸の方向ベクトル

$$\overrightarrow{(\text{rotate})} - \overrightarrow{(\text{translate})}$$

は、+Y 軸の方向ベクトル

$$\overrightarrow{(\text{handle})} - \overrightarrow{(\text{translate})}$$

と直交関係にある必要があります。

（translate）チャンクが（translate 0.0 0.0 0.0）のとき、ボーン座標系の Z 軸とモデル座標系の Z 軸を一致させるには、（rotate 0.0 0.0 1.0）と定義することができます。

【参照】図 12「ボーン座標系関係図」

【関連】4.2.6.1.4.（translate チャンク）、4.2.6.1.5（handle）チャンク

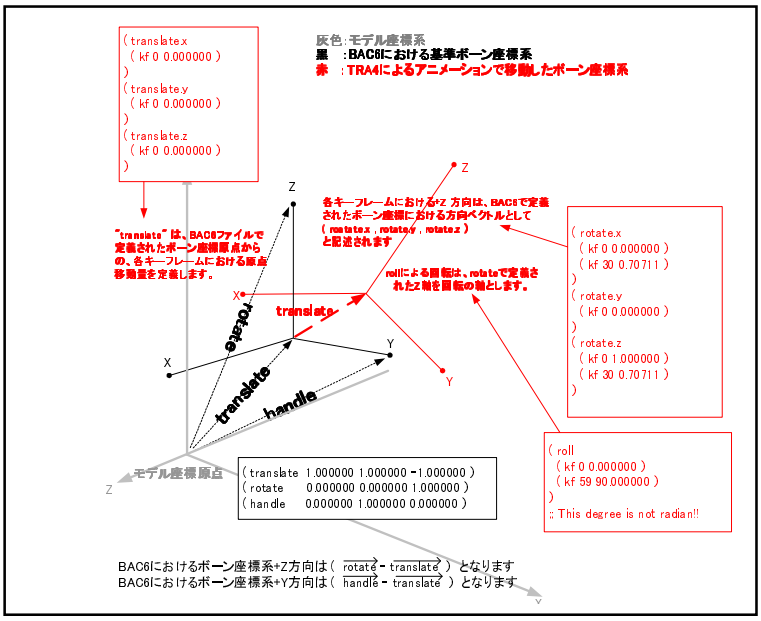


図 12「ボーン座標系関係図」

4.2.6.1.7. （vertexIndices [INT] [INT] … … ）チャンク

(vertexIndices)チャンクは、対象ボーンに属している頂点 ID を定義します。この頂点 ID とは（coords）チャンク内で定義されたインデックス番号を定義します。インデックス ID は上から順に 0 番から始まります。

対象ボーンに属している頂点がなければ、定義する必要はありませんが、どのボーンにも属していない頂点を含むポリゴンは表示されません。

【関連】4.2.5.1.（coords）チャンク

4.2.6.2. ボーン構造の補足

BAC6 におけるボーンの階層構造は、（hasChild）チャンクと（hasBrother）チャンクによって定義することは説明しました。

ここでは階層構造の定義における両チャンクの関係を説明します。ボーンの階層構造は、（hasChild）チャンクの定義を優先的に各ボーンを定義する必要があります。つまり子ボーンの存在の有無により、（bone）チャンクを定義する順番を決定します。下記の図 14「ボーン階層」を参照してください。

図 14 のようなボーン構造を持つモデルデータを定義する際には、図 13「BAC6 ボーン定義」のように定義します。

「child_1」ボーンの（bone）チャンクを定義した後に、その子ボーンである「child_1_1」ボーンの（bone）チャンクを定義します。そして、「child_1_1」ボーンには子ボーンが存在しないので（hasChild）チャンクは FALSE に定義できます。ここで「child_1」ボーンの階層定義を終えます。「child_1」は（hasBrother）チャンクが TRUE なので同じ階層にいる「child_2」ボーンを定義します。

以上のように子のボーン定義を優先し、同じ階層のボーン定義は子ボーンが存在しなくなるまで続ける必要があります。そして子のボーン定義が終わり次第同じ階層のボーンを定義します。

【関連】4.2.6.1.2（hasChild）チャンク
4.2.6.1.3（hasBrother）チャンク

```
( bone
        ( name  "parent" )
        ( hasChild    true )
        ( hasBrother  false )
        …
)
( bone
        ( name  "child_1" )
        ( hasChild    true )
        ( hasBrother  true )
        …
)
( bone
        ( name  "child_1_1" )
        ( hasChild    false )
        ( hasBrother  false )
        …
)
( bone
        ( name  "child_2" )
        ( hasChild    false )
        ( hasBrother  false )
        …
)
```

図 13「BAC6 ボーン定義」

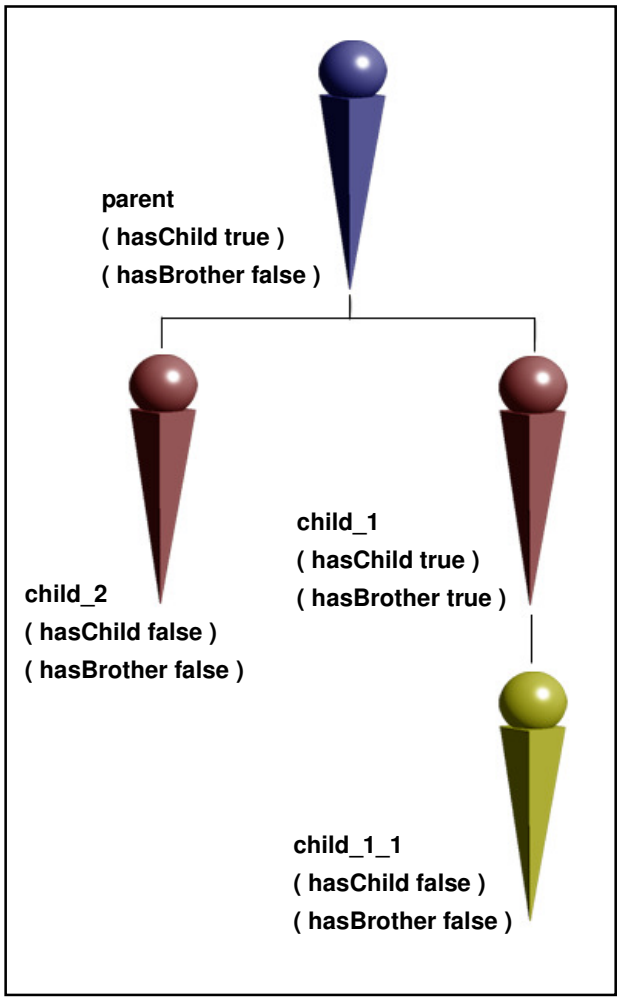


図 14「ボーン階層」

4.2.7. （TextureCoords）チャンク

（TextureCoords）チャンクはモデルデータに設定されているテクスチャの UV 座標値を定義します。

モデルデータにテクスチャ情報がない場合は（Figure）チャンクにこのチャンクを定義する必要はありません。

V3 の UV 座標は（0, 0）から（1, 1）の範囲にあります。従って、この範囲を超える場合は（0, 0）から（1.0、1.0）の範囲内でパディングを行ってください。

【参照】図 15「V3 テクスチャ座標系」

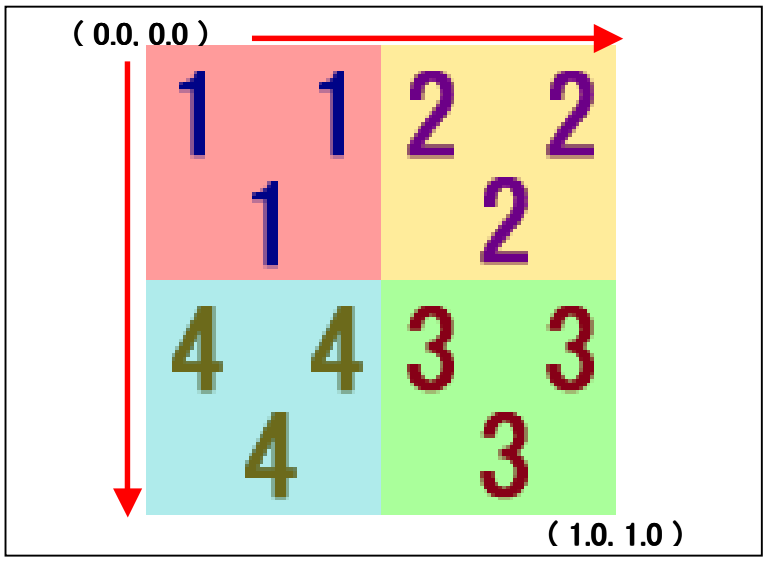


図 15「V3 テクスチャ座標系」

【関連】4.2.2.（Textures）チャンク 4.2.8.（Polygons）チャンク

以下では（TextureCoords）チャンク内で定義する、UV 座標値を定義する子チャンクについて説明します。

4.2.7.1.（f2 [FLOAT] [FLOAT]）チャンク

テクスチャの UV 座標値は、左から U 座標値を定義し、次に V 座標値として定義します。

このチャンクは上から順に 0 から始まるインデックス ID が自動的に割り当てられます。インデックス ID は（Polygons）チャンクに属する（face）チャンクによって参照されます。

【参照】図 16「TextureCoords チャンク」

```
( TextureCoords
        ( f2 0.000 0.000 ) ; index 0
        ( f2 0.000 1.000 ) ; index 1
        ( f2 1.000 0.000 ) ; index 2
        ( f2 1.000 1.000 ) ; index 3
)
```

図 16「TextureCoords チャンク」

【関連】4.2.8.1.（face）チャンク

4.2.8. （Polygons）チャンク

（Polygons）チャンクはモデルデータを形成する各ポリゴン情報を定義します。
（Polygons）チャンクは必ず（Figure）チャンク内に定義しなければなりません。

4.2.8.1.（face）チャンク

（face）チャンクは 1 ポリゴン面を形成する結線情報やポリゴンに設定されているマテリアルとの関連付け及び、テクスチャ UV 座標の関連付け等をインデックスによって定義します。
ポリゴン面が 3 頂点から構成される場合は（i3）チャンクを使用し、4 頂点から構成される場合は（i4）チャンクを使用します。5 頂点以上から構成さるポリゴン面はサポートしていません。
以下にて（i3）チャンクと（i4）チャンクの説明を行います。

【関連】4.2.5. (Vertices)チャンク
4.2.4.（Materials）チャンク
4.2.7.（TextureCoords）チャンク

4.2.8.1.1. （i3）チャンク

（i3)チャンクは 3 頂点から構成される 1 ポリゴン面の頂点座標及びテクスチャ座標を定義します。
【参照】図 17「(i3）チャンク定義」

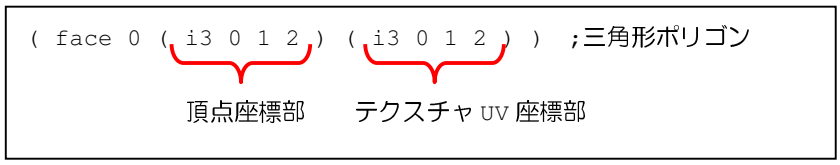


図 17「(i3）チャンク定義」

頂点座標部に定義されているインデックス番号は、(coords）チャンク内で定義する（pnt）チャンクに自動的に割り当てられるインデックス番号を定義します。
またテクスチャ UV 座標インデックス部に定義されているインデックス番号は、(TextureCoords）チャンク内で定義する（f2）チャンクに自動的に割り当てられるインデックス番号を定義します。
（i3）チャンクは（face）チャンク以外で定義することはできません。

【関連】4.2.5.1.1.（pnt）チャンク
4.2.7.1.（f2）チャンク

4.2.8.1.2. （i4）チャンク

（i4)チャンクは 4 頂点から構成される 1 ポリゴン面の頂点座標及びテクスチャ座標を定義します。
【参照】図 18「(i4）チャンク定義」

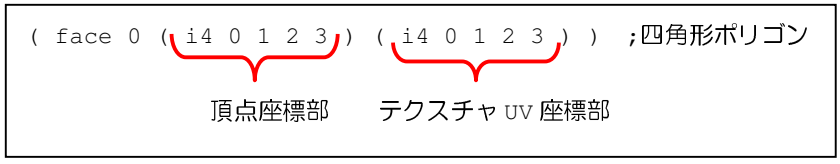


図 18「(i4）チャンク定義」

頂点座標部に定義されているインデックス番号は、(coords）チャンク内で定義する（pnt）チャンクに自動的に割り当てられるインデックス番号を定義します。
またテクスチャ UV 座標インデックス部に定義されているインデックス番号は、(TextureCoords）チャンク内で定義する（f2）チャンクに自動的に割り当てられるインデックス番号を定義します。

（i4）チャンクは（face）チャンク以外で定義することはできません。

【関連】4.2.5.1.1.（pnt）チャンク
4.2.7.1.（f2）チャンク

4.2.8.2.（face）チャンク定義の補足

ここでは、（face）チャンクを利用したポリゴン面の定義例を幾つか挙げます。

4.2.8.2.1. マテリアルの関連付け

マテリアルの関連付けは、（Materials）チャンク内で定義する（material）チャンクに自動的に割り当てられたインデックス番号を（face）チャンクに定義します。

【参照】図 19「マテリアルの関連付け」

```
( Polygons
    ( face 0 ( i3 0 1 2 )  ( i3 0 1 2 ))    ; インデックス 0 番のマテリアル
    ( face 1 ( i4 3 4 5 6 )( i4 3 4 5 6 )) ; インデックス 1 番のマテリアル
)    （material）チャンクのインデックス番号
```

図 19「マテリアル関連付け」

【関連】4.2.4.1.（material）チャンク

4.2.8.2.2. ポリゴンの関連付け

ポリゴンを構成する結線情報の定義は、3 頂点ポリゴンの場合は（face）チャンクと（i3）チャンクにより定義します。

また 4 頂点ポリゴンの場合は（face）チャンクと（i4）チャンクにより定義します。

【参照】図 20「ポリゴン関連付け」

```
( Polygons
    ( face 0 ( i3 0 1 2 )  ( i3 0 1 2 ) )   ; 三角形ポリゴン
    ( face 1 ( i4 3 4 5 6 )( i4 3 4 5 6 ) ); 四角形ポリゴン
)       頂点インデックス番号によるポリゴン形成
```

図 20「ポリゴン関連付け」

【関連】4.2.8.1.1. (i3)チャンク
4.2.8.1.2.（i4）チャンク

4.2.8.2.3. テクスチャ座標情報

ポリゴンに割り当てられているテクスチャ UV を定義するには、3 頂点ポリゴンの場合は（face）チャンクと（i3）チャンクにより定義します。4 頂点ポリゴンの場合は（face）チャンクと（i4）チャンクにより定義します。

ポリゴンにテクスチャが適用されていないカラーポリゴンの場合は-、–1 をインデックス ID に定義します。

【参照】図 21「テクスチャ座標関連付け」

```
( Polygons
    ( face 0 ( i3 0 1 2  ) ( i3 -1 -1 -1 ) ) ; テクスチャポリゴン（未使用）
    ( face 1 ( i4 3 4 5 6 )( i4 3 4 5 6  ) ) ; テクスチャポリゴン（使用）
)
```

図 21「テクスチャ座標関連付け」

【関連】4.2.8.1.1. (i3)チャンク　4.2.8.1.2.（i4）チャンク

4.2.9. （DynamicPolygons）チャンク

DynamicPolygon とは、(face) チャンクに定義するポリゴンに対してテクスチャ UV 座標やマテリアルを変更するアニメーションパターンを言います。

テクスチャ UV 座標を変化させるダイナミックポリゴンのパターン定義には、必ず（Textures）チャンク及び（TextureCoords）チャンクの定義が必要です。これは、後に解説する（group）チャンクに属する（face）チャンクが、テクスチャ座標のインデックスを参照するためです。

以下、図 22「（DynamicPolygons）チャンク定義例 1」、図 23「（DynamicPolygons）チャンク定義例 2」を参照してください。

図 22 はテクスチャを変化させるダイナミックポリゴンのパターンの定義例です。

```
( DynamicPolygons
    ( group      ; インデックス 0 番
        ( name "pattern0" )
        ( face 0 ( i4 3 2 0 1 )( i4 3 4 1 0 ) )
    )

    ( group      ; インデックス 1 番
        ( name "pattern1" )
        ( face 0 ( i4 3 2 0 1 )( i4 6 7 4 3 ) )
    )
)
```

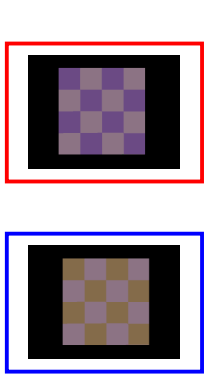


図 22「（DynamicPolygons）チャンク定義例 1」

図 23 はマテリアルを変化させるダイナミックポリゴンの定義例です。

```
( DynamicPolygons
    ( group      ; インデックス 0 番
        ( name "pattern0" )
        ( face 0 ( i4 3 2 0 1 )( i4 -1 -1 -1 -1 ) )
    )

    ( group      ; インデックス 1 番
        ( name "pattern1" )
        ( face 1 ( i4 3 2 0 1 )( i4 -1 -1 -1 -1 ) )
    )
)
```

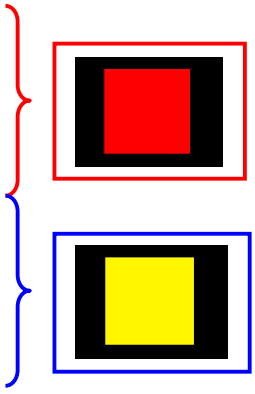


図 23「（DynamicPolygons）チャンク定義例 2」

【関連】4.2.1.（name）チャンク
4.2.8.1.（face）チャンク

ダイナミックポリゴンを使用しないデータの場合は、（Figure）チャンク内に（DynamicPolygons）チャンクを定義する必要はありません。
以下では（DynamicPolygons）チャンクに定義する（name）チャンク及び（face）チャンクについて説明します。

4.2.9.1.（group）チャンク

（group）チャンクはダイナミックポリゴンのアニメーションパターンを定義します。
（group）チャンクは上から順番に、インデックス 番号 0 から始まる ID が自動的に割り当てられます。この（group）チャンクのインデックス数だけダイナミックポリゴンのアニメーションパターンが定義されたことになります。インデックス ID は別紙「TRA File Ver.4.0 Data Format Specification」に記載されている（kgf）チャンクによって参照されます。
図 22「（DynamicPolygons）チャンク定義例 1」及び図 23「（DynamicPolygons）チャンク定義例 2」を参照してください。

4.2.9.1.1. （name）チャンク

（name）チャンクには任意でダイナミックポリゴンのアニメーションパターン名をつけることができます。
図 22「（DynamicPolygons）チャンク定義例 1」及び図 23「（DynamicPolygons）チャンク定義例 2」を参照してください。

【関連】4.2.1.（name）チャンク

4.2.9.1.2. （face）チャンク

（group）チャンク内の（face）チャンクは、ダイナミックポリゴンにおける各パターンのテクスチャ UV 座標及びマテリアルを定義します。
図 22「（DynamicPolygons）チャンク定義例 1」及び図 23「（DynamicPolygons）チャンク定義例 2」を参照してください。

【関連】4.2.8.1.（face）チャンク

5. サンプルコード

5.1. BAC6 サンプル1

ここではテクスチャが適用されているモデルデータを出力した BAC6 ファイルのサンプルを記述しています。

【Sample01.bac】

```
;BAC

( Head
	( bacVersion 6.0 )
)
( Figure
	( name "TexturePolygonSample" )
	( Textures
		( i2 256 256 )
	)
	( Colors
		( f3 0.500 0.500 0.500 )
	)
	( Materials
		( material
			( blendMode   normal )
			( doubleFace  true )
			( transparent false )
			( lighting    true )
			( textureIndex 0 )
			( colorIndex  -1 )
			( specular    1.000 )
		)
	)
	( Vertices
		( coords
			( pnt -1.500 0.000 0.000 )
			( pnt -1.500 3.000 0.000 )
			( pnt 1.500 0.000 0.000 )
			( pnt 1.500 3.000 0.000 )
		)
		( normals
			( vct 0.000 0.000 1.000 )
			( vct 0.000 0.000 1.000 )
			( vct 0.000 0.000 1.000 )
			( vct 0.000 0.000 1.000 )
		)
	)
```

```
        ( Bones
                ( bone
                        ( name          "bone" )
                        ( hasChild      false )
                        ( hasBrother    false )
                        ( translate     0.000 0.000 0.000 )
                        ( rotate        0.000 0.000 1.000 )
                        ( handle        0.000 1.000 0.000 )
                        ( vertexIndices
                                0 1 2 3
                        )
                )
        )
        ( TextureCoords
                ( f2 0.000 0.000 )
                ( f2 0.000 1.000 )
                ( f2 1.000 0.000 )
                ( f2 1.000 1.000 )
        )
         ( Polygons
                ( face 0 ( i4 3 2 0 1 ) ( i4 2 3 1 0 ) )
        )
)
```

5.2. BAC6 サンプル 2

ここではダイナミックポリゴンが適用されているモデルを出力した BAC6 ファイルのサンプルを記述しています。

【Sample02.bac】

```
;BAC

( Head
	( bacVersion 6.0 )
)
( Figure
	( name "DynamicPolygonSample" )
	( Textures
		( i2 128 128 )
	)
	( Colors
		 ( f3 1.000 0.500 0.500 )
	)
	( Materials
		( material
			( blendMode   normal )
			( doubleFace  true )
			( transparent false )
			( lighting    true )
			( textureIndex 0 )
			( colorIndex  -1 )
			( specular    1.000 )
		)
		( material
			( blendMode    normal )
			( doubleFace   true )
			( transparent  false )
			( lighting     true )
			( textureIndex -1 )
			( colorIndex   0 )
			( specular     1.000 )
		)
	)
	( Vertices
		( coords
			( pnt -400.000 -200.000 0.000 )
			( pnt -400.000 200.000 -0.000 )
			( pnt 0.000 -200.000 0.000 )
			( pnt 0.000 200.000 -0.000 )
			( pnt 400.000 -200.000 0.000 )
			( pnt 400.000 200.000 -0.000 )
		)
```

```
		( normals
			( vct 0.000 0.000 1.000 )
			( vct 0.000 0.000 1.000 )
			( vct 0.000 0.000 1.000 )
			( vct 0.000 0.000 1.000 )
			( vct 0.000 0.000 1.000 )
			( vct 0.000 0.000 1.000 )
		)
	)
	( Bones
		( bone
			( name        "bone" )
			( hasChild    false )
			( hasBrother  false )
			( translate   0.000 0.000 0.000 )
			( rotate      0.000 0.000 100.000 )
			( handle      0.000 100.000 0.000 )
			( vertexIndices
				0 1 2 3 4 5
			)
		)
	)
	( TextureCoords
		( f2 0.000 0.000 )
		( f2 0.000 0.500 )
		( f2 0.000 1.000 )
		( f2 0.500 0.000 )
		( f2 0.500 0.500 )
		( f2 0.500 1.000 )
		( f2 1.000 0.000 )
		( f2 1.000 0.500 )
		( f2 1.000 1.000 )
	)
	( Polygons
		( face 1 ( i4 3 2 0 1 ) ( i4 -1 -1 -1 -1 ) )
	)
	( DynamicPolygons
		( group
			( name "pattern0" )
			( face 0 ( i4 5 4 2 3 ) ( i4 3 4 1 0 ) )
		)
		( group
			( name "pattern1" )
			( face 0 ( i4 5 4 2 3 ) ( i4 6 7 4 3 ) )
		)
	)
)
```

BAC File Ver.6.0 Data Format Specification

MascotCapsuleV3

日本語版

バージョン　2.1
発行日　2007 年 5 月 28 日
発行者　株式会社 エイチアイ
〒153-0043　東京都目黒区東山 1-4-4 目黒東山ビル 5F
http://www.hicorp.co.jp/