課題（三次元物体の投影図表示）解答例

```
/* 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 */
    for(i=0; i<3; i++){
      cv[i]=0.0;
      for(j=0; j<3; j++){
        cv[i]=cv[i]+mat[i][j]*(xyz[k][j]-eye[j]);
      }
    }
    if(ipers==1){                                   /* 平行投影 */
      screen[k].h=cv[1];              /* h (奥行きを考慮しない)*/
      screen[k].v=cv[2];              /* v (奥行きを考慮しない)*/
    }
    else{                                           /* 中心投影 */
      screen[k].h=-cv[1]/cv[0];       /* h/(-d) (奥行きを考慮する)*/
      screen[k].v=-cv[2]/cv[0];       /* v/(-d) (奥行きを考慮する)*/
    }
  }
}

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 */
  hmin=screen[1].h;
  hmax=screen[1].h;
  vmin=screen[1].v;
```

```
  vmax=screen[1].v;

  for(k=2; k<=nv; k++){
    if( hmax<screen[k].h ) hmax=screen[k].h;
    if( hmin>screen[k].h ) hmin=screen[k].h;
    if( vmax<screen[k].v ) vmax=screen[k].v;
    if( vmin>screen[k].v ) vmin=screen[k].v;
  }
                                        /* 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 */
  Xscale=(ixmax-ixmin-2*id)/(hmax-hmin);      /* Xscale */
  Yscale=(iymax-iymin-2*id)/(vmax-vmin);      /* Yscale */
  Fscale=Xscale;            /* Fscaleの決定：XscaleとYscaleのうち小さい方 */
  if(Xscale>Yscale) Fscale=Yscale;
                                        /* 実画面座標値への変換 */
  for(k=1; k<=nv; k++){
      gamen[k].X= ixmin+id+ (int)( Fscale*(screen[k].h-hmin) );
    gamen[k].Y= iymin+id+ (int)( Fscale*(vmax-screen[k].v) );
  }
}

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();
}
```