



10/01/97

---

# Light の設定方法

1997/10/1

# NINJA LIGHT

## LIGHT の設定方法

### **void njCreateLight(NJS\_LIGHT\*, Int)**

パラメタ：        NJS\_LIGHT        \*ptr  
                 Int                lsrc  
説明：    光源の種類 lsrc を定め、ライト ptr を新たに登録します。  
戻り値：        なし  
備考：    なし

### **void njDeleteLight(NJS\_LIGHT\*)**

パラメタ：        NJS\_LIGHT        \*ptr  
説明：    設定されたライト ptr を削除します。  
戻り値：        なし  
備考：    なし

### **void njLightOff(NJS\_LIGHT\*)**

パラメタ：        NJS\_LIGHT        \*ptr  
説明：    設定されたライト ptr を反映させません。  
戻り値：        なし  
備考：    マクロで O K。

### **void njLightOn(NJS\_LIGHT\*)**

パラメタ：        NJS\_LIGHT        \*ptr  
説明：    設定されたライト ptr をモデルに反映させます。  
戻り値：        なし  
備考：    マクロで O K。

### **void njMultiLightMatrix(NJS\_LIGHT\*, NJS\_MATRIX\*)**

パラメタ：        NJS\_LIGHT        \*ptr  
                 NJS\_MATRIX        \*m  
説明：    njCreateLight によって登録されたライト行列に行列 m を掛けます。  
戻り値：        なし  
備考：    行列 m にはスケールファクタが入ってはならない。

### **void njSetLight(NJS\_LIGHT\*)**

パラメタ：        NJS\_LIGHT        \*ptr  
説明：    すでにツール等で定められたライト ptr を新たに登録します。

戻り値： なし

備考： なし

### **void njSetLightAlpha(NJS\_LIGHT\*, Float)**

パラメタ： NJS\_LIGHT \*ptr

Float alpha

説明： **njCreateLight** によって登録されたライトへアルファ値を設定します。

戻り値： なし

備考： 現在考慮中。

### **void njSetLightAngle(NJS\_LIGHT\*, NJS\_Angle, NJS\_Angle)**

パラメタ： NJS\_LIGHT \*ptr

NJS\_Angle iang

NJS\_Angle oang

説明： **njCreateLight** によって登録されたライトへ角度限界値を設定します。

戻り値： なし

備考： スポットライトのみ使用。（現在）

### **void njSetLightColor(NJS\_LIGHT\*, Float, Float, Float)**

パラメタ： NJS\_LIGHT \*ptr

Float red

Float green

Float blue

説明： **njCreateLight** によって登録されたライトへ RGB 値を設定します。

戻り値： なし

備考： なし

### **void njSetLightDirection(NJS\_LIGHT\*, Float, Float, Float)**

パラメタ： NJS\_LIGHT \*ptr

Float dx

Float dy

Float dz

説明： **njCreateLight** によって登録されたライトの光源方向を設定します。

戻り値： なし

備考： 平行光源、スポットライトのみ使用。（現在）

### **void njSetLightIntensity(NJS\_LIGHT\*, Float, Float, Float)**

パラメタ： NJS\_LIGHT \*ptr

Float spc

Float dif

Float amb

説明： **njCreateLight** によって登録されたライトの光の強度を設定します。

戻り値： なし

備考： なし

**void njSetLightLocation(NJS\_LIGHT\*, Float, Float, Float)**

パラメタ：       NJS\_LIGHT       \*ptr  
                  Float           px  
                  Float           py  
                  Float           pz

説明： **njCreateLight** によって登録されたライトの光源位置を設定します。

戻り値：       なし

備考： 点光源、スポットライトのみ使用。（現在）

**void njSetLightRange(NJS\_LIGHT\*, Float, Float)**

パラメタ：       NJS\_LIGHT       \*ptr  
                  Float           nrang  
                  Float           frang

説明： **njCreateLight** によって登録されたライトへ距離限界値を設定します。

戻り値：       なし

備考： スポットライト、点光源のみ使用。（現在）

**void njSetUserLight(NJS\_LIGHT\*, NJF\_LIGHT\_FUNC\*)**

パラメタ：       NJS\_LIGHT       \*ptr  
                  NJF\_LIGHT\_FUNC func

説明： ライト ptr にユーザ設定ライト関数 func を設定します。

戻り値：       なし

備考： なし

**void njUnitLightMatrix(NJS\_LIGHT\*)**

パラメタ：       NJS\_LIGHT       \*ptr

説明： **njCreateLight** によって登録されたライト行列を単位行列に設定します。

戻り値：       なし

備考： なし

**void njTranslateLightV(NJS\_LIGHT\*, NJS\_VECTOR\*)**

パラメタ：       NJS\_LIGHT       \*ptr  
                  NJS\_VECOTR    \*vctr

説明： **njCreateLight** によって登録されたライト行列をベクトル vctr で平行移動します。

戻り値：       なし

備考： なし

**void njTranslateLight(NJS\_LIGHT\*, Float, Float, Float)**

パラメタ：       NJS\_LIGHT       \*ptr  
                  Float           tx  
                  Float           ty  
                  Float           tz

説明： **njCreateLight** によって登録されたライト行列を ( tx, ty, tz ) で平行移動します。

戻り値： なし

備考： なし

### **void njRotateLightX(NJS\_LIGHT\*, NJS\_Angle)**

パラメタ：        NJS\_LIGHT        \*ptr

                 NJS\_Angle        ang

説明： **njCreateLight** によって登録されたライト行列を X 軸で角度 ang 回転します。

戻り値：        なし

備考： なし

### **void njRotateLightXYZ(NJS\_LIGHT\*, NJS\_Angle, NJS\_Angle, NJS\_Angle)**

パラメタ：        NJS\_LIGHT        \*ptr

                 NJS\_Angle        xang

                 NJS\_Angle        yang

                 NJS\_Angle        zang

説明： **njCreateLight** によって登録されたライト行列を X Y Z 軸で回転します。

戻り値：        なし

備考： なし

### **void njRotateLightY(NJS\_LIGHT\*, NJS\_Angle)**

パラメタ：        NJS\_LIGHT        \*ptr

                 NJS\_Angle        ang

説明： **njCreateLight** によって登録されたライト行列を Y 軸で角度 ang 回転します。

戻り値：        なし

備考： なし

### **void njRotateLightZ(NJS\_LIGHT\*, NJS\_Angle)**

パラメタ：        NJS\_LIGHT        \*ptr

                 NJS\_Angle        ang

説明： **njCreateLight** によって登録されたライトの行列を Z 軸で角度 ang 回転します。

戻り値：        なし

備考： なし

## **マクロ**

NJS\_LIGHT \*1 としています。

---

```
#define NJM_LIGHT_INIT_VECTOR(l)      l->vctr    ( ライトの行列計算前の方向 )
#define NJM_LIGHT_INIT_POINT(l) l->pnt      ( ライトの行列計算前の位置 )
#define NJM_LIGHT_MATRIX(l)          l->mtrx     ( ライトの行列 )
#define NJM_LIGHT_VECTOR(l)          (l->ltcal).lvctr ( ライトの現在の方向 )
#define NJM_LIGHT_POINT(l)           (l->ltcal).lpnt ( ライトの現在の位置 )
#define NJM_LIGHT_AMB(l)              (l->ltcal).amb ( 環境光強度 )
#define NJM_LIGHT_DIF(l) (l->ltcal).dif      ( デフューズ光強度 )
#define NJM_LIGHT_SPC(l) (l->ltcal).spc      ( スペキュラ光強度 )
#define NJM_LIGHT_EXP(l)              (l->ltcal).exp ( 指数：スペキュラの広がり )
#define NJM_LIGHT_COLOR(l)           (l->attr).argb ( ライトの色 )
```

## 使用法

光源計算は、位置・方向・カラー・種類等の必要な光源情報が記載されてあるライト構造体に基づいて行われます。ライト構造体は2種類のライト関数により設定されます。ライト関数 `njCreateLight` は一般的な使用法で、これは引数で決められた光源種類でライト構造体を新規に造ります。より詳しい情報は、`njSetLight....` を使用することでつけ加えることができます。他方のライト構造体の設定法は、`njSetLight` を用いるもので、予め情報が設定されている構造体に対して行います。これは、光源 (= ライト構造体) の登録を行う他は何も行わないことに注意してください。

登録した光源は、デフォルトでモデルへ反映されます。そこで、もしあるモデルへ光源の計算を止めたい時には、モデル描画の前に `njLightOff` を設定しなければなりません。`njLightOff`、`njLightOn` はカレントの状態を保ち続けることに注意してください。また、関数・引数・構造体等の詳しい情報はリファレンスを参照してください。

**例 1 )** ライト `lt1` をスポットライト、`lt2` を環境光 + 点光源 (ランバートモデル) に設定します。

```
#include <ninja.h>
.....
// ライトを宣言します。
NJS_LIGHT lt1, lt2;
.....
// 初期化ルーチンです。
/*ライトを初期化・登録します。*/
njCreateLight(&lt1, NJD_SPOT_LIGHT);
njSetLightAngle(&lt1, DegToAngle(30.f), DegToAngle(60.f));
njSetLightRange(&lt1, 1000.f, 1500.f);
njSetLightLocation(&lt1, 0.f, 10.f, 15.f);
njSetLightDirection(&lt1, 0.f, 1.f, 0.f);
njSetLightColor(&lt1, 1.f, 0.f, 0.f);
```

```
njCreateLight(&lt2, NJD_LAMBERTIAN_POINT);
/* ライト属性の各種設定します。 */
njSetLightColor(&lt2, 0.5f, 0.5f, 0.5f);
njSetLightIntensity(&lt2, 0.f, 1.f, 1.f); // デフォルト強度は ( 1.f, 1.f, 1.f ) です。
njSetLightRange(&lt2, 100.f, 1500.f);
```

```
.....
```

```
// 描画ルーチンは以下の通りです。
```

```
while(-1)
{
    .....

    /* ライト lt1, lt2 をモデルに反映させます。 */
    njDrawModel(...);

    .....

    /* ライト lt2 を消します。 */
    njLightOff(lt2);

    /* ライト lt1 をモデルに反映させます。 */
    njDrawModel(...);

    /* ライト lt2 を反映させます。 */
    njLightOn(lt2);

    .....
}
```

**例 2 )** 分岐処理等でライト lt1 のスポットライト色を変更し、平行光源 lt3 を新たに追加設定します。

```
.....
/* ライト lt1 のカラー変更します。 */
njSetLightColor(&lt1, 0.f, 1.f, 1.f);

/* ライトを初期化・登録します。 */
njCreateLight(&lt3, NJD_DIRECTIONAL_LIGHT);

/* ライト属性の各種設定します。 */
njSetLightDirection(&lt3, 1.f, 0.f, 0.f);
njSetLightIntensity(&lt3, 0.f, 1.f, 1.f);
```

```
.....
```

```
// 描画ルーチンは以下の通り。
```

---

```
while(-1)
{
    .....

    /* ライト lt1, lt2, lt3 をモデルに反映させます。 */
    njDrawModel(...);

    .....
}
```

**例 3 )** ライト lt にユーザ関数を設定します。

//ユーザ関数を設定します。(関数の引数は以下の通りです)

```
void
userfunc(NJS_ARGB* argb, NJS_POINT3* pnt, NJS_VECTOR* nml, NJS_LIGHT_PTR light)
{
    .....

    // ポリゴン法線と光線方向の内積をとります。
    deg = - nml->x * NJM_LIGHT_VECTOR(light).x
        - nml->y * NJM_LIGHT_VECTOR(light).y
        - nml->z * NJM_LIGHT_VECTOR(light).z;

    .....

    argb->a = deg * NJM_LIGHT_DIF(light).a;
    argb->r = deg * NJM_LIGHT_DIF(light).r;
    argb->g = deg * NJM_LIGHT_DIF(light).g;
    argb->b = deg * NJM_LIGHT_DIF(light).b;
}
```

//メインルーチン中です(一部省略)。

```
.....
njCreateLight(&lt, NJD_USER_LIGHT);

.....
/*ライト lt にユーザ関数 userfunc を設定*/
njSetUserLight (&lt, userfunc);
/*ライト lt のカラーの設定*/
njSetLightColor(&lt, 0.f, 1.f, 1.f);
.....
```

// 描画ルーチンは以下の通りです。

```
while(-1)
{
    .....

    /* ライト lt をモデルに反映させます。 */
```

```

DrawModel(...);

.....

}

```

## LIGHT 構造体の仕様

ユーザはライト構造体を原則として直接触る必要はありませんが、参考までにライト構造体の仕様を下記に示します。

Softimage \ Ninja	[NJS_MATERIAL 構造体]	[NJS_LIGHT_ATTR 構造体]
[ <i>specular</i> ]	argb or rgb	intensity_spec
[ <i>diffuse</i> ]	argb or rgb	intensity_diff
[ <i>ambient</i> ]	(argb or rgb :pending)	intensity_amb
[ <i>exponent</i> ]	exp	なし

*specular* : ハイライト。 **Softimage** では、通常 RGB 値に対しそれぞれ 0 ~ 1 を設定。

*diffuse* : 通常光。 **Softimage** では、通常 RGB 値に対しそれぞれ 0 ~ 1 を設定。

*ambient* : 環境光。 **Softimage** では、通常 RGB 値に対しそれぞれ 0 ~ 1 を設定。

*exponent* : ハイライトの広がり。 **Softimage** では通常 0 ~ 3 0 0 を設定。

( HSV はライブラリでサポートしない予定です。 )

### - NJS\_LIGHT 構造体のメンバ -

```

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

```

<stat>

**#define** NJS\_LIGHT\_ON 光を反映させます。

**#define** NJS\_LIGHT\_OFF 光を反映させません。

### - NJS\_LIGHT\_ATTR 構造体のメンバ -

```

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 )
}

```

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

**} NJS\_ LIGHT\_ATTR**

<lsrc>

光源の種類。

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

等

<ispc, idif, iamb>

光のバランス ( 以下の式によります )。

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

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

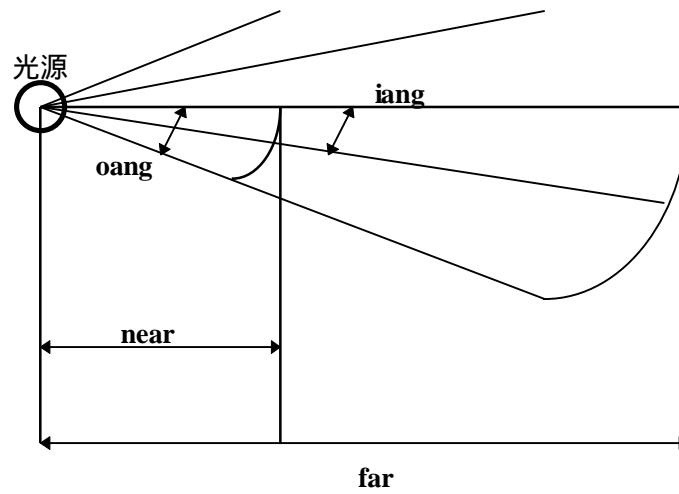
<iang, oang>

**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 )。

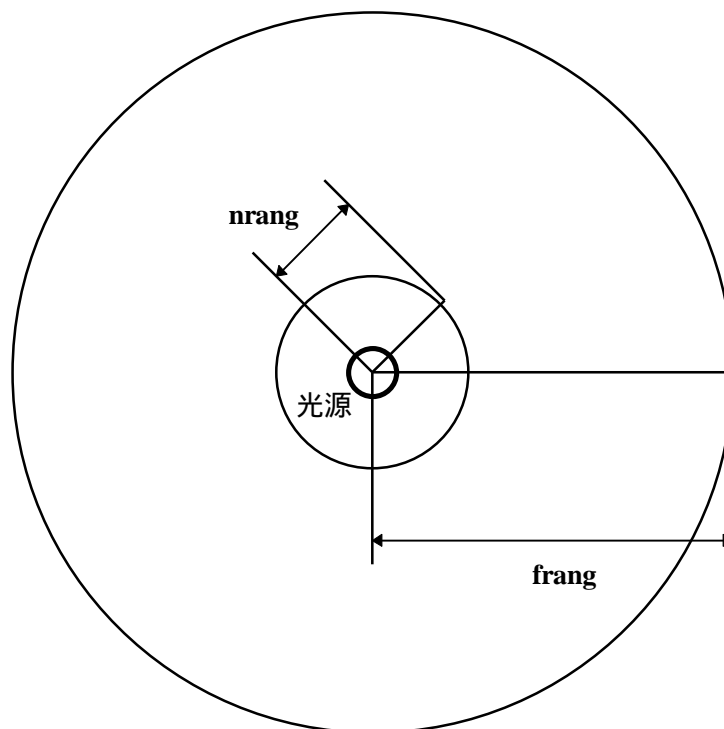
**iang** 光が上限値である角度の限界値。デフォルト値:(DEG)10.f

**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)



- NJS\_LIGHT\_CAL 構造体のメンバ -

```
struct
{
```

Float	ratten;	(減衰率：ブロックライトが使用)
Float	ipd;	(内積：ブロックライトが使用)

---

<b>Float</b>	<b>nrr;</b>	( 光源の限界判定値、近 : $\text{nrang} * \text{nrang}$ )
<b>Float</b>	<b>frr;</b>	( 光源の限界判定値、遠 : $\text{frang} * \text{frang}$ )
<b>Float</b>	<b>cosi;</b>	( 光源の限界判定値、内 : $\cos * \cos$ )
<b>Float</b>	<b>cose;</b>	( 光源の限界判定値、外 : $\cos * \cos$ )
<b>Float</b>	<b>idev;</b>	( 光源の分割判定値、内 )
<b>Float</b>	<b>odev;</b>	( 光源の分割判定値、外 )
<b>Float</b>	<b>rate;</b>	( 光源の減衰比率 - スポットライト用 )
<b>Float</b>	<b>intns;</b>	( 光源の強さ、0 ~ 1 )
<b>Int</b>	<b>exp;</b>	( 光源の拡散具合 )
<b>Int</b>	<b>reserve;</b>	( リザーブ )
<b>NJS_POINT3</b>	<b>lpnt;</b>	( 光源の位置 )
<b>NJS_VECTOR</b>	<b>lvctr;</b>	( 光源の方向ベクトル )
<b>NJS_VECTOR</b>	<b>lmvctr;</b>	( 光源の方向ベクトル : ブロックライトが使用 )
<b>NJS_ARGB</b>	<b>atten;</b>	( $\text{intns} * \text{argb}$ ( 光源色 ) )
<b>NJS_ARGB</b>	<b>amb;</b>	( $\text{iamb} * \text{atten}$ )
<b>NJS_ARGB</b>	<b>dif;</b>	( $\text{idif} * \text{atten}$ )
<b>NJS_ARGB</b>	<b>spc;</b>	( $\text{ispc} * \text{atten}$ )

**} NJS\_LIGHT\_CAL;**

<exp>

拡散幅をもたします。material 構造体で使します。