



Motion 仕様書

(7/13/98)

目次

1. 概要	3
2. オブジェクト構造体	3
3. カメラ構造体	5
4. ライト構造体	5
5. モーション構造体	8
6. オブジェクトモーション	12
7. カメラモーション	14
8. ライトモーション	15
9. その他	17

1. 概要

Ninja はモデル、カメラ、ライトのモーションを同一構造体で定義します。キーフレームを設定する単位にデータを配列化し、そのポインタテーブルでモーション全体を定義します。この方法により、必要箇所のみのモーションを持たせたり、パラメータごとにキーフレーム補完を行うことができます。また、カメラとライトのモーション共通部分を使い回すこともできます。

2. オブジェクト構造体

オブジェクト構造体は、child、sibling ポインタで他のオブジェクトを連結することにより、親子階層モデルを表現します。親子階層を child、sibling の順にトレースし、その順番でノードを一系列に並べたモーションデータとオブジェクト構造体の pos、ang、scl の値を組み合わせ、階層モデルのモーションを実現します。evalflags はマトリクス計算を省略したり、モーション時のオブジェクト動作を制御します。モデルのデータ構造に関しては、Basic Model 仕様書または Chunk Model 仕様書を参照してください。

Basic Object 構造体

```
typedef struct obj {
    Uint32      evalflags; /* 評価方法の最適化フラグ */
    NJS_MODEL   *model;    /* モデル構造体ポインタ */
    Float       pos[3];    /* 平行移動 */
    Angle       ang[3];    /* 回転 */
    Float       scl[3];    /* スケール */
    struct obj  *child;    /* 子ども object へのポインタ */
    struct obj  *sibling;  /* 兄弟 object へのポインタ */
} NJS_OBJECT;
```

Chunk Object 構造体

```
typedef struct cnkobj {
    Uint32      evalflags; /* 評価方法の最適化フラグ */
    NJS_CNK_MODEL *model;  /* モデル構造体ポインタ */
    Float       pos[3];    /* 平行移動 */
    Angle       ang[3];    /* 回転 */
    Float       scl[3];    /* スケール */
    struct obj  *child;    /* 子ども object へのポインタ */
    struct obj  *sibling;  /* 兄弟 object へのポインタ */
} NJS_CNK_OBJECT;
```

- モデルの親子階層を与えます。
- model にはポリゴン、TRIMESH(連続ポリゴン)がセットされます。

evalflags フラグ説明

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

これらのフラグはコンバータで設定されます。

- NJD_EVAL_UNIT_POS は、平行移動量がゼロの場合に設定されます。平行移動のマトリクス計算が省略されます。
- NJD_EVAL_UNIT_ANG は、回転量がゼロの場合に設定されます。回転のマトリクス計算が省略されます。
- NJD_EVAL_UNIT_SCL は、スケールが xyz とともに 1 の場合に設定されます。スケールのマトリクス計算が省略されます。
- NJD_EVAL_UNIT_POS, NJD_EVAL_UNIT_ANG, NJD_EVAL_UNIT_SCL の三つがセットされていると、すべてのマトリクス演算に加え、マトリクスの pushpop も省略されます。
- NJD_EVAL_HIDE フラグは、ユーザによって立てられます。このフラグが立っている場合、モデルが描画されません。モデルに装着されたガンとブレードの切り替えなどに使用します。
- NJD_EVAL_BREAK フラグは、ユーザによって立てられます。このフラグが立っている場合、**child の探索**をここで打ち切ります。例えば、ルートノードでこのフラグを立てることにより、モデル全体が消えます。NJD_EVAL_BREAK は、モーションと組み合わせるとデータがずれるので、ルートノードのみで使用するをお勧めしますが、ユーザの責任において途中のノードで使用することもできます。
- LightWave3D では、回転の評価の順番が ZXY となります。Ninja は通常 XYZ の順番であるため、LightWave3D の場合の評価順をライブラリで実行するためのフラグとして、NJD_EVAL_ZXY_ANG が用意されます。このフラグが ON の場合、回転の計算順序が ZXY に変更されます。
- NJD_EVAL_SKIP は、このノードがモーションデータに含まれないことを示します。モーション実行時はモーションのノードを次に進めずに、オブジェクト構造体の値でマトリクス計算を行い、次のノードに進みます。これにより、ポリゴンモデルの構成が異なるモデルでも、骨の構造が同じであればモーションが行えます。
- NJD_EVAL_SHAPE_SKIP は、このノードがシェイプモーションデータに含まれないことを示します。

3. カメラ構造体

カメラの構造体は以下のとおりです。

モーションとして使用するパラメータは、ポジション、ベクトル、ロール、画角の4つです。

NJS_CAMERA 構造体

```
typedef struct{
    Float px, py, pz;      (カメラのポジション)
    Float vx, vy, vz;      (カメラの単位方向ベクトル[ローカル Z 軸])
    Angle roll;            (カメラのロール)
    Angle ang;             (カメラの画角)
    Float n_clip;          (ニアクリップ)
    Float f_clip;          (ファークリップ)
    NJS_VECTOR k, l;       (カメラのローカル X, Y 軸)
} NJS_CAMERA
```

4. ライト構造体

ライトの構造体は以下のとおりです。

モーションできるパラメータとして、スポットライト以外の場合はポジション、ベクトル、カラーの4つが、スポットライトの場合はこれに前方の限界値、後方の限界値、内側の限界角、外側の限界角が加わります。ライト構造体に関する詳しい説明は、「ライトの設定方法」を参照してください。

NJS_LIGHT 構造体

```
struct {
    NJS_MATRIX      mtrx;      (光源マトリクス)
    NJS_POINT3      pnt;      (光源のポジション)
    NJS_VECTOR      vctr;      (光源の単位方向ベクトル)
    BOOL            stat;      (ステータス：光源使用・不使用)
    Int             reserve;    (リザーブ)
    NJS_LIGHT_CAL    ltcal;     (ライト計算用構造体)
    NJS_LIGHT_ATTR   attr;     (アトリビュート構造体)
} NJS_LIGHT;

<stat>
#define NJS_LIGHT_ON      光を反映させる。
#define NJS_LIGHT_OFF     光を反映させない。
```

NJS_LIGHT_ATTR 構造体

```
struct {
    Int      lsrc;      (光源の種類)
    Float    ispc;      (スペキュラライトの強さ：0 から 1)
    Float    idif;      (デフューズの強さ：0 から 1)
    Float    iamb;      (アンビエントの強さ：0 から 1)
    Float    nrang;      (光の強度が最大である距離：前方の限界値)
```

Float	frang;	(光の強度をカットオフする距離：後方の限界値)
void*	func;	(コールバック関数のポインタ)
Angle	iang;	(光の強度が最大である角度：内側の限界角)
Angle	oang;	(光の強度をカットオフする角度：外側の限界角)
NJS_ARGB	argb;	(光の色)
} NJS_ LIGHT_ATTR		

<lsrc>

光源の種類。

#define NJD_SPOT_LIGHT	スポットライト
#define NJD_DIR_LIGHT	平行光源
#define NJD_POINT_LIGHT	点光源
#define NJD_AMBIENT	アンビエント
#define NJD_SPEC_DIR	平行光源ハイライト
#define NJD_SPEC_POINT	点光源ハイライト
#define NJD_LAMBERT_DIR	平行光源ランバート
#define NJD_LAMBERT_POINT	点光源ランバート
#define NJD_PHONG_DIR	平行光源フォン
#define NJD_PHONG_POINT	点光源フォン
#define NJD_USER_LIGHT	ユーザ設定ライト
#define NJD_BLOCK_LIGHT	ブロックライト
等	

<ispc, idif, iamb>

光のバランス（以下の式による）

$\text{SPECULAR}(R,G,B) \times \text{ispc} + \text{DIFFUSE}(R,G,B) \times \text{idif} + \text{AMBIENT}(R,G,B) \times \text{iamb}$

ただし、上限（下限）はクランプします。

<near, far>

NJD_POINT_LIGHT（点光源） NJD_SPOT_LIGHT（スポットライト）

等で指定される光の有効範囲（距離）

nrang 光が上限値である距離の限界値。デフォルト値:1.f

frang 光の計算を行う距離の限界値。デフォルト値:65535.f

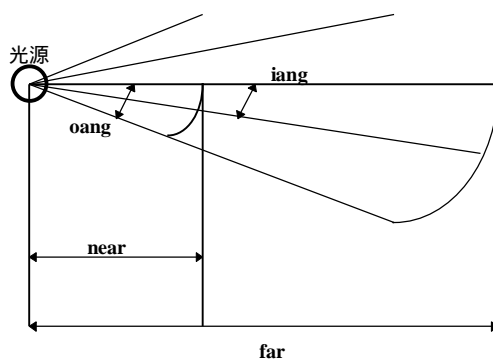
<iang, oang>

NJD_SPOT_LIGHT（スポットライト）等で指定される光の有効範囲（距離）

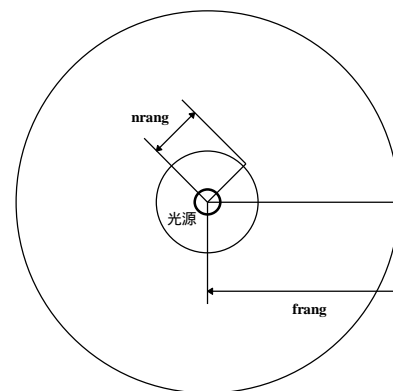
iang 光が上限値である角度の限界値。デフォルト値:(DEG)10.f

oang 光の計算を行う角度の限界値。デフォルト値:(DEG)30.f

（例：スポットライト）

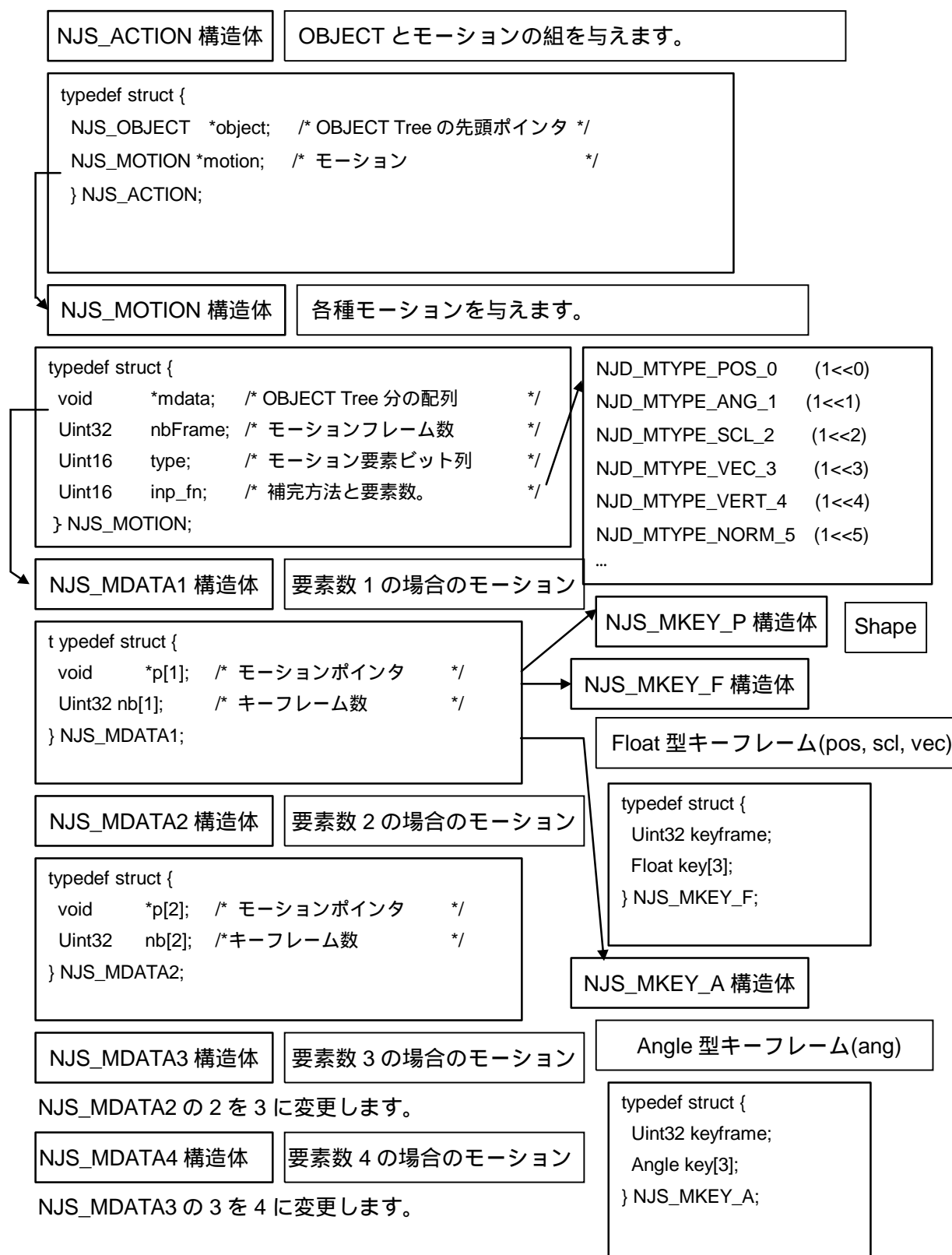


（例：点光源）



5. モーション構造体

構造体図



モデル用アクション構造体

```
typedef struct {
    NJS_OBJECT    *object;    /* OBJECT Tree の先頭ポインタ */
    NJS_MOTION    *motion;    /* モーションリスト           */
} NJS_ACTION;
```

- object は親子階層を持つ tree 構造です。
- motion には、object にしたい motion をセットします。

カメラ用アクション構造体

```
typedef struct {
    NJS_CAMERA    *camera;    /*カメラ構造体へのポインタ */
    NJS_MOTION    *motion;    /*モーションリスト           */
}NJS_CACTION
```

ライト用アクション構造体

```
typedef struct {
    NJS_LIGHT     *light;     /*ライト構造体へのポインタ */
    NJS_MOTION    *motion;    /*モーションリスト           */
}NJS_LACTION
```

モーション構造体

```
typedef struct {
    void          *mdata;     /* OBJECT Tree 分の配列      */
    Uint32        nbFrame;    /* モーションフレーム数      */
    Uint16        type;       /* モーション要素ビット列     */
    Uint16        inp_fn;     /* 補完方法と要素数。        */
} NJS_MOTION;
```

- mdata には、OBJECT Tree に含まれる全 NJS_OBJECT に対応する数の NJS_MDATA が配列として入ります。
- NJS_MDATA では、モーション構成要素数により NJS_MDATA1～5 構造体をそれぞれ使用します。

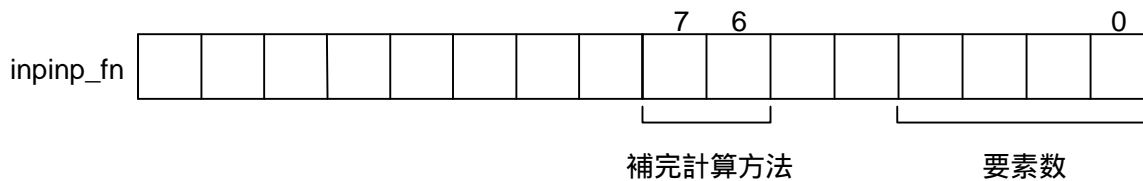
```
#define NJD_MTYPE_POS_0          (1<<0)  /* NJS_MKEY_F を利用          */
#define NJD_MTYPE_ANG_1        (1<<1)  /* NJS_MKEY_A を利用          */
#define NJD_MTYPE_SCL_2        (1<<2)  /* NJS_MKEY_F を利用          */
#define NJD_MTYPE_VEC_3        (1<<3)  /* NJS_MKEY_F を利用          */
#define NJD_MTYPE_VERT_4       (1<<4)  /* NJS_MKEY_P を利用          */
#define NJD_MTYPE_NORM_5       (1<<5)  /* NJS_MKEY_P を利用          */
#define NJD_MTYPE_TARGET_3     (1<<6)  /* NJS_MKEY_F を利用          */
#define NJD_MTYPE_ROLL_6       (1<<7)  /* NJS_MKEY_A1 を利用         */
#define NJD_MTYPE_ANGLE_7      (1<<8)  /* NJS_MKEY_A1 を利用         */
#define NJD_MTYPE_RGB_8        (1<<9)  /* NJS_MKEY_UI32 を利用       */
#define NJD_MTYPE_INTENSITY_9  (1<<10) /* NJS_MKEY_F1 を利用         */
#define NJD_MTYPE_SPOT_10      (1<<11) /* NJS_MKEY_SPOT を利用       */
#define NJD_MTYPE_POINT_10     (1<<12) /* NJS_MKEY_F2 を利用         */
```

- 通常のモーションにおいて平行移動 (pos) \ 回転 (ang) \ スケール (scl) \ 三要素が含まれる場合は、NJS_MDATA 3 が利用されます。後ろの番号にはモーション要素の順番を与えます。
- ベクトル成分 vec は、pos とともに光源、カメラのモーションに利用します。

- 補完計算方法を inp_fn の上位 2 ビットで指定します。

```
#define NJD_MTYPE_LINER    0x0000 /* 線形補完          */
#define NJD_MTYPE_SPLINE   0x0040 /* スプライン補完   */
#define NJD_MTYPE_USER     0x0080 /* ユーザ関数補完   */
#define NJD_MTYPE_MASK     0x00c0 /* 抽出マスク       */
```

- どの構造体を使っているかを示すために、inp_fn の下位 4 ビットには要素数が格納されます。



NJS_MDATA1 ~ 5 構造体

```
typedef struct {
void      *p[1]; /* モーションポインタ */
Uint32    nb[1]; /* キーフレーム数     */
} NJS_MDATA1;

typedef struct {
void      *p[2]; /* モーションポインタ */
Uint32    nb[2]; /* キーフレーム数     */
} NJS_MDATA2;

typedef struct {
void      *p[3]; /* モーションポインタ */
Uint32    nb[3]; /* キーフレーム数     */
} NJS_MDATA3;
```

- すべてのデータは、キーフレーム構造で表現されます。
- nb[i]には p[i]要素のモーションキーフレーム数が入ります。

また光源用に MDATA4、MDATA5 が定義されます。MDATA5 はスポットライト光源の場合のみに使用されます。

```
typedef struct {
void      *p[4]; /* モーションポインタ */
Uint32    nb[4]; /* キーフレーム数     */
} NJS_MDATA4;

typedef struct {
void      *p[5]; /* モーションポインタ */
Uint32    nb[5]; /* キーフレーム数     */
} NJS_MDATA5;
```

キー構造体

```
typedef struct {
    Uint32    keyframe;    /* キーフレーン番号          */
    Float     key[3];      /* Float 型キー値(配列3)      */
} NJS_MKEY_F;
```

- 平行移動 (POS) スケール (SCL) ベクトル (VEC) に利用します。

```
typedef struct {
    Uint32    keyframe;    /* キーフレーン番号          */
    Angle     key[3];      /* Angle 型キー値(配列)      */
} NJS_MKEY_A;
```

- 回転 (ANG) に利用します。

```
typedef struct {
    Uint32    keyframe;    /* キーフレーン番号          */
    Void      *key;        /* ポインタ                  */
} NJS_MKEY_P;
```

- シェイプ(SHAPE)に利用します。

```
typedef struct {
    Uint32    keyframe;    /* キーフレーン番号          */
    Uint32    key;         /* 符号無し int32 型キー値    */
} NJS_MKEY_UI32;
```

- ライト色(RGB)に利用します。

```
typedef struct {
    Uint 32   keyframe;    /* キーフレーン番号          */
    Sint32    key;         /* 符号有り int32 型キー値    */
} NJS_MKEY_A1;
```

- カメラロール(ROLL)、角度 (ANGLE)に利用します。

```
typedef struct {
    Uint32    keyframe;    /* キーフレーン番号          */
    Float     key;         /* Float 型キー値            */
} NJS_MKEY_F1;
```

- ライト強度(INTENSITY)に利用します。

```
typedef struct {
    Uint32    keyframe;    /* キーフレーン番号          */
    Float     key[2];      /* Float 型キー値(配列3)      */
} NJS_MKEY_F2;
```

- 点光源(POINT)に利用。

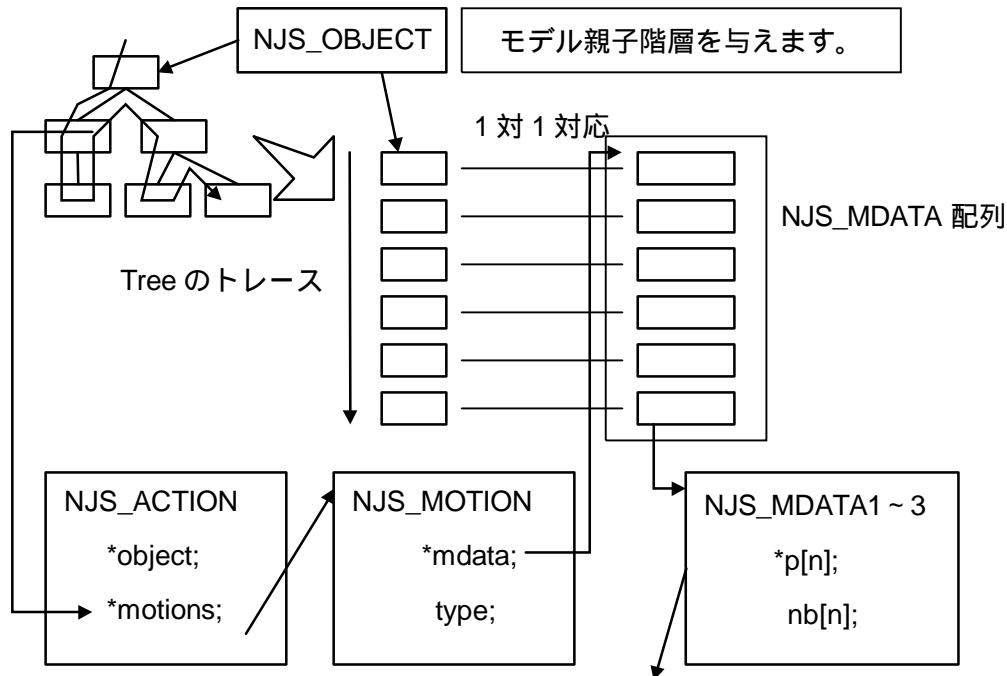
```
typedef struct {
    Uint32    keyframe;    /* キーフレーン番号          */
    Float     nrang;       /* float 型・前方の限界値のキー値 */
    Float     frang;       /* float 型・後方の限界値キー値   */
    Angle     iang;        /* Angle 型・内側の限界角キー値   */
    Angle     oang;        /* Angle 型・外側の限界角キー値   */
} NJS_MKEY_SPOT;
```

- スポットライト(SPOT)に利用します。

6. オブジェクトモーション

構造体関連図

Object Tree



(例) pos のみの場合：要素数 1 なので NJS_MDATA1 を使います。

NJS_MOTION 構造体の type = NJD_MKEY_POS_0;

NJS_MKEY_F pos[] = {, , ...};

NJS_MDATA1 mdata[] = {{pos, poskey_n}, ...};

(例) pos, ang, scl の場合：要素数 3 なので、NJS_MDATA3 を使います。

type = NJD_MTYPE_POS_0 | NJD_MTYPE_ANG_1 | NJD_MTYPE_SCL_2;

MKEY_F pos[] = {, , ...};

MKEY_A ang[] = {, , ...};

MKEY_F scl[] = {, , ...};

MDATA3 mdata[] = {{pos, ang, scl, poskey_n, angkey_n, sclkey_n}, ...};

(例) 光源の pos, vec の場合：要素数 2 なので NJS_MDATA2 を使います。

type = NJD_MTYPE_POS_0 | NJD_MTYPE_VEC_3;

NJS_MKEY_F pos[] = {, , ...};

NJS_MKEY_F vec[] = {, , ...};

NJS_MDATA2 mdata[] = {{pos, vec, poskey_n, veckey_n}, ...};

構造解説

- モーションはすべてキーフレームデータで与えられます。
- ユーザはキーフレームデータを線形補完、スプライン補完を使用することにより、モーションを実行します。
- Ninja ライブラリでは、コールバック関数により補完方法をユーザ定義できます(現在未対応)。
- キーフレーム番号はゼロから始まります。マイナス値は使用できません。

Position	平行移動
Angle	回転
Scale	拡大縮小
Vertex	ポリゴン頂点移動によるアニメーション。(shape)
Normal	ポリゴン頂点移動アニメーション(shape)の際の法線。

- オブジェクトモーションの要素として、上記の五つを考えます。
- ライブラリ実装面の問題から、shape データである Vertex、Normal について、Position、Angle、Scale (.nam) とは別データ (.nas) として出力します。このため最大要素数は三要素です。オブジェクトモーション用として、NJS_MDATA1 ~ 3 構造体が利用されます。光源、カメラ用に NJS_MDATA4、NJS_MDATA5 が定義されています。
- NJS_MDATA の各要素を格納するポインタは void であり、各場合におけるデータの格納順番を規定する必要があります。

```
#define NJD_MTYPE_POS_0      (1<<0)      /* NJS_MKEY_F を利用      */
#define NJD_MTYPE_ANG_1      (1<<1)      /* NJS_MKEY_A を利用      */
#define NJD_MTYPE_SCL_2      (1<<2)      /* NJS_MKEY_F を利用      */
#define NJD_MTYPE_VEC_3      (1<<3)      /* NJS_MKEY_F を利用      */
#define NJD_MTYPE_VERT_4      (1<<4)      /* NJS_MKEY_P を利用      */
#define NJD_MTYPE_NORM_5      (1<<5)      /* NJS_MKEY_P を利用      */
```
- define 文字列の最後に示す番号の若いデータが先に並びます。上記フラグはモーション構造体のメンバ type に設定されます。
(例) pos と ang の場合

```
type = NJD_MTYPE_POS_0 | NJD_MTYPE_ANG_1;
mdata[] = {pos, ang, ...}
```
- モーション補完方法は、type の上位 2 ビットで指定されます。

```
#define NJD_MTYPE_LINER      0x0000
#define NJD_MTYPE_SPLINE     0x0040
#define NJD_MTYPE_USER       0x0080
#define NJD_MTYPE_MASK       0x00c0
```
- NJD_MTYPE_LINER は線形補完です。
- NJD_MTYPE_SPLINE はスプライン補完です。
- NJD_MTYPE_USER はユーザ定義のルーチンによる補完です。

- ルートが pos, ang で他は ang のみのモーションモデルでは、NJS_MDATA2 構造体を使用し、ルート以外の pos には NULL ポインタを使用することにより対応します。

```
type = NJD_MTYPE_POS_0 | NJD_MTYPE_ANG_1;
NJS_MDATA2 mdata[] = {
    { *pos1, *ang1 },
    { NULL, *ang2 },
    { NULL, *ang3 },
    .... }
```

- この場合、ang2, ang3 を左につめてはならないことに注意してください。

7. カメラモーション

カメラは親子階層を構成しないため、1 オブジェクトに対するモーション構造と基本的に同じです。

アクション構造体には NJS_CACTION を使用します。

カメラには

- 位置(POS)
- 方向(VEC) またはターゲット(TARGET)
- ロール(ROLL)
- 画角(ANGLE)

の4要素があり、必要に応じて NJS_MTYPE_1 から NJS_MTYPE_4 を使用します。

```
#define NJD_MTYPE_POS_0      (1<<0)    /* NJS_MKEY_F を利用      */
#define NJD_MTYPE_VEC_3     (1<<3)    /* NJS_MKEY_F を利用      */
#define NJD_MTYPE_TARGET_3 (1<<6)    /* NJS_MKEY_F を利用      */
#define NJD_MTYPE_ROLL_6    (1<<7)    /* NJS_MKEY_A1 を利用     */
#define NJD_MTYPE_ANGLE_7   (1<<8)    /* NJS_MKEY_A1 を利用     */
```

Vec はベクトル、Target はターゲットのポジションを意味します。

NJD_MTYPE_VEC_3 と NJD_MTYPE_TARGET_3 が同じ3なのは、同時に使えないことを示します。

- フリーカメラ(向きをベクトルで持つカメラ)の場合

```
type=NJD_MTYPE_POS_0|NJD_MTYPE_VEC_3|NJD_MTYPE_ROLL_6|NJD_MTYPE_ANGLE_7;
```

- ターゲットカメラ(向きをターゲット位置で持ち画角アニメーションを持つカメラ)の場合

```
type=NJD_MTYPE_POS_0|NJD_MTYPE_TARGET_3|NJD_MTYPE_ROLL_6|NJD_MTYPE_ANGLE_7;
```

- 補完計算方法を inp_fn の上位2ビットで指定します。オブジェクトモーションと同じです。

8. ライトモーション

ライトは親子階層を構成しないため、1 オブジェクトに対するモーション構造と基本的に同じです。

アクション構造体は NJS_LACTION を使用します。

ライトのモーションの対象としては、点光源、平行光源、スポットライトを想定しています。

点光源には、

- 位置(POS)
- 範囲(POINT)
- 色(RGB)
- 光の強度(INTENSITY)

の4要素があります。

範囲の要素は、前方の限界値(NearRange)後方の限界値(FarRange)をひとまとめにします。

平行光源には、

- 位置(POS)
- 方向(VEC) またはターゲット(TARGET)
- 色(RGB)
- 光の強度(INTENSITY)

の4要素があります。

スポットライトには、平行光源の4要素に前方の限界値(near)、後方の限界値(far)、内側の限界角(iang)、外側の(oang)限界角をひとまとめにした、

- スポット(SPOT)

という要素を加えて、要素数は5つとなります。

よって、点光源、平行光源では NJS_MDATA_1 から NJS_MDATA_4 を使用し、スポットライトでは NJS_MDATA_1 から NJS_MDATA_5 を使用します。

```
#define NJD_MTYPE_POS_0          (1<<0)      /* NJS_MKEY_F を利用      */
#define NJD_MTYPE_VEC_3          (1<<3)      /* NJS_MKEY_F を利用      */
#define NJD_MTYPE_TARGET_3       (1<<6)      /* NJS_MKEY_F を利用      */
#define NJD_MTYPE_RGB_8          (1<<9)      /*NJS_MKEY_UI32 を利用    */
#define NJD_MTYPE_INTENSITY_9    (1<<10)     /* NJS_MKEY_F1 を利用     */
#define NJD_MTYPE_SPOT_10        (1<<11)     /* NJS_MKEY_SPOT を利用   */
#define NJD_MTYPE_POINT_10       (1<<12)     /* NJS_MKEY_F2 を利用     */
```

Vec はベクトル、Target はターゲットのポジションを意味します。

NJD_MTYPE_VEC_3 と NJD_MTYPE_TARGET_3 が同じ 3、NJD_MTYPE_SPOT_10 と NJD_MTYPE_POINT_10 が同じ 10なのは、同時に使用できないことを示します。

Type の設定例

- 点光源

```
type=NJD_MTYPE_POS_0|NJD_MTYPE_RGB_8|NJD_MTYPE_INTENSITY_9|  
NJD_MTYPE_POINT_11;
```

- 平行光源(向きをベクトルで持つ平行光源)の場合

```
type=NJD_MTYPE_POS_0|NJD_MTYPE_VEC_3|NJD_MTYPE_RGB_8|  
NJD_MTYPE_INTENSITY_9;
```

- スポットライト(向きをターゲットで持つスポットライト)の場合

```
type=NJD_MTYPE_POS_0|NJD_MTYPE_TARGET_3|NJD_MTYPE_RGB_8  
|NJD_MTYPE_INTENSITY_9|NJD_MTYPE_SPOT_10;
```

補完計算方法を `inp_fn` の上位 2 ビットで指定します。オブジェクトモーションと同じです。

9. その他

ライトファイルの先頭には、

```
#if USE_LIGHT_ALIGN
#pragma USE_ALIGNDATA(LightName)
#endif
```

という記述がありますが、これはS H用の記述で、アライメント調整を設定するための記述です。