Relatório sobre o PIPE3D
Pip eline para Visualização Tri-dimensional

Sílvia Delgado Olabarriaga

INESC - Projecto CAD/CAM
Av. Alves Redol, 9 - 2º
1000 - Lisboa

Novembro 1989

1. Objectivo

Esta biblioteca foi desenvolvida a fim de fornecer aos alunos do curso do FSE um conjunto de funções genéricas para visualização de linhas e polígonos tri-dimensionais em uma janela sobre o ecran. Para tanto, foram adaptados procedimentos e equacionamentos matemáticos desenvolvidos para o trabalho descrito em [Olabarriaga 87].

2. Funcionamento geral

A comunicação entre a aplicação e o pipeline de visualização é feita por meio de funções que permitem basicamente as seguintes operações:

a) manipulação dos parâmetros de visualização:
- inicialização
- posição e orientação do observador
- tipo de projecção
- instanciação
- viewport

b) desenho de linhas e polígonos de acordo com os parâmetros anteriormente definidos. Transformações, projecção, clipping e conversão em coordenadas de ecran são responsabilidade do pipeline, isolando a aplicação da implementação de tais operações.

3. Transformações

Os vértices das linhas e polígonos são especificados em coordenadas de objecto, sendo aplicadas as seguintes transformações:

1) Instanciação: multiplica-se as coordenadas por uma matriz que aplica as transformações de escala nos três eixos, rotação sobre os eixos x, y, z e translacção, nesta ordem.

2) Transformação para o sistema de coordenadas do observador, pela multiplicação pela matriz obtida a partir da posição do olho, do alvo e da rotação sobre o eixo z resultante (ver figura 1).

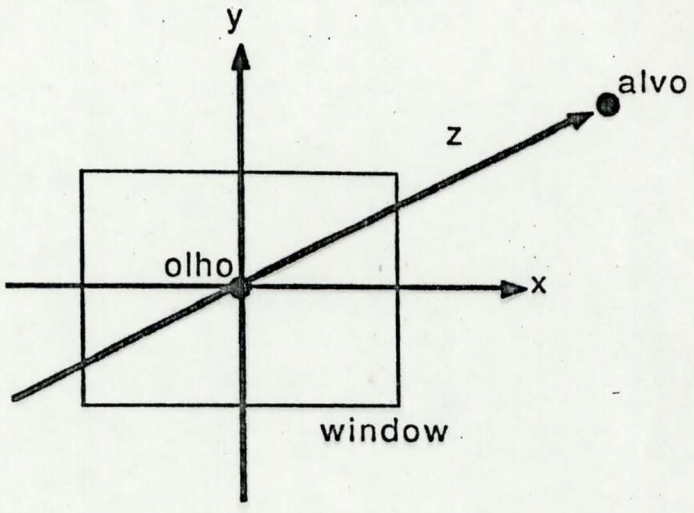

Figura 1 - Sistema de referência da câmera

3) Clipping da linha ou polígono contra o plano da projecção (tri-dimensional). O plano é definido por z=0 (do sistema de referência do observador) e passa pelo ponto definido como "olho". Com esta operação, são eliminados os troços de linhas situados antes do plano da projecção.

4) Projecção no plano, conforme o tipo especificado:

- perspectiva: o centro da projecção é localizado sobre o eixo z negativo, à distância do olho informada como "distância focal" (ver figura 2)
- paralela: a coordenada z é simplesmente descartada

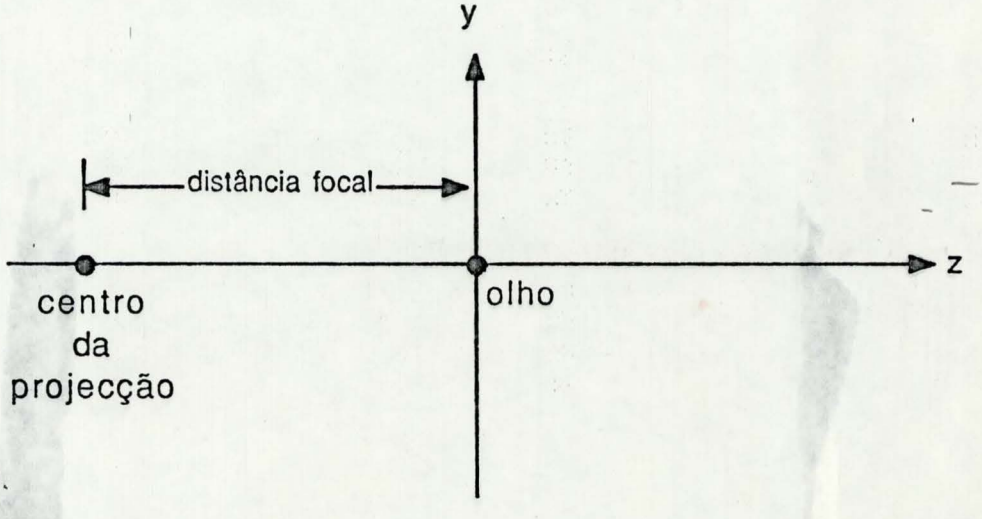

Figura 2 - Posição do centro da projecção

5) Clipping contra a window (bi-dimensional). No caso de projecção perspectiva, os limites da window imitam um slide (0.035 X 0.024).

6) Transformação para coordenadas de ecran.

O pipeline não assume qualquer premissa a respeito dos limites do sistema de coordenadas do mundo, que são especificadas pela aplicação em números reais. Entretanto, os valores assumidos para distância focal e window no caso de projecção perspectiva têm em conta que 1.0 equivale a um metro.

4. Uso em aplicações

Para usar as funções do pipeline, a aplicação deve incluir o ficheiro "pipe3d.h" e ser ligada à biblioteca "libpipe3d.a". No caso de alguns dispositivos (X11, por exemplo), as funções gráficas de saída exigem parâmetros como o identificador da janela, etc. Assim sendo, a aplicação deve incluir também um ficheiro específico para o dispositivo e inicializar o pipeline com uma estrutura contendo os valores necessários.

- iris 4D: não é necessário
- X11: p3d_X11.h

Deve incluir, também, os ficheiros correspondentes ao software gráfico utilizado:

- iris4D: gl.h e device.h
- X11: X11/Xlib.h

Para informações sobre as funções e exemplo de utilização, ver anexos. O primeiro anexo apresenta o header das funções, com a descrição do funcionamento, parâmetros e valores retornados. Os demais anexos contêm um programa de teste do pipeline para iris4d e outro para o X11. Basicamentepermite a visualização de malhas de polígonos do tipo "MESH" [Olabarriaga 89]. Através de interface textual, o utilizador pode especificar os parâmetros de visualização e comandar o desenho da malha.

5. Implementação

As funções do pipeline foram implementadas na linguagem C, independentes do sistema operativo. O programa está organizado nos seguintes ficheiros:

- pipe3d.h: define o tipo de valor retornado pelas funções e a estrutura para armazenamento de coordenadas tri-dimensionais

- p3d_dados.c: define as variáveis de estado do pipeline:
  - factor de escala, ângulo de rotação e translação.
  - posição e orientação do observador, definidas por dois pontos (olho e alvo) e um ângulo (rotação em z).
  - tipo de projecção e distância focal.
  - limites da window e viewport
  - variáveis internas com valores pré-calculados para optmização (factores de escala, matrizes).

• p3d_dados.h: declara as variáveis de estado compartilhadas por todos os módulos. Deve ser incluido em todos os módulos do pipeline 3D, excepto em p3d_dados.c.

• p3d_defaults.h: declara constantes usadas para inicialização dos parâmetros de visualização (função P3D_init)

• p3d_princ.c: contém as funções que podem ser chamadas pela aplicação.

• p3d_vis.c: contém as rotinas que desenham as linhas e polígonos, efectuando as transformações para visualização segundo os parâmetros contidos nas variáveis de estado (ver item 4 - Transformações). Possui rotinas de clipping diferentes para linhas e polígonos. As funções de desenho utilizam interface genérica para saída no dispositivo gráfico (ver "gráficos.c")

• p3d_vis.h: contém definições de constantes, macros e funções usadas no módulo p3d_vis.c

• p3d_util.c: contém funções matemáticas de uso geral.

• p3d_util.h: declara as funções do módulo p3d_util.c.

• p3d_calc.c: contém funções para cálculo de matrizes de transformação e factores de escala a partir dos parâmetros de visualização.

• gráficos.c: módulo que implementa as funções de acesso ao dispositivo gráfico:

- GR_init: inicializa o driver internamente (guarda identificadores da janela, contexto gráfico, etc, quando for este o caso)
- GR_color: programa a cor corrente
- GR_move: desloca a posição de desenho para um ponto do ecran
- GR_line: desenha uma linha do ponto anterior ao ponto indicado.

Há um ficheiro para cada dispositivo, p3d_iris4d.c e p3d_X11.c, além de um ficheiro especial para passagem de parâmetros referentes ao contextoi gráfico em que a aplicação está a correr.

6. Referências Bibliográficas

[Olabarriaga 87] Sílvia Olabarriaga. "SPA - Um sistema computacional para produção de animação". Porto Alegre, CPGCC-UFRGS, Brasil, 1987

[Olabarriaga 89] Sílvia Olabarriaga. "Proposta para descrição de malhas de polígonos (MESH)". Lisboa, INESC, 1989.

ANEXO 1

Descrição das Funções do Pipeline 3D

```
*m-------------------------------------------------------------------------+
   Contactar : Silvia Delgado Olabarriaga (Ext.220)                        |
                                                                           |
   Funcoes :                                                               |
                P3D_init                                                   |
                P3D_set_transf                                             |
                P3D_set_camera                                             |
                P3D_set_ortho                                              |
                P3D_set_viewport                                           |
                P3D_draw_line                                              |
                P3D_draw_face                                              |
|                                                                          |
|  Copyright 1989, INESC. All rights reserved.                             |
+-------------------------------------------------------------------------*/

/*-----------------------------------------------------+
| Tipo para armazenamento de um ponto em tres dimensoes |
| Deve ser usado para passagem de parametros para as    |
| funcoes do pipeline. (definida em pipe3d.h            |
+-----------------------------------------------------*/

typedef struct  {
                float   x,
                        y,
                        z;
                } P3D_XYZ;


/*f-------------------------------------------------------------------------+
|   P3D_init                                                               |
+-------------------------------------------------------------------------+
|   Accao : inicializa variaveis internas do pipeline com valores default e |
|       calcula matrizes e escalas.                                        |
|       Transformacoes: escala = (1,1,1), rotacao = (0,0,0), transl.= (0,0,0) |
|       Observador:     olho = (0,0,20), alvo = (0,0,0), angulo = 0        |
|       Projeccao:      perspectiva, com foco = 0.05                       |
|       Window:         (-0.012, -0.017) a (0.012 a 0.017)                 |
|       Viewport:       (0,0) a (1,1)                                      |
|       Chama a rotina de inicializacao de variaveis dependentes do dispositivo |
|   Argumentos:                                                            |
|               param_disp = ponteiro p/ estrutura com parametros especificos |
|                            para acesso ao dispositivo grafico.           |
|                            O formato da estrutura depende do dispositivo. |
+-------------------------------------------------------------------------*/

void P3D_init ( param_disp )
P3D_X11 * param_disp;
    {
    }

/*-----------------------------------------------+
| Estrutura para representacao do ambiente no X11 |
| (--> definida em p3d_X11.h)                     |
+-----------------------------------------------*/

typedef struct  {
                Display *display;
                Window  window;
                GC      gc;
                } P3D_X11;
```

```
/*f-------------------------------------------------------------------------+
|   P3D_set_transf                                                          |
+-------------------------------------------------------------------------+
|   Accao : Inicializa variaveis internas com os parametros para instancia- |
|       mento (escala, rotacao e translacao) e calcula matriz de transformacao
|       Ordem das transformacoes: escala, rotacao e translacao.             |
|   Argumentos :                                                            |
|               scale  = factor de escala (em tres eixos)                   |
|               rotate = angulo de rotacao (em tres eixos)                  |
|               translate = translaccao (em tres eixos)                     |
+-------------------------------------------------------------------------*/

void P3D_set_transf ( scale, rotate, translate )
P3D_XYZ scale;
P3D_XYZ rotate;
P3D_XYZ translate;
    {
    }

/*f-------------------------------------------------------------------------+
|   P3D_set_camera                                                          |
+-------------------------------------------------------------------------+
|   Accao : Posiciona o observador (olho, alvo e angulo) e inicializa variaveis
|       internas para projeccao perspectiva (foco).                         |
|       O tamanho da window e' pre-definido para simular um visor (4X3).    |
|       O plano da projeccao esta' em z=0 e a direccao da projeccao coincide
|       com o eixo z.                                                       |
|   Argumentos :                                                            |
|               eye = posicao do observador                                 |
|               target  = alvo da camera                                    |
|               tilt    = rotacao sobre o eixo z da camera                  |
|               focal   = distancia focal (lente)                           |
+-------------------------------------------------------------------------*/

void P3D_set_camera ( eye, target, tilt, focal )
P3D_XYZ eye;
P3D_XYZ target;
float tilt;
float focal;
    {
    }

/*f-------------------------------------------------------------------------+
|   P3D_set_ortho                                                           |
+-------------------------------------------------------------------------+
|   Accao : Posiciona o observador (olho, alvo e angulo) e inicializa variaveis
|       para projeccao paralela. O tamanho da window e' determinado pelo pro-
|       gramador, sendo esta a unica forma de controle do tamanho da imagem.
|       O plano da projeccao esta' em z=0 e a direccao da projeccao coincide
|       com o eixo z.                                                       |
|       OBS: a relacao de aspecto em relacao 'as dimensoes da viewport deve
|       ser controlada para evitar distorcoes da imagem                     |
|   Argumentos :                                                            |
|               eye = posicao do observador                                 |
|               target  = alvo da camera                                    |
|               tilt    = rotacao sobre o eixo z da camera                  |
|               low_x, low_y = canto inferior esquerdo da window            |
|               up_x, up_y = canto superior direito da window               |
+-------------------------------------------------------------------------*/

void P3D_set_ortho ( eye, target, tilt, low_x, low_y, up_x, up_y )
P3D_XYZ eye;
P3D_XYZ target;
float tilt;
float low_x, low_y;
```

```
loat up_x, up_y;
    {
    }

/*f-------------------------------------------------------------------------+
|   P3D_set_viewport                                                        |
+-------------------------------------------------------------------------+
|   Accao : Modifica limites para desenho no ecran. Armazena os novos valores |
|       nas variaveis internas (em coordenadas de ecran) e recalcula factores |
|       de escala                                                           |
|   Argumentos :                                                            |
|               x_low, y_low    = canto inferior esquerdo                   |
|               x_up, y_up      = canto superior direito                    |
+-------------------------------------------------------------------------*/

void P3D_set_viewport ( x_low, y_low, x_up, y_up )
int x_low, y_low;
int x_up, y_up;
    {
    }

/*f-------------------------------------------------------------------------+
|   P3D_draw_line                                                           |
+-------------------------------------------------------------------------+
|   Accao : Desenha uma linha entre dois pontos do espaco, considerando a   |
|       posicao e orientacao corrente do observador, tipo de projeccao, window |
|       e viewport.                                                         |
|       Transformacoes:                                                     |
|               1- escala, rotacao e translacao                             |
|               2- para sistema de referencia do observador                 |
|               3- recorte 3D de linha contra o plano da projeccao.         |
|               4- projeccao                                                |
|               5- recorte 2D de linha contra window                        |
|               6- para coordenadas de ecran.                               |
|   Argumentos :                                                            |
|               color   = cor da linha (indice da palete)                   |
|               p1      = ponteiro p/ ponto inicial da linha                |
|               p2      = ponteiro p/ ponto final da linha                  |
+-------------------------------------------------------------------------*/

void P3D_draw_line ( color, p1, p2 )
int color;
P3D_XYZ * p1;
P3D_XYZ * p2;
    {
    }

/*f-------------------------------------------------------------------------+
|   P3D_draw_face                                                           |
+-------------------------------------------------------------------------+
|   Accao : Desenha o contorno de uma faceta determinada por uma lista de   |
|       pontos do espaco. Considera a posicao e orientacao corrente do      |
|       observador, tipo de projeccao, window e viewport.                   |
|       Transformacoes:                                                     |
|               1- escala, rotacao e translacao                             |
|               2- para sistema de referencia do observador                 |
|               3- recorte 3D de poligono contra o plano da projeccao.      |
|               4- projeccao                                                |
|               5- recorte 2D de poligono contra window                     |
|               6- para coordenadas de ecran.                               |
|   Argumentos :                                                            |
|               color   = cor da linha (indice da palete)                   |
|               n       = numero de vertices da faceta                      |
|               vertex  = ponteiro para a lista de vertices                 |
+-------------------------------------------------------------------------*/
```

```
void P3D_draw_face ( color, n, vertex )
int color;
int n;
P3D_XYZ vertex[];
    {
    }
```

ANEXO 2

Exemplo de utilização em ambiente de X11

(testado em Sun e Aviion)

```
/*m-------------------------------------------------------------------------+
|   Modulo: exemplo.c                                                      |
+-------------------------------------------------------------------------+
|   Descricao : Programa para carregas malhas de poligonos no formato MESH e |
|       visualizar com a pipeline 3D.                                      |
|                                                                          |
|   Contactar : Silvia Delgado Olabarriaga                                 |
|                                                                          |
|   Historia :                                                             |
|       Data      Autor     Comentarios                                    |
|     14-Out-89   silvia    Versao Inicial                                 |
|     16-Out-89   silvia    com projeccao paralela e instanciamento        |
|     17-Out-89   silvia    para X11                                       |
|                                                                          |
|   Copyright 1989, INESC. All rights reserved.                            |
+-------------------------------------------------------------------------*/

#include <math.h>
#include <ctype.h>
#include <X11/Xlib.h>
#include <pipe3d.h>
#include <p3d_X11.h>
#include <mesh.h>

#define PERSPECTIVA 1
#define ORTOGONAL   2

/*-----------------------------------+
| Variaveis globais ao programa todo |
+-----------------------------------*/
static M_Mesh   m;

static P3D_XYZ  pos,
                alvo;

static struct {
        float xmin, ymin, zmin;
        float xmax, ymax, zmax;
       } e;

static P3D_X11 pdisp;


/*f-------------------------------------------------------------------------+
|   main                                                                   |
+-------------------------------------------------------------------------+
|   Accao : Fica em loop, a ler o nome do ficheiro e visualiza-lo pela camera. |
|       Para quando ler um nome "fim".                                     |
+-------------------------------------------------------------------------*/

main ()
    {
    inic_graficos();
    while ( carrega() )
        {
        adapta();
        visualiza();
        }
    }

/*f-------------------------------------------------------------------------+
|   inic_graficos                                                          |
+-------------------------------------------------------------------------+
|   Accao : Obtem limites da janela, programa pipeline da iris para 2D e   |
|       inicializa pipeline 3D.
```

```
+-----------------------------------------------------------------------------*/

static inic_graficos ( )
    {
    Window parent;
    int screen;
    int win_x, win_y;
    int width, height;
    XGCValues values;

    pdisp.display = XOpenDisplay(NULL);
    if (pdisp.display == NULL) {
        printf("fatal error: can't open display\n");
        exit(1);
    }
    screen = DefaultScreen(pdisp.display);
    parent = RootWindow(pdisp.display, screen);
    XMapWindow (pdisp.display, parent);
    win_x = win_y = 0;
    width = DisplayWidth(pdisp.display, screen);
    height = DisplayHeight(pdisp.display, screen);
    pdisp.window = XCreateWindow(
        pdisp.display, parent, win_x, win_y, width, height, 0, 0,
        CopyFromParent, CopyFromParent, 0, 0);
    values.background = WhitePixel(pdisp.display, screen);
    values.foreground = BlackPixel(pdisp.display, screen);
    pdisp.gc = XCreateGC (pdisp.display, pdisp.window,
                          GCForeground | GCBackground, &values);
    XSetWindowBackground (pdisp.display, pdisp.window,
                          BlackPixel(pdisp.display, screen));
    XSetForeground (pdisp.display, pdisp.gc, WhitePixel(pdisp.display, screen));
    XMapWindow (pdisp.display, pdisp.window);
    XSynchronize(pdisp.display, 1);
    XClearWindow( pdisp.display, pdisp.window);

    P3D_init(&pdisp);
    P3D_set_viewport( (int) win_x, (int) height-1, (int) width-1, (int) win_y );
    }

/*f-----------------------------------------------------------------------------+
|   carrega                                                                    |
+-----------------------------------------------------------------------------+
|   Accao : Pergunta nome do ficheiro e carrega a malha de poligonos. Fica
|       em loop ate' conseguir carregar uma malha corretamente ou ler um nome
|       nulo.
+-----------------------------------------------------------------------------*/

static int carrega ()
    {
    char nome[20];
    M_Erro r;

    if ( m != NULL )
        M_free_mesh(m);
    m = M_make_mesh();
    if ( m == NULL )
        {
        printf("erro criacao da malha\n");
        exit();
        }
    while (1)
        {
        printf("\nEntre nome do ficheiro: (fim) ");
        scanf("%20s", nome);
        if ( strcmp(nome, "fim") == 0 )
            return(0);
```

```
        if ( (r = M_load_mesh(nome, m)) == M_OK )
            return(1);
        else
            printf("erro de carga (%d)\n", r);
        }
    }

/*f-------------------------------------------------------------------------+
|   adapta                                                                  |
+--------------------------------------------------------------------------+
|   Accao : calcula raio da esfera que contem o objeto, a fim de permitir que |
|       a posicao inicial da camera  seja calculada corretamente.           |
|       Percorre todos os vertices para determinar os extremos do paralelepi- |
|       pedo que envolve o objeto. Depois calcula a diagonal deste e determina |
|       que este e' o diametro da esfera que contem o objeto.               |
+--------------------------------------------------------------------------*/

static adapta ()
    {
    M_Vertex v;
    void verif_env();
    float diam;

    v = M_get_vertex(m, 1);                    /* min e max = prim. vertice */
    e.xmin = e.xmax = v->x;
    e.ymin = e.ymax = v->y;
    e.zmin = e.zmax = v->z;
    M_for_all_vertex(m, NULL, verif_env);                  /* acha min, max */
    e.xmax -= e.xmin;                                     /* calcula tamanho */
    e.ymax -= e.ymin;
    e.zmax -= e.zmin;
                                                          /* calcula diagonal */
    diam = sqrt(e.xmax*e.xmax + e.ymax*e.ymax + e.zmax*e.zmax);
    alvo.x = pos.x = e.xmin + e.xmax / 2.0;               /* centro do objeto */
    alvo.y = pos.y = e.ymin + e.ymax / 2.0;
    alvo.z = e.zmin + e.zmax / 2.0;
    pos.z = alvo.z + 3 * diam;
    }

/*f-------------------------------------------------------------------------+
|   verif_env                                                               |
+--------------------------------------------------------------------------+
|   Accao : Verifica se um vertice esta' fora da envoltoria recebida como pa- |
|       rametro. Se estiver, atualiza envoltoria para conter o vertice      |
|   Argumentos :                                                            |
|               e = descritor da envoltoria (minimo e maximo)               |
|               v = ponteiro para vertice                                   |
+--------------------------------------------------------------------------*/
static void verif_env ( lixo, v )
NODE *lixo;
M_Vertex v;
    {
    if ( e.xmin > v->x )                   /* atualiza xmin */
        e.xmin = v->x;
    else if (e.xmax < v->x )               /* atualiza xmax */
            e.xmax = v->x;
    if ( e.ymin > v->y )                   /* atualiza ymin */
        e.ymin = v->y;
    else if (e.ymax < v->y )               /* atualiza ymax */
            e.ymax = v->y;
    if ( e.zmin > v->z )                   /* atualiza zmin */
        e.zmin = v->z;
    else if (e.zmax < v->z )               /* atualiza zmax */
            e.zmax = v->z;
    }
```

```
*f-------------------------------------------------------------------------+
   visualiza                                                               |
---------------------------------------------------------------------------+
   Accao : Le parametros da camera e desenha a malha de poligonos          |
---------------------------------------------------------------------------*/

tatic visualiza ()
   {
   float   tilt;
   float   focal;
   float   low_x, low_y, up_x, up_y;
   char    resp[10];
   int     tipo;
   P3D_XYZ trans, rot, escala;

   tilt = 0.0;
   low_x = 0.0; low_y = 0.0;
   up_x = 0.0; up_y = 0.0;
   tipo = PERSPECTIVA;
   focal = 0.05;
   escala.x = 1.0; escala.y = 1.0; escala.z = 1.0;
   rot.x = 0.0; rot.y = 0.0; rot.z = 0.0;
   trans.x = 0.0; trans.y = 0.0; trans.z = 0.0;

   while (1)
       {
       printf("\n(E)ye (T)arget ti(L)t (F)ocal (W)indow");
       printf("\ne(S)cala (R)otacao tra(N)slacao");
       printf("\n(D)esenhar (A)pagar (O)utro ficheiro\n>");
       scanf("%10s", resp);
       switch ( toupper(resp[0]) )
           {
           case 'E' :  printf("\nEntre posicao da camera: ");
                       printf("(%g, %g, %g)  ", pos.x, pos.y, pos.z);
                       scanf("%f %f %f", &pos.x, &pos.y, &pos.z);
                       break;
           case 'T' :  printf("\nEntre alvo da camera: ");
                       printf("(%g, %g, %g)  ", alvo.x, alvo.y, alvo.z);
                       scanf("%f %f %f", &alvo.x, &alvo.y, &alvo.z);
                       break;
           case 'L' :  printf("\nEntre angulo: ");
                       printf("(%g)  ", tilt);
                       scanf("%f", &tilt);
                       break;
           case 'F' :  printf("\nEntre distancia focal: ");
                       printf("(%g)  ", focal);
                       scanf("%f", &focal);
                       tipo = PERSPECTIVA;
                       break;
           case 'W' :  printf("\nEntre limites da window: ");
                       printf("(%g, %g, %g, %g)  ", low_x, low_y,
                                                    up_x, up_y);
                       scanf("%f %f %f %f", &low_x, &low_y, &up_x, &up_y);
                       tipo = ORTOGONAL;
                       break;
           case 'S' :  printf("\nEntre factor de escala: ");
                       printf("(%g, %g, %g)  ", escala.x, escala.y, escala.z);
                       scanf("%f %f %f", &escala.x, &escala.y, &escala.z);
                       break;
           case 'R' :  printf("\nEntre angulo de rotacao: ");
                       printf("(%g, %g, %g)  ", rot.x, rot.y, rot.z);
                       scanf("%f %f %f", &rot.x, &rot.y, &rot.z);
                       break;
           case 'N' :  printf("\nEntre nova posicao: ");
                       printf("(%g, %g, %g)  ", trans.x, trans.y, trans.z);
                       scanf("%f %f %f", &trans.x, &trans.y, &trans.z);
```

```
                    break;
            case 'D' :  P3D_set_transf( escala, rot, trans );
                        if ( tipo == PERSPECTIVA )
                            P3D_set_camera( pos, alvo, tilt, focal );
                        else
                            P3D_set_ortho(pos,alvo,tilt,low_x,low_y,up_x,up_y );
                        wire_mesh();
                        break;
            case 'A' :  XClearWindow( pdisp.display, pdisp.window);
                        break;
            case 'O' :  return;
                        }
        }
    }

/*f-------------------------------------------------------------------------+
|   wire_mesh                                                               |
+--------------------------------------------------------------------------+
|   Accao : Desenha as arestas de uma malha de poligonos.                   |
+-------------------------------------------------------------------------*/

static wire_mesh ( )
  {
  int wire_face();

  M_for_all_face(m, m, wire_face);
  }

/*f-------------------------------------------------------------------------+
|   wire_face                                                               |
+--------------------------------------------------------------------------+
|   Accao : Desenha as arestas de uma face da malha de poligonos            |
+-------------------------------------------------------------------------*/

static wire_face ( m, f )
M_Mesh m;
M_Face f;
  {
  int i;
  P3D_XYZ  *buffer;
  register M_Vertex v;

  buffer = (P3D_XYZ *) malloc( f->nedges * sizeof(P3D_XYZ) );
  for ( i = 0; i < f->nedges; i++ )
    {
    v = M_get_vertex(m, f->edges[i]);
    buffer[i].x = v->x;
    buffer[i].y = v->y;
    buffer[i].z = v->z;
    }
  P3D_draw_face( 1, f->nedges, buffer );
  free(buffer);
  }
```

ANEXO 3

Exemplo de utilização
em ambiente de iris 4D

(testado em Cyber 910)

```
/*m-------------------------------------------------------------------------+
|   Modulo: ex2-iris4d.c                                                    |
+-------------------------------------------------------------------------+
|   Descricao : Programa para carregas malhas de poligonos no formato MESH e |
|       visualizar com a pipeline 3D.                                       |
|                                                                           |
|   Contactar : Silvia Delgado Olabarriaga                                  |
|                                                                           |
|   Historia :                                                              |
|       Data        Autor       Comentarios                                 |
|     14-Out-89     silvia      Versao Inicial                              |
|     16-Out-89     silvia      com projeccao paralela e instanciamento     |
|                                                                           |
|   Copyright 1989, INESC. All rights reserved.                             |
+-------------------------------------------------------------------------*/

#include <mesh.h>
#include <gl.h>
#include <device.h>
#include <math.h>
#include <pipe3d.h>

#define PERSPECTIVA 1
#define ORTOGONAL   2


static M_Mesh   m;

static P3D_XYZ  pos,
                alvo;

struct {
        Coord xmin, ymin, zmin;
        Coord xmax, ymax, zmax;
       } e;


/*f-------------------------------------------------------------------------+
|   main                                                                    |
+-------------------------------------------------------------------------+
|   Accao : Fica em loop, a ler o nome do ficheiro e visualiza-lo pela camera. |
|       Para quando ler um nome "fim".                                      |
+-------------------------------------------------------------------------*/

main ()
    {
    inic_graficos();
    while ( carrega() )
        {
        adapta();
        visualiza();
        }
    }

/*f-------------------------------------------------------------------------+
|   inic_graficos                                                           |
+-------------------------------------------------------------------------+
|   Accao : Obtem limites da janela, programa pipeline da iris para 2D e    |
|       inicializa pipeline 3D.                                             |
+-------------------------------------------------------------------------*/

static inic_graficos ( )
    {
    long window;
    long xleft, xright,
```

```
        ybottom, ytop;
    P3D_XYZ pos, alvo;

/*-----------------------------------------------+
| Abre a janela e inicializa pipeline da iris |
+-----------------------------------------------*/
    foreground();
    keepaspect(4,3);
    window = winopen("");
    getorigin( &xleft, &ybottom );
    getsize( &xright, &ytop );
    xright += xleft;
    ytop += ybottom;
    ortho2( xleft, xright, ybottom, ytop );
    reshapeviewport();
    color(BLACK);
    clear();

    P3D_init();
    P3D_set_viewport( (int) xleft, (int) ybottom, (int) xright, (int) ytop );
    }

/*f-------------------------------------------------------------------------+
|   carrega                                                                 |
+-------------------------------------------------------------------------+
|   Accao : Pergunta nome do ficheiro e carrega a malha de poligonos. Fica  |
|       em loop ate' conseguir carregar uma malha corretamente ou ler um nome|
|       nulo.                                                               |
+-------------------------------------------------------------------------*/

static int carrega ()
    {
    char nome[20];
    M_Erro r;

    if ( m != NULL )
        M_free_mesh(m);
    m = M_make_mesh();
    if ( m == NULL )
        {
        printf("erro criacao da malha\n");
        exit();
        }
    while (1)
        {
        printf("\nEntre nome do ficheiro: (fim) ");
        scanf("%20s", nome);
        if ( strcmp(nome, "fim") == 0 )
            return(0);
        if ( (r = M_load_mesh(nome, m)) == M_OK )
            return(1);
        else
            printf("erro de carga (%d)\n", r);
        }
    }

/*f-------------------------------------------------------------------------+
|   adapta                                                                  |
+-------------------------------------------------------------------------+
|   Accao : calcula raio da esfera que contem o objeto, a fim de permitir que|
|       a posicao inicial da camera  seja calculada corretamente.           |
|       Percorre todos os vertices para determinar os extremos do paralelepi-|
|       pedo que envolve o objeto. Depois calcula a diagonal deste e determina|
|       que este e' o diametro da esfera que contem o objeto.               |
+-------------------------------------------------------------------------*/
```

```
static adapta ()
    {
    M_Vertex v;
    void verif_env();
    float diam;

    v = M_get_vertex(m, 1);                          /* min e max = prim. vertice */
    e.xmin = e.xmax = v->x;
    e.ymin = e.ymax = v->y;
    e.zmin = e.zmax = v->z;
    M_for_all_vertex(m, NULL, verif_env);                  /* acha min, max */
    e.xmax -= e.xmin;                                     /* calcula tamanho */
    e.ymax -= e.ymin;
    e.zmax -= e.zmin;
                                                          /* calcula diagonal */
    diam = sqrt(e.xmax*e.xmax + e.ymax*e.ymax + e.zmax*e.zmax);
    alvo.x = pos.x = e.xmin + e.xmax / 2.0;               /* centro do objeto */
    alvo.y = pos.y = e.ymin + e.ymax / 2.0;
    alvo.z = e.zmin + e.zmax / 2.0;
    pos.z = alvo.z + 3 * diam;
    }

/*f-------------------------------------------------------------------------+
|   verif_env                                                               |
+--------------------------------------------------------------------------+
|   Accao : Verifica se um vertice esta' fora da envoltoria recebida como pa-|
|       rametro. Se estiver, atualiza envoltoria para conter o vertice      |
|   Argumentos :                                                            |
|               e = descritor da envoltoria (minimo e maximo)               |
|               v = ponteiro para vertice                                   |
+--------------------------------------------------------------------------*/
static void verif_env ( lixo, v )
NODE *lixo;
M_Vertex v;
    {
    if ( e.xmin > v->x )                    /* atualiza xmin */
        e.xmin = v->x;
    else if (e.xmax < v->x )                /* atualiza xmax */
            e.xmax = v->x;
    if ( e.ymin > v->y )                    /* atualiza ymin */
        e.ymin = v->y;
    else if (e.ymax < v->y )                /* atualiza ymax */
            e.ymax = v->y;
    if ( e.zmin > v->z )                    /* atualiza zmin */
        e.zmin = v->z;
    else if (e.zmax < v->z )                /* atualiza zmax */
            e.zmax = v->z;
    }

/*f-------------------------------------------------------------------------+
|   visualiza                                                               |
+--------------------------------------------------------------------------+
|   Accao : Le parametros da camera e desenha a malha de poligonos          |
+--------------------------------------------------------------------------*/

static visualiza ()
    {
    float   tilt;
    float   focal;
    float   low_x, low_y, up_x, up_y;
    char    resp[10];
    int     tipo;
    P3D_XYZ trans, rot, escala;

    tilt = 0.0;
    low_x = 0.0; low_y = 0.0;
```

```
up_x = 0.0; up_y = 0.0;
tipo = PERSPECTIVA;
focal = 0.05;
escala.x = 1.0; escala.y = 1.0; escala.z = 1.0;
rot.x = 0.0; rot.y = 0.0; rot.z = 0.0;
trans.x = 0.0; trans.y = 0.0; trans.z = 0.0;

while (1)
    {
    printf("\n(E)ye (T)arget ti(L)t (F)ocal (W)indow");
    printf("\ne(S)cala (R)otacao tra(N)slacao");
    printf("\n(D)esenhar (A)pagar (O)utro ficheiro\n>");
    scanf("%10s", resp);
    switch ( toupper(resp[0]) )
        {
        case 'E' :  printf("\nEntre posicao da camera: ");
                    printf("(%g, %g, %g)  ", pos.x, pos.y, pos.z);
                    scanf("%f %f %f", &pos.x, &pos.y, &pos.z);
                    break;
        case 'T' :  printf("\nEntre alvo da camera: ");
                    printf("(%g, %g, %g)  ", alvo.x, alvo.y, alvo.z);
                    scanf("%f %f %f", &alvo.x, &alvo.y, &alvo.z);
                    break;
        case 'L' :  printf("\nEntre angulo: ");
                    printf("(%g)  ", tilt);
                    scanf("%f", &tilt);
                    break;
        case 'F' :  printf("\nEntre distancia focal: ");
                    printf("(%g)  ", focal);
                    scanf("%f", &focal);
                    tipo = PERSPECTIVA;
                    break;
        case 'W' :  printf("\nEntre limites da window: ");
                    printf("(%g, %g, %g, %g)  ", low_x, low_y,
                                                 up_x, up_y);
                    scanf("%f %f %f %f", &low_x, &low_y, &up_x, &up_y);
                    tipo = ORTOGONAL;
                    break;
        case 'S' :  printf("\nEntre factor de escala: ");
                    printf("(%g, %g, %g)  ", escala.x, escala.y, escala.z);
                    scanf("%f %f %f", &escala.x, &escala.y, &escala.z);
                    break;
        case 'R' :  printf("\nEntre angulo de rotacao: ");
                    printf("(%g, %g, %g)  ", rot.x, rot.y, rot.z);
                    scanf("%f %f %f", &rot.x, &rot.y, &rot.z);
                    break;
        case 'N' :  printf("\nEntre nova posicao: ");
                    printf("(%g, %g, %g)  ", trans.x, trans.y, trans.z);
                    scanf("%f %f %f", &trans.x, &trans.y, &trans.z);
                    break;
        case 'D' :  P3D_set_transf( escala, rot, trans );
                    if ( tipo == PERSPECTIVA )
                        P3D_set_camera( pos, alvo, tilt, focal );
                    else
                        P3D_set_ortho(pos,alvo,tilt,low_x,low_y,up_x,up_y );
                    wire_mesh();
                    break;
        case 'A' :  color(BLACK);
                    clear();
                    break;
        case 'O' :  return;
        }
    }
}

/*f----------------------------------------------------------------------+
```

```
|   wire_mesh                                                                  |
+------------------------------------------------------------------------------+
|   Accao : Desenha as arestas de uma malha de poligonos.                      |
+------------------------------------------------------------------------------*/

static wire_mesh ( )
  {
  int wire_face();

  M_for_all_face(m, m, wire_face);
  }

/*f-------------------------------------------------------------------------+
|   wire_face                                                               |
+---------------------------------------------------------------------------+
|   Accao : Desenha as arestas de uma face da malha de poligonos            |
+---------------------------------------------------------------------------*/

static wire_face ( m, f )
M_Mesh m;
M_Face f;
  {
  int i;
  P3D_XYZ  *buffer;
  register M_Vertex v;

  buffer = (P3D_XYZ *) malloc( f->nedges * sizeof(P3D_XYZ) );
  for ( i = 0; i < f->nedges; i++ )
    {
    v = M_get_vertex(m, f->edges[i]);
    buffer[i].x = v->x;
    buffer[i].y = v->y;
    buffer[i].z = v->z;
    }
  P3D_draw_face( WHITE, f->nedges, buffer );
  free(buffer);
  }
```