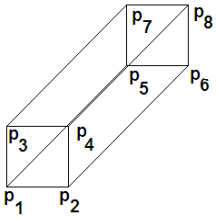
Universidade Federal de São Carlos – UFSCar *campus* Sorocaba

Primeira Prova – Computação Gráfica – 06/maio/2010

Curso de Ciência da Computação

Questão

Considere um sistema SRU (*Sistema de Referência do Universo*), assumindo os limites extremos de x_{min}=-1000*mm*, x_{max}=1000*mm*, y_{min}=-1000*mm*, y_{max}=1000*mm*, z_{min}=-1000*mm* e z_{max}=1000*mm*. O sistema de coordenadas cartesianas está centralizado no SRU, isto é, a origem sendo O=(0,0,0). Assuma um objeto centralizado neste sistema de coordenadas, conforme ilustrado na figura abaixo.



Os pontos que definem este objeto são dados por p1=(30,-10,10), p2=(30,-10,-10), p3=(30,10,10), p4=(30,10,-10), p5=(-30,-10,10), p6=(-30,-10,-10), p7=(-30,10,10) e p8=(-30,10,-10).

Dadas as coordenadas do centro de projeção C=(0,0,C_{PZ}), um ponto do espaço W=(0,0,T_z), que pertença ao plano de projeção π (paralelo ao plano *xy*) e um vetor normal a este plano, **n**=(0,0,1), onde $C_{PZ} \neq 0$ e $T_Z \neq 0$, são dados pelo usuário, construa as seguintes rotinas:

(a) Implemente uma função para rotacionar o objeto em um ângulo de 90º em relação ao eixo *z*. Faça um comentário no código dizendo se o sistema de coordenadas considerado está ou não orientado em relação a regra da mão direita.

(b) Após a rotação de 90º executada pela função anterior, implemente uma função para determinar as novas coordenadas do objeto no plano de projeção π, considerando o centro de projeção C e o ponto W. Faça um comentário no código mencionando como é classificada esta projeção. O protótipo para esta função deve ser *struct ponto3D * Projeta(double, double, struct ponto3D *)*, onde *struct ponto3D { double x, y, z; };*

O primeiro parâmetro formal da função corresponde ao argumento C_{PZ}, o segundo, ao argumento T_z e o terceiro ao ponto que será projetado.

(c) Considere um Retângulo de Visualização (*window*) definido pelos valores (rx_{min},rx_{max}) e (ry_{min},ry_{max}) e que o centro do nosso plano de projeção π, dado por W, coincide com o centro deste retângulo. Os valores dos pontos (rx_{min},rx_{max}) e (ry_{min},ry_{max}) são definidos pelo usuário. Implemente uma função que converta as coordenadas dos pontos projetados, definidas em um SRU, para um SRN (*Sistema de Referência Normalizado*). Posteriormente, utilizando o ambiente *X-Window*, implemente uma função para *visualizar o objeto,* convertendo as coordenadas do SRN para o SRD (*Sistema de Referência do Dispositivo*), isto é, para as coordenadas inteiras da tela.

Observações:

1) Será permitida a consulta aos materiais das aulas, como livros e anotações. Não serão admitidas consultas aos colegas e à internet;
2) Será levado em consideração, na correção do exercício, a estruturação do código, documentação e a capacidade do aluno em interpretar o problema.

Boa prova!

Exemplo de implementação:

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <X11/Xlib.h>

#define PI 3.1415
#define round(x) ((int)((x)+(float)0.5))

/* estruturas de dados */

struct matriz_transformacao {
            float a11, a12, a13;
            float a21, a22, a23;
            float a31, a32, a33;
        };

struct ponto3D {
            float x, y, z;
        };

struct ponto2D {
            int x, y;
        };

/* protótipos das funções */
struct ponto3D * Rotacao(struct matriz_transformacao *, struct ponto3D *);
struct ponto3D * Projeta(float, float, struct ponto3D *);
struct ponto3D * Sru2srn(float, float, float, float, struct ponto3D *);
struct ponto2D * Srn2srd(int, int, struct ponto3D *);
int Desenha_reta(Display *, GC, Window, struct ponto2D, struct ponto2D);
int Visualiza(int, int, struct ponto2D *);

int main(void) {
 int largura, altura, i;
 float Cpz, Tz, rxmin, rxmax, rymin, rymax;
 struct ponto3D * P[9], * PR[9], * PP[9], * PPN[9];
 struct ponto2D * PD[9], pontos[9];
 struct matriz_transformacao * matriz_rotacao;

 for (i=1;i<=8;++i) P[i] = (struct ponto3D *) malloc(sizeof(struct ponto3D));

 Cpz = -50.0;
 Tz = 100.0;

 P[1]->x = 30.0; P[1]->y = -10.0; P[1]->z = 10.0;
 P[2]->x = 30.0; P[2]->y = -10.0; P[2]->z = -10.0;
 P[3]->x = 30.0; P[3]->y = 10.0; P[3]->z = 10.0;
 P[4]->x = 30.0; P[4]->y = 10.0; P[4]->z = -10.0;
 P[5]->x = -30.0; P[5]->y = -10.0; P[5]->z = 10.0;
 P[6]->x = -30.0; P[6]->y = -10.0; P[6]->z = -10.0;
 P[7]->x = -30.0; P[7]->y = 10.0; P[7]->z = 10.0;
 P[8]->x = -30.0; P[8]->y = 10.0; P[8]->z = -10.0;
```

```
 matriz_rotacao = (struct matriz_transformacao *) malloc(sizeof(struct matriz_transformacao));

 (matriz_rotacao->a11) = cos(PI/2.0);
 (matriz_rotacao->a12) = -sin(PI/2.0);
 (matriz_rotacao->a13) = 0.0;
 (matriz_rotacao->a21) = sin(PI/2.0);
 (matriz_rotacao->a22) = cos(PI/2.0);
 (matriz_rotacao->a23) = 0.0;
 (matriz_rotacao->a31) = 0.0;
 (matriz_rotacao->a32) = 0.0;
 (matriz_rotacao->a33) = 1.0;

 for (i=1;i<=8;++i) PR[i] = Rotacao(matriz_rotacao, P[i]);

 for (i=1;i<=8;++i) PP[i] = Projeta(Cpz, Tz, PR[i]);

 /* retângulo de visualização */
 rxmin = -150.0; rxmax = 150.0; rymin = -150.0; rymax = 150.0;

 for (i=1;i<=8;++i) PPN[i] = Sru2srn(rxmin, rxmax, rymin, rymax, PP[i]);

 largura = (int) 2*rxmax;
 altura = (int) 2*rymax;

 for (i=1;i<=8;++i) PD[i] = Srn2srd(largura, altura, PPN[i]);

 for (i=1;i<=8;++i) pontos[i] = *PD[i];

 Visualiza(largura, altura, pontos);

 return 0;
 }

struct ponto3D * Rotacao(struct matriz_transformacao * m, struct ponto3D * p) {
 struct ponto3D * pr;

 pr = (struct ponto3D *) malloc(sizeof(struct ponto3D));

 (pr->x) = (m->a11)*(p->x) + (m->a12)*(p->y) + (m->a13)*(p->z);
 (pr->y) = (m->a21)*(p->x) + (m->a22)*(p->y) + (m->a23)*(p->z);
 (pr->z) = (m->a31)*(p->x) + (m->a32)*(p->y) + (m->a33)*(p->z);

 return pr;

 /* O sistema de coordenadas considerado está orientado em relação a regra da mão direita */
 }

struct ponto3D * Projeta(float cpz, float tz, struct ponto3D * p) {
 float lambda = 0.0;
 struct ponto3D * pp;

 pp = (struct ponto3D *) malloc(sizeof(struct ponto3D));
```

```
 lambda = (tz - (p->z))/((p->z) - cpz);

 pp->x = p->x + lambda*(p->x);
 pp->y = p->y + lambda*(p->y);
 pp->z = p->z + lambda*((p->z) - cpz);

 return pp;

 /* Este algoritmo é classificado como uma Projeção Perspectiva */
 }

struct ponto3D * Sru2srn(float rxmin, float rxmax, float rymin, float rymax, struct ponto3D * P) {
 struct ponto3D * ponto;

 ponto = (struct ponto3D *) malloc(sizeof(struct ponto3D));

 /* transformação de visualização */
 ponto->x = (P->x - rxmin) / (rxmax - rxmin);
 ponto->y = (P->y - rymin) / (rymax - rymin);
 ponto->z = 0.0;

 return ponto;
 }

struct ponto2D * Srn2srd(int largura, int altura, struct ponto3D * P) {
 struct ponto2D * ponto;

 ponto = (struct ponto2D *) malloc(sizeof(struct ponto2D));

 ponto->x = (int) round((P->x)*(largura-1));
 ponto->y = (int) round((P->y)*(altura-1));

 return ponto;
 }

int Desenha_reta(Display * display, GC gc, Window win, struct ponto2D primeiro, struct ponto2D
segundo) {
 float a, b, x, y;
 struct ponto2D aux;

 if (primeiro.x >= segundo.x) {
   aux.x = primeiro.x;
   aux.y = primeiro.y;
   primeiro.x = segundo.x;
   primeiro.y = segundo.y;
   segundo.x = aux.x;
   segundo.y = aux.y;
   }

 if (primeiro.x == segundo.x) {
    x = primeiro.x;
    y = primeiro.y;
    while (y <= segundo.y) {
      XDrawPoint(display, win, gc, (int)round(x), (int)round(y));
```

```
      y++;
      }
    }
  else {
    a = ((float)(segundo.y - primeiro.y))/((float)(segundo.x - primeiro.x));
    b = (primeiro.y) - a*(primeiro.x);
    x = (float) primeiro.x++;
    y = a*x + b;
    while (x <= segundo.x) {
      XDrawPoint(display, win, gc, (int)round(x), (int)round(y));
      y = a*(++x) + b;
      }
    }

 return 0;
 }

int Visualiza(int largura, int altura, struct ponto2D * P) {
 Display * display;
 GC gc;
 Window win, root_window;
 unsigned long valuemask = 0;
 XGCValues values;
 XColor cor;
 int x = 0, y = 0, screennumber, espessura = 4, i;
 unsigned long white_pixel, black_pixel;

 display = XOpenDisplay(NULL);
 screennumber = DefaultScreen(display);
 root_window = RootWindow(display, screennumber);
 black_pixel = BlackPixel(display,screennumber);
 white_pixel = WhitePixel(display, screennumber);
 win =
XCreateSimpleWindow(display,root_window,x,y,largura,altura,espessura,black_pixel,white_pixel);
 XMapWindow(display, win);
 gc = XCreateGC(display, win, valuemask, &values);
 XSync(display, False);
 XSetForeground(display, gc, white_pixel);
 XSetBackground(display, gc, black_pixel);
 XAllocNamedColor(display, XDefaultColormap(display, screennumber), "red", &cor, &cor);
 XSetForeground(display, gc, cor.pixel);
 for (i=1;i<=8;++i) XDrawPoint(display, win, gc, P[i].x, P[i].y);
 Desenha_reta(display, gc, win, P[1], P[2]);
 Desenha_reta(display, gc, win, P[1], P[3]);
 Desenha_reta(display, gc, win, P[2], P[4]);
 Desenha_reta(display, gc, win, P[3], P[4]);
 Desenha_reta(display, gc, win, P[5], P[6]);
 Desenha_reta(display, gc, win, P[6], P[8]);
 Desenha_reta(display, gc, win, P[5], P[7]);
 Desenha_reta(display, gc, win, P[7], P[8]);
 Desenha_reta(display, gc, win, P[3], P[7]);
 Desenha_reta(display, gc, win, P[4], P[8]);
 Desenha_reta(display, gc, win, P[1], P[5]);
 Desenha_reta(display, gc, win, P[2], P[6]);
```

```
XFlush(display);
getchar();
XFreeGC(display,gc);
XCloseDisplay(display);

return 0;
}
```