三次元物体の投影図表示

１．立体の投影

３次元立体形状を２次元平面へ投影する方法には、

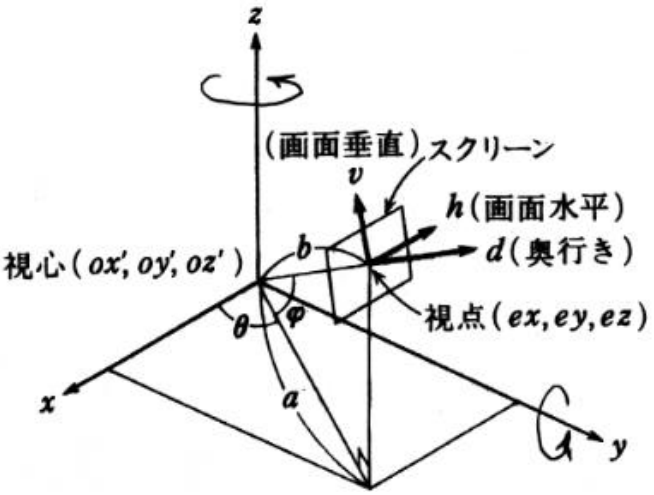
・平行投影（軸測図）

・中心投影（透視図；遠近感を表すもの）

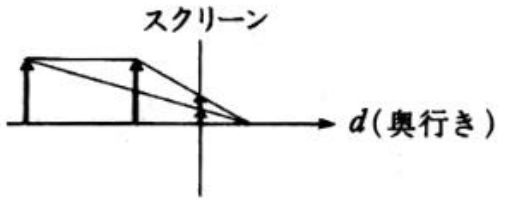
がある。立体形状は任意の平面へ投影できるが、その投影を確定するためには以下の情報が必要となる。

・視点（どこから見るか）の位置：　$\boldsymbol{e}$（ *ex, ey, ez* ）

・視心（どこを見るか）　の位置：　$\boldsymbol{o}'$ (*ox', oy', oz'*)



(a)任意方向への投影変換



(b)透視変換　　（図は参考文献 4 より）

１）ベクトル$\overrightarrow{o'e}$に垂直な面上のスクリーン座標への投影変換

視点位置を原点とする座標(*h, v, d*)を導入

d 軸は$\overrightarrow{o'e}$方向

h 軸は$\overrightarrow{o'e}$に垂直なスクリーン水平方向

v 軸は$\overrightarrow{o'e}$に垂直なスクリーン垂直方向

下記変換１～３により、物体が置かれている座標系 (*x, y, z*) 上の点を(*h, v, d*)系で表す。

変換１．原点を $\boldsymbol{e}$ に平行移動

変換２．z 軸まわりにθだけ座標軸を回転（*x* 軸が *d* 軸の *xy* 平面への投影方向に一致）

変換３．y 軸まわりに－φだけ座標軸を回転（*x* 軸が *d* 軸に一致）

（変換３）　　（変換２）　（変換１）

$$\begin{pmatrix} d \\ h \\ v \end{pmatrix} = \begin{bmatrix} \cos f & 0 & \sin f \\ 0 & 1 & 0 \\ -\sin f & 0 & \cos f \end{bmatrix} \begin{bmatrix} \cos q & \sin q & 0 \\ -\sin q & \cos q & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x-ex \\ y-ey \\ z-ez \end{pmatrix}$$

$$= \begin{bmatrix} \cos f \cos q & \cos f \sin q & \sin f \\ -\sin q & \cos q & 0 \\ -\sin f \cos q & -\sin f \sin q & \cos f \end{bmatrix} \begin{pmatrix} x-ex \\ y-ey \\ z-ez \end{pmatrix}$$

ただしベクトル $\overrightarrow{o'e}$ の方向を表わす角 θ,φ の正弦、余弦の値は以下のように求められる。

$$a = \sqrt{(ex - ox')^2 + (ey - oy')^2}$$
$$b = \sqrt{(ex - ox')^2 + (ey - oy')^2 + (ez - oz')^2}$$
$$\sin q = (ey - oy')/a, \quad \cos q = (ex - ox')/a, \quad \sin f = (ez - oz')/b, \quad \cos f = a/b$$

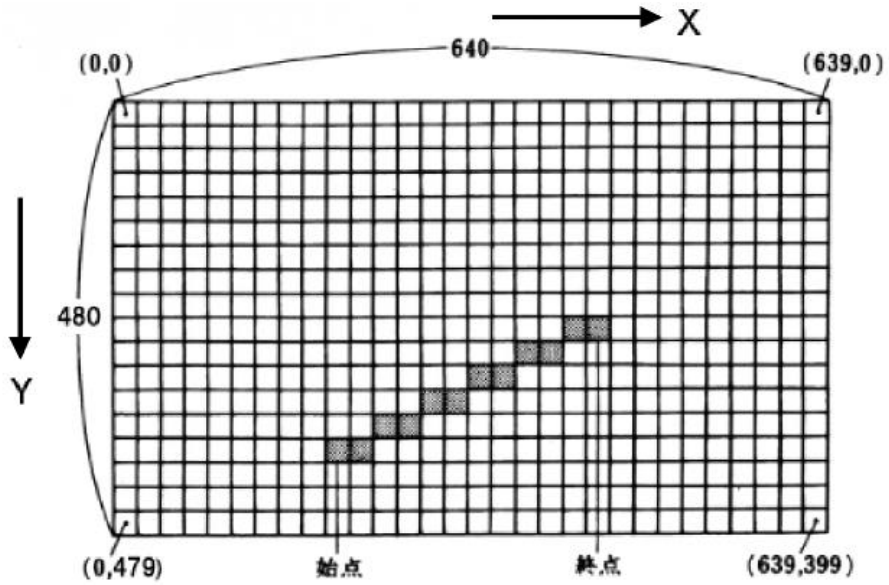
(a) 平行投影の場合 (*h, v*) をスクリーン座標とする。

(b) 中心投影の場合 (*h', v'*) [*h' = h/(-d), v' = v/(-d)*] をスクリーン座標とする。
（遠くにあるほど小さく見える）

2）スクリーン座標 (*h, v*)の実画面座標 (*X, Y*) への変換

実画面座標 (*X, Y*)

パーソナル・コンピュータの画面は下図のように pixel（通常横 640x 縦 480 個）に分割されており、この pixel に色をつけることにより図形を表示することができる。
各 pixel は実画面座標（*X, Y*）により指定される。たとえば図のような画面に始点と終点の座標値を示せば、色をつけるべき pixel がわかるので線分を描くことができる。



pixel による表示と実画面座標　　（図の一部は参考文献 4 より）

投影図形がパソコン画面内に収まるように実画面座標に変換

実画面座標の最大・最小値　　：*Xmax, Xmin, Ymax, Ymin*
物体を表わすスクリーン座標の最大・最小値：*hmax, hmin, vmax, vmin*

パソコン画面の枠に ΔX の空白ができるようにスケールを決定

$Xscale = (Xmax - Xmin - 2\Delta X) / (hmax - hmin)$

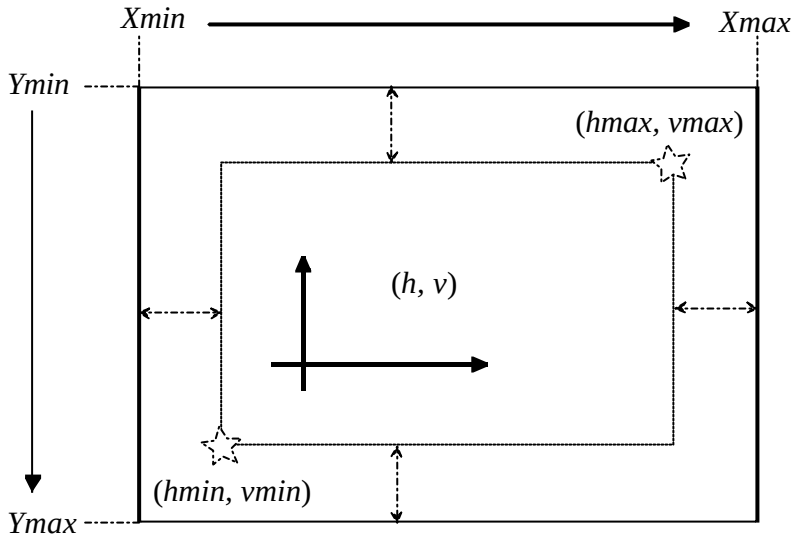
$Yscale = (Ymax - Ymin - 2\Delta X) / (vmax - vmin)$

$Fscale = min(Xscale, Yscale)$

変換式（*Y* と *v* は逆向きであることに注意）

$X = Xmin + \Delta X \quad + \quad Fscale\ (\ h\ -\ hmin\ )$

$Y = Ymin + \Delta X \quad + \quad Fscale\ (\ vmax\ -\ v\ )$

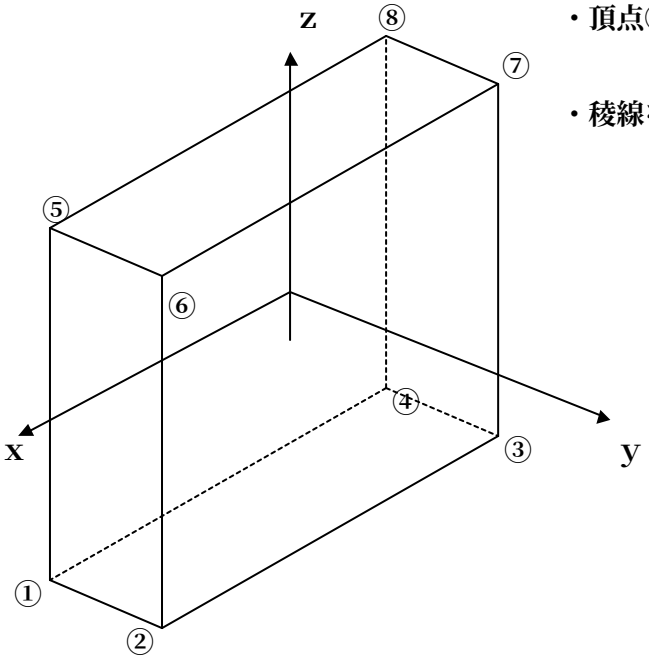


２．立体の表示方法

立体の形状定義

ワイヤモデル　　：頂点座標を与え、稜線（ワイヤ）を定義
サーフェスモデル：ワイヤモデルにおいてワイヤで囲まれた部分に面（サーフェス）を定義
ソリッドモデル　：サーフェスモデルにおいてサーフェスのいずれの側に実体が存在するかを定義

例１）ワイヤモデルによる形状定義



・頂点①～⑧の座標を定義

・稜線を頂点番号で定義

①－②，②－③，③－④，④－①，
⑤－⑥，⑥－⑦，⑦－⑧，⑧－⑤，
①－⑤，②－⑥，③－⑦，④－⑧，

例２）サーフェスモデルによる形状定義

・頂点①～⑧の座標を定義

・面を頂点番号で定義

①－④－③－②,
①－⑤－⑧－④,
④－⑧－⑦－③,
①－⑤－⑥－②,
⑤－⑧－⑦－⑥,
②－⑥－⑦－③,

簡単な陰れ線消去方法

最も原始的な隠れ線消去の方法は、最も視点から離れている所から前方に向かって順番に面要素を塗り潰して描いていくというやり方である。

ワイヤモデルの形状定義データ例

```
8                          頂点数
1    10  -5 –15             頂点番号　頂点座標 x, y, z
2    10   5  -15
3   -10   5  -15
4   -10  -5  -15
5   10  -5    15
6   10   5    15
7   -10   5    15
8   -10  -5    15
12                         稜線の数
1   2                      頂点番号による稜線の定義
2   3
3   4
4   1
5   6
6   7
7   8
8   5
1   5
2   6
3   7
4   8
```

今期（2001年度後期）のプリント参考文献

1．若林茂雄著：初級C言語　Introduction C Language、株式会社アイテック、1997年。
2．椋田實：はじめてのC、改訂第3版［ANSI C対応］、技術評論社、1993年。
3．平林正英：ANSI C言語辞典、技術評論社。
4．永野三郎、長島忍：Pascal 入門―TURBO Pascal演習―、第2版、東京大学出版会、1994年。
5．Michael T. Heath, Scientific Computing: An Introductory Survey, second edition, McGraw-Hill Companies, Inc., 2002.
6．Dr. Robert Bickel 講義録

課題（三次元物体の投影図表示）

ワイヤーモデルの平行投影／中心投影を行うプログラム（配布）を完成させよ。

関数　virtual_screen：物体の３次元座標を２次元スクリーン座標に変換

===> 完成させること。

関数　real_gamen　：２次元スクリーン座標を実画面座標に変換（物体が画面内に収まるように）

===> 完成させること。

・まず４頁に記したワイヤモデルの形状定義データ例（配布；ファイル名 “wire.dat” ）で示される物体の投影図を画面上に表示せよ。

・次に任意の形状データを作成（簡単なものでよい）してその投影図を表示せよ。

――――――――――――――――――――――――

提出期限：　２月１５日（金）

提出物：　課題プログラム　　ファイル名は　p○○××e10.c とすること

形状データファイル　　ファイル名は　p○○××e10.txt とすること

○○は学籍番号の上２桁の数字

××は学籍番号の下２桁の数字

＊ プログラムの第１行目には、コメントで（ /*　*/ でくくって）、学籍番号と名前を記述すること

＊ 形状データファイル p○○××e10.txt は、各自が作成した形状データを定義したもの

提出形態：　E-mail に上記提出ファイルを添付すること

・あて先　　cp2-69@cc.tuat.ac.jp

・題　　cp2ex10

・本文　　感想、学籍番号、名前を記述

配布プログラム

```
/* ワイヤー・フレーム・モデルの平行投影・中心投影表示 */
                        /* ヘッダー・ファイルの取り込み */
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <gucc/gucc.h>

#define PSIZE 100
                        /* グラフィック関数のプロトタイプの宣言 */
void G_INIT(void);                  /* 画面初期化関数 */
void G_CLS(void);                   /* 画面消去関数 */
void G_LINE(int, int, int, int);    /* 線分を引く関数 */

                                    /* 関数プロトタイプの宣言 */
void f_input(void);
void trans_mat(void);
void virtual_screen(void);
void real_gamen(void);
void wire_draw(void);

struct Szahyo{                      /* 構造体である Szahyo 型の宣言 */
       float h;
       float v;
};
struct Gzahyo{                      /* 構造体である Gzahyo 型の宣言 */
       int X;
       int Y;
};
                                                /* 外部変数の宣言 */
struct Szahyo screen[PSIZE];
struct Gzahyo gamen[PSIZE];

float eye[3],dpt[3],mat[3][3],xyz[PSIZE][3];
int   nv,ns,ipers;
int   edge[PSIZE][2];

void main(void)
{
  f_input();                        /* 物体の形状データをファイルから読み込む */

  trans_mat();                /* 視点と視心をコンソール入力し、変換行列を計算 */

  virtual_screen();                 /* 物体の３次元座標をスクリーン座標へ変換 */
```

```
  real_gamen();                    /* 物体のスクリーン座標を実画面座標へ変換 */

  wire_draw();                          /* ワイヤ・モデルを描く */

}

void f_input(void)                 /* 物体の形状データをファイルから読み込む */
{
  char  fname[20];
  FILE  *fin;
  int   i,xdum;

  printf("データ・ファイル名  ===>  ");     /* ファイル名の入力を促す */
  scanf("%s", fname);                       /* ファイル名を入力 */
                              /* データ・ファイルのオープンとエラー処理 */
  if( (fin=fopen(fname,"r")) == NULL){
      printf("ファイルがオープンできません。\n");
      exit(1);
  }
                                      /* ３次元物体の頂点座標を読み込む */
  fscanf(fin,"%d",&nv);                                   /* 個数 */
  for(i=1; i<=nv; i++){                                   /* 座標 */
    fscanf(fin,"%d %f %f %f ",&xdum,&xyz[i][0],&xyz[i][1],&xyz[i][2]);
  }
                                                /* 稜線定義を読み込む */
  fscanf(fin,"%d",&ns);                             /* 個数 */
  for(i=1; i<=ns; i++){                             /* 稜線 */
    fscanf(fin,"%d %d",&edge[i][0],&edge[i][1]);
  }

  fclose(fin);                                    /* ファイル・クローズ */
}

void trans_mat(void)       /* 視点と視心をコンソール入力し、変換行列を計算 */
{
  float a,b,sint,cost,sinp,cosp;
                        /* 軸則図か透視図か？及び視点と視心をコンソール入力 */
  printf("1.軸測図  2.透視図 ? \n");
  scanf("%d",&ipers);
  printf("視点の位置(x y z) ? \n");
  scanf("%f %f %f",&eye[0],&eye[1],&eye[2]);
  printf("視心の位置(x y z) ? \n");
  scanf("%f %f %f",&dpt[0],&dpt[1],&dpt[2]);

  a=(eye[0]-dpt[0])*(eye[0]-dpt[0])+(eye[1]-dpt[1])*(eye[1]-dpt[1]);/* aの２乗 */
  b=a+(eye[2]-dpt[2])*(eye[2]-dpt[2]);                              /* bの２乗 */
```

```
  a=(float)sqrt(a);                    /* a */
  b=(float)sqrt(b);                    /* b */
  sint=(eye[1]-dpt[1])/a;              /* sin θ*/
  cost=(eye[0]-dpt[0])/a;              /* cos θ*/
  sinp=(eye[2]-dpt[2])/b;              /* sin φ*/
  cosp=a/b;                            /* cos φ */
                                         /* 以下は3行3列の変換行列の要素 */
  mat[0][0]=cosp*cost;  mat[0][1]=cosp*sint;  mat[0][2]=sinp;
  mat[1][0]=-sint;      mat[1][1]=cost;       mat[1][2]=0.0;
  mat[2][0]=-sinp*cost; mat[2][1]=-sinp*sint; mat[2][2]=cosp;
}

void virtual_screen(void)    /* 物体の３次元座標(x,y,z)を２次元スクリーン座標(h,v)へ変換 */
{
  int   i,j,k;
  float cv[3];

  /* eye[0], eye[1], eye[2]             :視点の座標 (x,y,z) */
  /* mat[i][j], i=0,1,2,  j=0,1,2       :変換行列 */
  /* xyz[k][0], xyz[k][1], xyz[k][2], k=1,..,nv：３次元物体の頂点座標 (x,y,z) (nv個) */
  /* cv[0], cv[1], cv[2]                 ：３次元物体のスクリーン座標 (d,h,v) */
  /* screen[k].h, screen[k].v, k=1,..,nv:物体の最終的な２次元スクリーン座標 (h,v) */

  for(k=1; k<=nv; k++){
                                     /* 3行3列の変換行列を作用させる */




    if(ipers==1){                                    /* 平行投影 */
      screen[k].h=        ;
      screen[k].v=        ;
    }
    else{                                            /* 中心投影 */
      screen[k].h=         ;
      screen[k].v=         ;
    }
  }
}

void real_gamen(void)                /* スクリーン座標を実画面座標へ変換 */
{
  int   ixmin=0,ixmax=639,iymin=0,iymax=479,id=20;  /* 実画面は(640x480)pixel */
  int   k;
  float hmin,hmax,vmin,vmax,Xscale,Yscale,Fscale;
```

```
  /* ixmin, ixmax, iymin, iymax：実画面座標の最小・最大値 */
  /* id                        ：実画面座標の空白枠の幅 */
  /* hmin, hmax, vmin, vmax    ：スクリーン座標の最小・最大値 */
  /* Xscale, Yscale, Fscale    ：X,Y 方向、及び最終的な変換のスケール値 */
  /* gamen[k].X, gamen[k].Y, k=1,2,...nv ：投影された物体の実画面座標 (X,Y) (nv 個) */

                                    /* スクリーン座標の最大・最小値の検出 */




                                          /* スケールの決定 */
  /* ウィンドウ枠の幅として、x,y ともに id(=20)ずつ(合計 40 ずつ)余裕をとる */


                                          /* 実画面座標値への変換 */


}

void wire_draw(void)                /* ワイヤ・モデルを描く */
{
  int i,apex,ix0,iy0,ix1,iy1;

  G_INIT();                         /* 画面初期化 */

  G_CLS();                          /* 画面消去 */

  for(i=1; i<=ns; i++){
                                    /* 線分の端点座標をセット */
    apex=edge[i][0];
    ix0=gamen[apex].X;
    iy0=gamen[apex].Y;
    apex=edge[i][1];
    ix1=gamen[apex].X;
    iy1=gamen[apex].Y;
                                    /* 線分を描く */
    G_LINE(ix0,iy0,ix1,iy1);
  }
  getch();
}
```