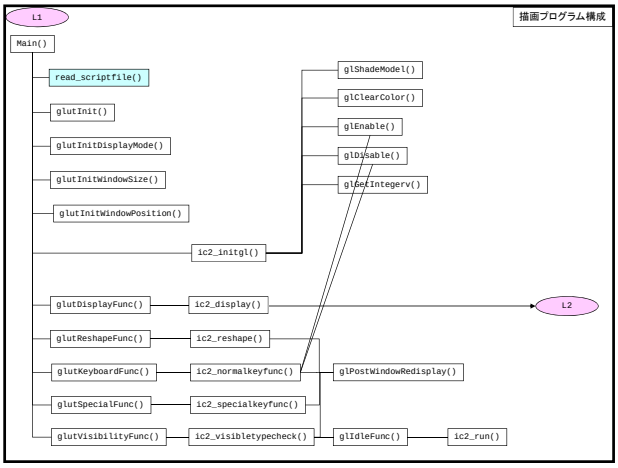
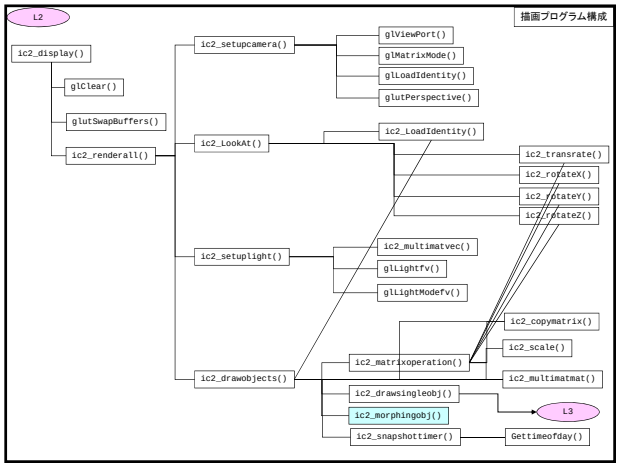
計算機序論2
2009/11/09
亀田能成

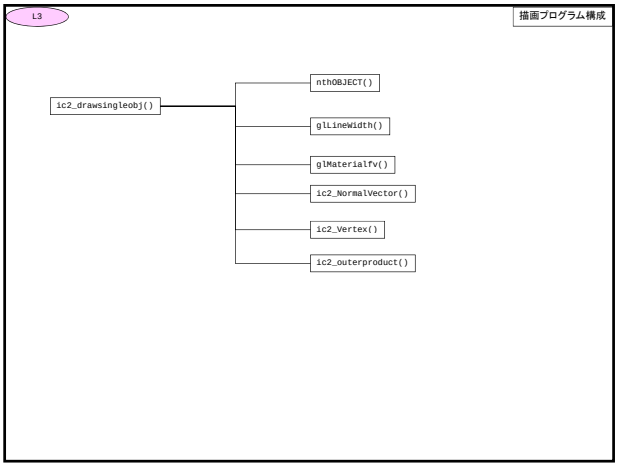
- 課題DはWWW参照

プログラム構成

特に描画部







アニメーション

コマ送りの実装

アニメーション

- 動作や形が少しずつ異なる多くの絵や人形を一齣(ひとこま)ずつ撮影し、映写した時に画像が連続して動いて見えるようにするもの。ビデオ-レコーダーによるものやコンピューターグラフィックスを応用するものもある。アニメ。動画。

三省堂提供「大辞林 第二版」

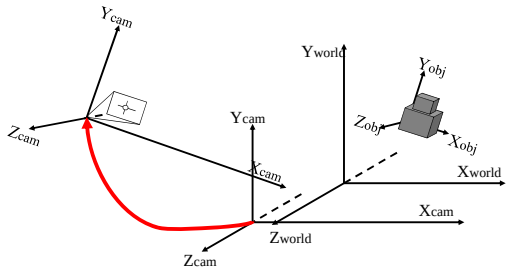
CGを用いたアニメーション作成

アニメーションの実現手段

- カメラの移動
  - ic2LookAt()
- CG物体の移動
  - ic2drawobjects()
- CG物体の変形（モーフィング【後述】）
  - ic2_morphingobj()

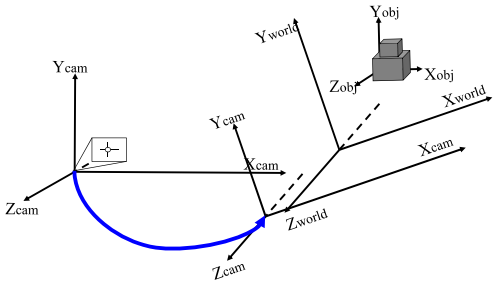
カメラの移動によるアニメーション

- 視点の移動による見え方の変化



カメラの移動によるアニメーション

- 実際は、世界･物体座標系が移動している

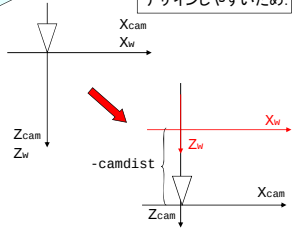


カメラの位置･姿勢設定
ic2_LookAt(float *mvm)

```
void ic2_LookAt (float *r) {
    float t[16];      // 一時的な幾何変換行列
    float k[16];      // 蓄積行列

    ic2_LoadIdentity(k);  // k[]を単位行列で初期化

    // Z軸上での移動
    ic2_translate(t, 0, 0, -camdist);
    ic2_multmatmat(k, k, t);

    // 上下方向の回転
    ic2_rotateX(t, camtheta_v);
    ic2_multmatmat(k, k, t);

    // 左右方向の回転
    ic2_rotateY(t, -(camtheta_h - 90.0));
    ic2_multmatmat(k, k, t);

    // 最終的に得られた蓄積行列を返す
    ic2_copymatrix(r, k);
}
```

camdistだけカメラを引く
（実際は、世界を遠のける）

世界座標の原点付近に物体を置いた方が、デザインしやすいため.

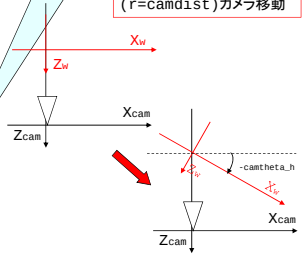


カメラの位置･姿勢設定
ic2_LookAt(float *mvm)

```
void ic2_LookAt (float *r) {
    float t[16];      // 一時的な幾何変換行列
    float k[16];      // 蓄積行列

    ic2_LoadIdentity(k);  // k[]を単位行列で初期化

    // Z軸上での移動
    ic2_translate(t, 0, 0, -camdist);
    ic2_multmatmat(k, k, t);

    // 上下方向の回転
    ic2_rotateX(t, camtheta_v);
    ic2_multmatmat(k, k, t);

    // 左右方向の回転
    ic2_rotateY(t, -(camtheta_h - 90.0));
    ic2_multmatmat(k, k, t);

    // 最終的に得られた蓄積行列を返す
    ic2_copymatrix(r, k);
}
```

Yw軸周りに-camtheta_h回転
（実際は、逆方向に世界が回転）

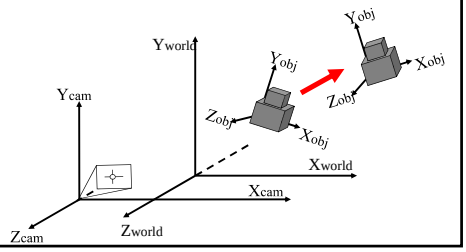
物体の周りを周回する
(r=camdist)カメラ移動



アニメーション

物体の移動によるアニメーション

- 物体の移動による見え方の変化
  - カメラ～世界座標系の幾何関係は一定
  - 世界～物体座標系の幾何関係が変化
  - 課題プログラムのスクリプトで実行されるアニメーション



アニメーション

スクリプトによるアニメーションの実行

- 世界～物体座標系の移動変換
- 移動情報（平行・回転）の与え方
  1. コマ毎に、(tx,ty,tz)(rx,ry,rz)を関数に入力
  2. 全体の移動情報を最初に与え、それをコマ数で分割した移動情報（1コマの移動情報）を関数に入力
- 課題プログラムでは、2. を採用

アニメーション

スリプト中のアニメーション記述1

A: [animation]　一つ目のアニメーション

A ObjectID step interval tx ty tz rx ry rz sx sy sz

世界座標系に対する物体座標系の移動・回転・伸縮の操作を示す
　tx,ty,tz: 平行移動
　rx,ry,rz: 回転（軸回り）[degree]
　sx,sy,sz: 拡大縮小
　step：アニメーションにかける回数
　interval: アニメーション一回の時間
同時指定した場合の演算は、プログラム上の出現順序で 平行移動、回転（X,Y,Z軸周りの順）、拡大縮小の通り

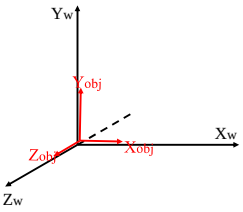
（例）A 1 200 0 0.0 0.0 0.0 0 0 360 1.0 1.0 1.0
　　200コマかけて、360度回転

アニメーション

スリプト中のアニメーション記述1

A: [animation]　一つ目のアニメーション

A ObjectID step interval tx ty tz rx ry rz sx sy sz



アニメーション開始時に、
蓄積変換行列C　を単位行列に初期化する

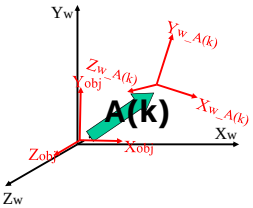
$$C = I$$

アニメーション

スリプト中のアニメーション記述1

A: [animation]　一つ目のアニメーション

A ObjectID step interval tx ty tz rx ry rz sx sy sz



アニメーションの進行度k(0≦k≦1)の場合の座標変換は、

$$\begin{bmatrix} X_{w\_A(k)} \\ Y_{w\_A(k)} \\ Z_{w\_A(k)} \\ 1 \end{bmatrix} = C\,T(k)Rx(k)Ry(k)Rz(k)S(k) \begin{bmatrix} X_{obj} \\ Y_{obj} \\ Z_{obj} \\ 1 \end{bmatrix}$$

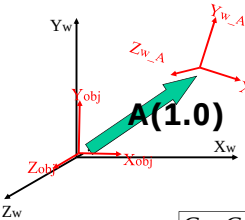
蓄積変換行列　C(=I)
平行移動変換行列　T(k)
回転移動変換行列　Rx(k), Ry(k), Rz(k)
スケーリング　S(k)

アニメーション

スリプト中のアニメーション記述1

A: [animation]　一つ目のアニメーション

A ObjectID step interval tx ty tz rx ry rz sx sy sz



アニメーションの終了時の、平行移動変換
T(1.0)Rx(1.0)Ry(1.0)Rz(1.0)
S(1.0)を蓄積変換行列C　に蓄積.

$$\begin{bmatrix} X_{w\_A} \\ Y_{w\_A} \\ Z_{w\_A} \\ 1 \end{bmatrix} = C\,T(1.0)Rx(1.0)Ry(1.0)Rz(1.0)S(1.0) \begin{bmatrix} X_{obj} \\ Y_{obj} \\ Z_{obj} \\ 1 \end{bmatrix}$$

$$C = C\,T(1.0)Rx(1.0)Ry(1.0)Rz(1.0)S(1.0)$$

代入

スクリプト中のアニメーション記述2

C: [animation Continued]　二つ目以降のアニメーション

C ObjectID step interval tx ty tz rx ry rz sx sy sz

世界座標系に対する物体座標系の移動・回転・伸縮の操作に加え物体の変形を表現する

(1) 移動・回転・伸縮 A:と同じ

(2) 変形 ObjectIDが前のAnimation（A行 or C行）と同じ場合は、アニメーション操作を表す

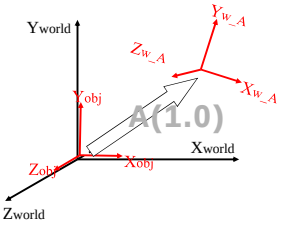
(※) 変形 ObjectIDが前のAnimation（A行 or C行）と異なる場合は、変形操作を表す→ モーフィングで再出

スクリプト中のアニメーション記述2

C: [animation Continued]　二つ目以降のアニメーション

C ObjectID step interval tx ty tz rx ry rz sx sy sz

`ObjectID`がAの行と同じ場合：アニメーションの継続



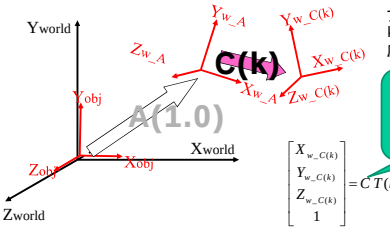
アニメーション開始時、現在の蓄積変換行列Cをそのまま利用（＝アニメーションの継続）

スリプト中のアニメーション記述2a

C: [animation Continued]　二つ目以降のアニメーション

C ObjectID step interval tx ty tz rx ry rz sx sy sz

`ObjectID`がAの行と同じ場合：アニメーションの継続



アニメーションの進行度 `k(0.0≦k≦1.0)`の場合の座標変換はA行の場合と同様.

一つ前のAを示す行列がここに格納されている

$$\begin{bmatrix} X_{w\_C(k)} \\ Y_{w\_C(k)} \\ Z_{w\_C(k)} \\ 1 \end{bmatrix} = C\,T(k)Rx(k)Ry(k)Rz(k)S(k) \begin{bmatrix} X_{obj} \\ Y_{obj} \\ Z_{obj} \\ 1 \end{bmatrix}$$

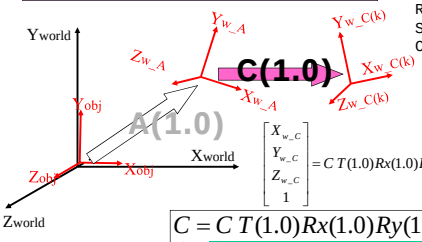
スリプト中のアニメーション記述2a

C: [animation Continued]　二つ目以降のアニメーション

C ObjectID step interval tx ty tz rx ry rz sx sy sz

`ObjectID`がAの行と同じ場合：アニメーションの継続

アニメーションの終了時の、移動変換`T(1.0)Rx(1.0)Ry(1.0)Rz(1.0)S(1.0)`を蓄積変換行列Cに蓄積.



$$\begin{bmatrix} X_{w\_C} \\ Y_{w\_C} \\ Z_{w\_C} \\ 1 \end{bmatrix} = C\,T(1.0)Rx(1.0)Ry(1.0)Rz(1.0)S(1.0) \begin{bmatrix} X_{obj} \\ Y_{obj} \\ Z_{obj} \\ 1 \end{bmatrix}$$

$$C = C\,T(1.0)Rx(1.0)Ry(1.0)Rz(1.0)S(1.0)$$

さらに代入

アニメーションデータの構造体

1回分のアニメーションのための構造体

```
struct ic2ANIME {
  int type; // [A] new anime, [C] anime continued
  int id; // object ID
  int step; // intermediate snapshots to morph
  int interval; // interval between intermediate snapshots in milliseconds
  float tx; // translate along x axis
  float ty; // translate along y axis
  float tz; // translate along z axis
  float rx; // rotate around x axis
  float ry; // rotate around y axis
  float rz; // rotate around z axis
  float sx; // scale in x
  float sy; // scale in y
  float sz; // scale in z
  struct ic2ANIME *next;  //次のアニメーションへのポインタ
};
```


アニメーション処理

- 変換情報は構造体animnowに記録
  struct ic2ANIME *animnow
  連続アニメーションはanimnow->next で連結
- 連続するアニメーション操作では、（アニメーションの始点からの）相対的な変換情報が与えられる
  float cumulatedmatrix[16]　に蓄積
- 現在までの表示枚数に応じてアニメーションの進行度合い（大きさ）が決まる
  r = (float)snapshotnum / animnow->step;

→ic2_matrixoperation(mvm_work, animnow, r, mvm_work);

アニメーション

ic2_matrixoperation()

アニメーション中の1コマ(A(k)ないしC(k))の座標変換

```
void ic2_matrixoperation
(
float *rmat,
// アニメーション終了のモデルビュー行列

struct ic2ANIME*animnow,
// 処理するアニメーション構造体(平行･回転移動･スケールの情報が
    格納されている)

float r,
// アニメーションの進度(0.0-1.0) ※0.0はstep=0(アニメーションなし)
    となる特殊値

float *basematrix
// アニメーション開始時のモデルビュー行列
)
```

モーフィング

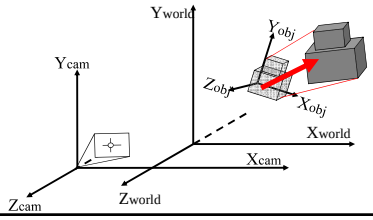
物体の変形:モーフィング

今回は線形補間で単純に実装

モーフィング

物体の変形によるアニメーション

- 物体の変形による見え方の変化
  - カメラ～世界座標系の幾何関係は一定
  - 世界～物体座標系の幾何関係は一定(独立)
  - 課題プログラムのスクリプトで実行されるアニメーション



モーフィング

モーフィング

- ある形状から別の形状へ徐々に変化していく様子を動画で表現するために、その中間を補うための画像を作成すること。(中略)前後のコマの画像の要素から、中間の画像を作り出すことを何度も行なうと、滑らかに変化するモーフィングの動画(CG物体の変形によるアニメーション)が得られる。

(e-words)

モーフィング

線形補間によるモーフィング

- 始点A(Xa,Ya,Za)と終点B(Xb,Yb,Zb)をr:1-rに内分(内挿)した点C(Xc,Yc,Zc)

$$\begin{pmatrix} Xc \\ Yc \\ Zc \end{pmatrix} = \begin{pmatrix} Xa \\ Ya \\ Za \end{pmatrix} \times r + \begin{pmatrix} Xb \\ Yb \\ Zb \end{pmatrix} \times (1-r)$$

- 色情報の内挿も忘れずに

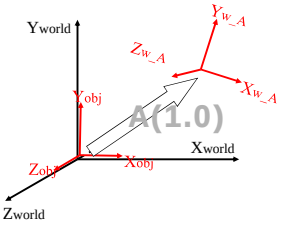
アニメーション

スリプト中のアニメーション記述2b

C: [animation Continued]　二つ目以降のアニメーション

C ObjectID step interval tx ty tz rx ry rz sx sy sz

ObjectIDが前の行(例:A行)と異なる場合:モーフィングの実行



アニメーション開始時、現在の蓄積変換行列Cをそのまま利用
(=アニメーションの継続)

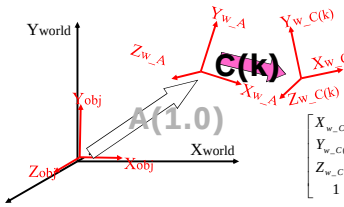
アニメーション

スリプト中のアニメーション記述2b

C: [animation Continued]　二つ目以降のアニメーション

C ObjectID step interval tx ty tz rx ry rz sx sy sz

`ObjectID`が前の行(例:A行)と異なる場合:モーフィングの実行



アニメーションの進行度
`k(0.0≦k≦1.0)`の場合の座標変換は、A行の場合と同様.

$$\begin{bmatrix} X_{w\_C(k)} \\ Y_{w\_C(k)} \\ Z_{w\_C(k)} \\ 1 \end{bmatrix} = C\,T(k)Rx(k)Ry(k)Rz(k)S(k) \begin{bmatrix} X_{obj} \\ Y_{obj} \\ Z_{obj} \\ 1 \end{bmatrix}$$

同時に、`ic2_morphingobj()`内で、物体の色と形状(頂点の位置)の内挿処理が行われる(=モーフィング).

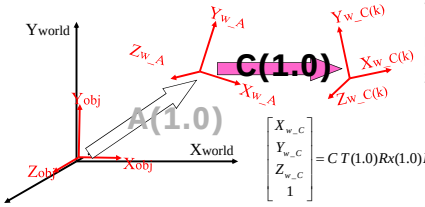
アニメーション

スリプト中のアニメーション記述2b

C: [animation Continued]　二つ目以降のアニメーション

C ObjectID step interval tx ty tz rx ry rz sx sy sz

`ObjectID`が前の行(例:A行)と異なる場合:モーフィングの実行



アニメーションの終了時の移動変換`T(1.0) Rx(1.0) Ry(1.0) Rz(1.0) S(1.0)`を蓄積変換行列Cに蓄積.

$$\begin{bmatrix} X_{w\_C} \\ Y_{w\_C} \\ Z_{w\_C} \\ 1 \end{bmatrix} = C\,T(1.0)Rx(1.0)Ry(1.0)Rz(1.0)S(1.0) \begin{bmatrix} X_{obj} \\ Y_{obj} \\ Z_{obj} \\ 1 \end{bmatrix}$$

$$C = C\,T(1.0)Rx(1.0)Ry(1.0)Rz(1.0)S(1.0)$$

モーフィング

モーフィング処理用関数

- float ic2_li (float a, float b, float r)
  - aとbの内分値を戻り値とする関数
- void ic2_liPOINT (struct ic2POINT *i, struct ic2POINT a, struct ic2POINT b, float r)
  - 3次元点aと点bの内分点iを計算する関数

アニメーション

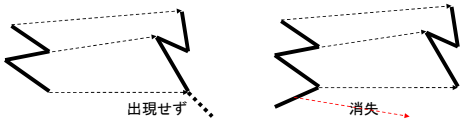
モーフィング前後のCG要素数

- モーフィングの前後で物体を構成するCG要素(線や面)の数が異なる場合どうなる?
- 前後で対応が取れる(数が少ない方の要素数)だけ、モーフィングが行われる
  - 線形リストがつながっているだけ
- 新しく要素が出現するようなモーフィングをしたい場合は、前のCG物体に冗長な要素を持たせる.

アニメーション

モーフィング前後のCG要素数

- 前後で対応が取れる(数が少ない方の要素数)だけ、モーフィングが行われる



- 新しく要素が出現するようなモーフィングをしたい場合は、前のCG物体に冗長な要素を持たせる.