mod_SC3D_C 説明

Index

- 他のモジュールの推奨
- 一言説明
- 依存するモジュール
- 数学的な事前定義
- 定義の継承
- オブジェクトとは？
- オブジェクト構造体
- オブジェクトの撮影に関して
- 命令・関数リファレンス

＜ 他のモジュールの推奨 ＞

このモジュールに加えてさらに、モジュール「mod_SC3D_C2」をお勧めします。
mod_SC3D_C2 は mod_SC3D_C に比べてオブジェクト管理機能が大幅に強化されています。

＜ 一言説明 ＞

mod_SC3D シリーズの規則に基づいてオブジェクトを扱うモジュールです。

＜ 依存するモジュール ＞

mod_Sort2
mod_SC3D_B

上記のモジュールを使用可能な状態にしておいてください。

＜ 数学的な事前定義 ＞

あるベクトル →V について、「→V まわりの回転」と言うときは、時計の背中に→V の始点を鉛直にくっつけた時に時計の針が回る向きを正の回転とします。
角度は、特に断りのない限りラジアンで表現します。

＜ 定義の継承 ＞

モジュール mod_SC3D_A ， mod_SC3D_B で定義されたことは本モジュールでも引き続き有効となりま

す。

＜ オブジェクトとは？ ＞

点・線分・四角形平面・球 等の基本的な図形が相対的な位置関係を保ったまま一塊になったものを本モジュールでは「オブジェクト」と呼んでいます。

これに対し、オブジェクトに属する点・線分・四角形平面・球 等の基本的な図形を「パーツ」と呼んでいます。

オブジェクトには独自の座標系があり、これを「オブジェクト座標系」と呼んでいます。この座標系の大きさの尺度はグローバル座標系のそれと同じです。

パーツはオブジェクトに従属しており、オブジェクトの移動・回転に伴い、簡単な命令で適切な位置に再設定できますので、ユーザーは楽に物体を管理できます。

オブジェクト座標系の原点を0_objとし、そのオブジェクトの「中心」とします。
オブジェクト座標系におけるx,y,z軸をそれぞれ x_obj ,y_obj ,z_obj 軸と表現します。
オブジェクトの主観に関してですが、x軸の正の向きを「前」、y軸の正の向きを「左側」、z軸の正の向きを「天頂」と考えることを推奨します。
オブジェクト座標系における単位ベクトル(1,0,0),(0,1,0),(0,0,1)をそれぞれ →e_obj_x , →e_obj_y , →e_obj_z で表します。

オブジェクトには「オブジェクトの基本状態」という状態が存在します。これは、次の条件の連立により定義されます。

- オブジェクト座標系の原点がグローバル座標系の原点と重なる
- オブジェクト座標系のx,y,z軸の正の向きがそれぞれグローバル座標系のx,y,z軸の正の向きと一致する

つまり、基本状態では、オブジェクトはグローバルx,y平面を地面とし、グローバルx軸の正の向きを向いている状態です。

オブジェクトの状態は、位置と姿勢で決まります。
位置については、中心のグローバル座標を決めればよいです。
姿勢は、→e_obj_x , →e_obj_y , →e_obj_z で定まります。

姿勢を決定する方法は以下の3つを用意しています。

- オイラー角による表現
- axis-angleによる表現
- オブジェクト座標系単位ベクトルの手動設定

上2つには専用の命令が用意されています。

①＜ *オイラー角による表現* ＞

オブジェクトの基本状態をベースとして、x_obj,y_obj,z_obj軸回りに、この順にオブジェクトを回転させます。
x_obj ,y_obj ,z_obj軸回りにオブジェクト座標系における単位ベクトルを回転させます。

②＜ *axis-angle による表現* ＞

［構想］

→e_obj_x を定め、その回りにオブジェクトを回転させます。但し、回転前の基準は、オブジェクトが「最も立っている(つまり、→e_obj_z と z 軸の正の向きのなす角が最小となる)」状態を 0 とします。

→e_obj_x 回りにオブジェクト座標系における単位ベクトルを回転させればよいです。

[具体的な手順]

オブジェクトが見つめるターゲットの点を T とします。
まず ベクトル 0_obj→T を求め、その長さを 1 に調節したものを →e_obj_x とします。

次にオブジェクトの頭の向きを決めるために →e_obj_z を決めます。これが少し複雑です。

(ｉ) →e_obj_x が z_g 軸と平行でない場合

→e_obj_z と →e_g_z の内積が最大になるようにする。
詳しくは別資料を参照。

(ⅱ) →e_obj_x が z_g と平行である場合

→e_obj_z = →e_g_y とする。

③< *オブジェクト座標系単位ベクトルの手動設定* >

これは強力ですが、危険も伴います。
空間ベクトルを十分に理解している方が使用する方法です。

後述するオブジェクト構造体の要素(0, 3)～(0, 11)の値を直接書き換えることで、オブジェクトの姿勢を設定できます。
空間ベクトルを理解している方にとっては、これは**最も自由度の高い、優れた姿勢の表現方法**です。

但し、繰り返しますが、危険も伴います。甚だしく間違った値を指定すると最悪の場合内部エラーが発生するので、注意してください。

< オブジェクト構造体 >

オブジェクトの情報をまとめた配列変数を本モジュールでは「**オブジェクト構造体**」と呼んでいます。
これはオブジェクトを管理する上で非常に重要なものです。

構造体は 2 次元実数型配列変数です。不必要な要素は任意の値で埋めてかまいません。
以下に構造を示します。

< 要素(0, *) >

(0, 0), (0, 1), (0, 2) : オブジェクトの中心のグローバル x, y, z 座標

(0, 3), (0, 4), (0, 5) : →e_obj_x の x, y, z 成分
(0, 6), (0, 7), (0, 8) : →e_obj_y の x, y, z 成分

(0, 9), (0, 10), (0, 11)　:　　→e_obj_z の x, y, z 成分

(0, 3)～(0, 11)は通常は本モジュール側で計算して書き込む値であり、通常は人力で書き込むものではありません。オブジェクトの姿勢を高い自由度で設定したい場合に限り人力で入力します。

＜ 要素(1, *) ＞

(1, 0)　:　　そのオブジェクトの描画順序優先度クラス
(1, 1)　:　　オブジェクトの保有するパーツの総数
(1, 2)　:　　パーツの Z ソートスイッチ。(0.0 , 1.0) = (行わない, 行う)

＜ 要素(n, *) (※n≧2) ＞

要素(n, *)は(n-1)個目のパーツに関する情報を格納します。

要素(n, 0) はパーツの種類を記録します。
0～3 の整数でパーツの種類を指定します。
(0.0 , 1.0 , 2.0 , 3.0) = (点, 線分, 四角形平面, 球)
となります。

要素(n, 1) はパーツの可視/不可視の設定を記録します。
(0.0, 1.0)=(不可視, 可視)

要素(n, 2)～(n, 9)は現在使用されていません。放置してください。

パーツの種類に応じて、以降のデータ規則が変わります。下を御覧ください。

＜ 「点」 の場合 ＞++

各要素の内容は下の表のようになります。

要素	内容
(n, 10), (n, 11), (n, 12)	オブジェクト座標系座標
(n, 13), (n, 14), (n, 15)	グローバル座標 (※人力ではなく、モジュール側で計算して書き込む。)

(n, 16)以降の構造は点構造体に酷似しています。

要素	内容
(n, 16)	描画時の半径〔px〕
(n, 17)	ブレンド率
(n, 18)	光の効果の適用スイッチ。(0.0 , 1.0)=(適用しない, する)
(n, 19)	最低光量(※光の効果を適用する場合)
(n, 20)	画像の割り当ての有無。(0.0 , 1.0)=(無, 有)

(n, 21), (n, 22), (n, 23)	画像を割り当てない場合の、点の色の r, g, b 値。
(n, 24)	画像を割り当てる場合の、素材画像のあるウィンドウの ID
(n, 25), (n, 26)	画像を割り当てる場合の、素材画像の左上座標。
(n, 27), (n, 28)	画像を割り当てる場合の、素材画像の x, y サイズ。
(n, 29)	画像を割り当てる場合の、透過色の有効・無効。(0.0 , 1.0)=(無効, 有効)
(n, 30), (n, 31), (n, 32)	透過色有効の場合の、透過色の r, g, b 値。
(n, 33)	光沢度(0～255)

＜ 「線分」の場合 ＞++

各要素の内容は下の表のようになります。

要素	内容
(n, 10), (n, 11), (n, 12)	1 つ目の端点のオブジェクト座標系座標
(n, 13), (n, 14), (n, 15)	2 つ目 〃
(n, 16), (n, 17), (n, 18)	1 つ目の端点のグローバル座標 (※人力ではなく、モジュール側で計算して書き込む。)
(n, 19), (n, 20), (n, 21)	2 つ目 〃

(n, 22)以降の構造は線分構造体に酷似しています。

要素	内容
(n, 22)	描画時の太さ〔px〕
(n, 23), (n, 24), (n, 25)	色の RGB 値
(n, 26)	ブレンド率
(n, 27)	光の効果の適用スイッチ。(0.0 , 1.0)=(適用しない, する)
(n, 28)	最低光量(※光の効果を適用する場合)
(n, 29)	光沢度(0～255)

＜ 「四角形平面」の場合 ＞++

各要素の内容は下の表のようになります。

要素	内容
(n, 10), (n, 11), (n, 12)	1 つ目の頂点のオブジェクト座標系座標 (※4 点は環状になる順番で指定)
(n, 13), (n, 14), (n, 15)	2 つ目 〃
(n, 16), (n, 17), (n, 18)	3 つ目 〃

(n, 19), (n, 20), (n, 21)	4つ目〃
(n, 22), (n, 23), (n, 24)	1つ目の頂点のグローバル座標 (※人力ではなく、モジュール側で計算して書き込む。)
(n, 25), (n, 26), (n, 27)	2つ目〃
(n, 28), (n, 29), (n, 30)	3つ目〃
(n, 31), (n, 32), (n, 33)	4つ目〃

(n, 34)以降の構造は四角形構造体に酷似しています。

要素	内容
(n, 34)	ブレンド率
(n, 35)	光の効果の適用スイッチ。(0.0 ,1.0)=(適用しない, する)
(n, 36)	最低光量(※光の効果を適用する場合)
(n, 37)	描画モード (0.0 ,1.0 ,2.0)=(ワイヤーフレーム, 単色塗り潰し, テクスチャ張り付け)
(n, 38), (n, 39), (n, 40)	ワイヤーフレームまたは単色塗り潰しの場合の、色のr, g, b値。
(n, 41)	ワイヤーフレームの場合の、辺の描画時の太さ(px)
(n, 42)	テクスチャを張り付ける場合の、素材画像のあるウィンドウのID。
(n, 43), (n, 44), (n, 45), (n, 46)	テクスチャを張り付ける場合の、素材画像の頂点ABCDのx座標群。 ※ABCDは環状に並ぶように指定。
(n, 47), (n, 48), (n, 49), (n, 50)	〃y座標群。
(n, 51)	テクスチャを張り付ける場合の、透過色の有効・無効。 (0.0 ,1.0)=(無効, 有効)
(n, 52), (n, 53), (n, 54)	透過色有効の場合の、透過色のr, g, b値。
(n, 55)	光沢度(0～255)

＜ 「球」の場合 ＞++

各要素の内容は下の表のようになります。

要素	内容
(n, 10), (n, 11), (n, 12)	中心のオブジェクト座標系座標
(n, 13), (n, 14), (n, 15)	中心のグローバル座標 (※人力ではなく、モジュール側で計算して書き込む。)

(n, 16)以降の構造は球構造体に酷似しています。

要素	内容
(n, 16)	外径
(n, 17), (n, 18), (n, 19)	色のr, g, b値
(n, 20)	ブレンド率
(n, 21)	塗り潰しか中空か？ (0.0 ,1.0)=(中空, 塗り潰し)
(n, 22)	中空の場合の、内径
(n, 23)	光の効果の適用スイッチ。(0.0 ,1.0)=(適用しない, する)
(n, 24)	最低光量(※光の効果を適用する場合)

オブジェクト構造体を作っただけでは、描画処理に対して準備が十分ではありません。
必ず姿勢情報を決定してください。これはMSC3D_C_SetObj命令で行うか、要素(0,3)～(0,11)に手動で書き込むことで可能です。
その後も、オブジェクトの姿勢を変化させる場合はその都度姿勢情報を更新する必要があります。

同じ形状のオブジェクトを大量に管理する場合、個々のために構造体を用意するのではなく、同じ構造体を使って言わば「影分身」させることで、管理が非常に楽になります。
例えば、画面更新を停止した後に単一の構造体の位置と姿勢の情報のみを高速で何回も変更して描画を繰り返し、一通り済んだ後で画面更新を行えば、一つの構造体だけで数十個でも数百個でも好きなだけオブジェクトを表示できます。この場合、ユーザーはオブジェクトの位置情報のみを独自に管理しておくだけで、オブジェクト構造体はたった１つで済みます。

＜ オブジェクトの撮影に関して ＞

隠面処理は「**画家のアルゴリズム**」で行います。これは世の中では最も単純で非効率だと言われている方法です。
本モジュールでパーツを描画するおおまかな手順は以下のようになります。

1. オブジェクトについて、描画順序優先度クラスが小さいものから順番に並び、かつ、同じ優先度クラスのオブジェクトについてはその中心と眼点との距離が遠いものから順番に並ぶように、リストを作成する。

2. そのリストに従って、遠いオブジェクトから順に描画する。その際、設定に応じて、描画対象のオブジェクトに属するパーツは眼点から遠いものから順に描画する。

つまり、パーツのＺソートはあくまで１つのオブジェクトに属するパーツ同士で行われるのであり、すべてのオブジェクトのパーツを一挙にソートするわけではありません。
従って、オブジェクト同士が噛み合っていたり、巨大で長細いオブジェクトのすぐ背後に小さなオブジェクトが隠れていたりするような状況では不自然な描画になりますが、ご了承ください。
処理速度を考慮した結果この制約を設けました。いつかもっと高速に処理できるようになった時は考え直すかもしれません。

尚、パーツのＺソートを行うかどうかについては、各オブジェクト毎に個別に設定できます。例えば床などのオブジェクトで、同一平面上にタイルが並んだような構造になっているものは、タイル同士でのＺソートは不要で、その分処理時間を短縮できます。

＜ 描画順序優先度クラス ＞

これはオブジェクトに割り当てる階級で、1～10 の 10 段階があります。
数が小さいものほど先に描画が行われます。
オブジェクト同士の Z ソート時に、この階級が考慮され、同じ階級のもの同士でソートが行われます。

例えば戦闘機ゲームであれば、遙か彼方にあってルール的に絶対に到達しないような場所にあるものは優先度クラスを 1 にして、敵機やミサイルを 2、自機を 3 にしておけば普通は大丈夫です。

＜ 命令・関数リファレンス ＞

命令一覧

- MSC3D_C_init
- MSC3D_C_SetObj
- MSC3D_C_ZSortObj
- MSC3D_C_DrawObj

注意

速度重視のため、不正なパラメータに対するチェックはほとんど行っていません。
不正なパラメータが渡されると、内部エラーとなる場合があります。（例えば、0 除算ラー。）

命令	説明
MSC3D_C_init	モジュールの初期化 初めて使用する前に必ず行ってください。 ［書式］ MSC3D_C_init
MSC3D_C_SetObj	オブジェクトの位置と姿勢の決定。 オブジェクトの単位ベクトルとパーツのグローバル座標を決定します。 ［書式］ MSC3D_C_SetObj array1 ,Opt1 ,a,b,c,d array1 ：　オブジェクト構造体(※中

	心のグローバル座標は手動で書き込んでおくこと) Opt1　：　姿勢の決定手段。 0 ： オイラー角 1 ： axis-angle 2 ： 手動設定 *＜ オイラー角の場合 ＞* a, b, c にそれぞれ x_obj 軸, y_obj 軸, z_obj 軸まわりの、基本状態からの回転角度を指定。 回転はこの順に行われることに留意。 d の値は任意。 *＜ axis-angle の場合 ＞* a, b, c にそれぞれオブジェクトが見つめる点のグローバル x, y, z 座標を指定。 その点を見つめた後の、x_obj 軸まわりの回転角度を d で指定。 ＜ 手動設定の場合 ＞ 既に構造体に書き込まれている →e_obj_x ,→e_obj_y ,→e_obj_z に従ってパーツのグローバル座標を決定する。 a, b, c, d の値は任意。
MSC3D_C_ZSortObj	オブジェクトの Z ソート オブジェクトを描画すべき順番を調べます。 [書式] MSC3D_C_ZSortObj array1 ,n ,result1,result2 array1 ：　ソート対象のオブジェクトの情報を格納した実数型配列変数(※構造は後述) n　：　ソート対象のオブジェクトの個数 result1 ： result2 ：　ソート結果の ID リスト result1 と連動した距離リスト。眼点とオブジェクトの中心間の距離が記録される。 *＜ 「array1」変数の構造 ＞*

	要素(k,0),(k,1),(k,2) (k=0,1,2,･･･)に任意のオブジェクトの中心のグローバル x,y,z 座標を格納。 要素(k,?)に情報を格納したオブジェクトの ID を便宜的に k とする。 要素(k,3)に描画順序優先度クラスの値を格納。 ＜ *「result1」整数型配列変数の構造* ＞ 要素(0)～(n-1)までに、眼点からの距離が大きい順にオブジェクトの ID が記録される。 オブジェクトの描画はこの順番で行えばよい。 ＜ 「result2」実数型配列変数の構造 ＞ 「result1」変数と同じ並びで、オブジェクトの中心と眼点との距離が格納される。
MSC3D_C_DrawObj	オブジェクトの描画 [書式] MSC3D_C_DrawObj array1 array1 ： オブジェクト構造体