UGF - Universidade Gama Filho
Campus Piedade – Turma 305
INF438 – Computação Gráfica

Caminhão Guincho e Carro de Corrida em OpenGL

Prof.: Cláudio Márcio
Ruan Pedro Dias Seraphim – Mat. 20101075779
Sérgio da Silva Pereira – Mat. 20101609418

Rio de Janeiro – Junho de 2013

SUMÁRIO

Apresentação

Este trabalho acadêmico exibe os códigos fontes dos programas compilados em C++ na IDE Dev C++ juntamente com as bibliotecas de OpenGL para o desenvolvimento dos projetos Caminhão Guincho e do Carro de Corrida em atendimento a disciplina de Computação Gráfica do curso de Ciências da Computação da Universidade Gama Filho sob orientação do professor Cláudio Márcio, sendo utilizado o AutoCAD como ferramenta auxilia elaboração dos desenhos.

1 – Caminhão Guincho

1.1 – Desenho

Desenho 1 – Caminhão Guincho

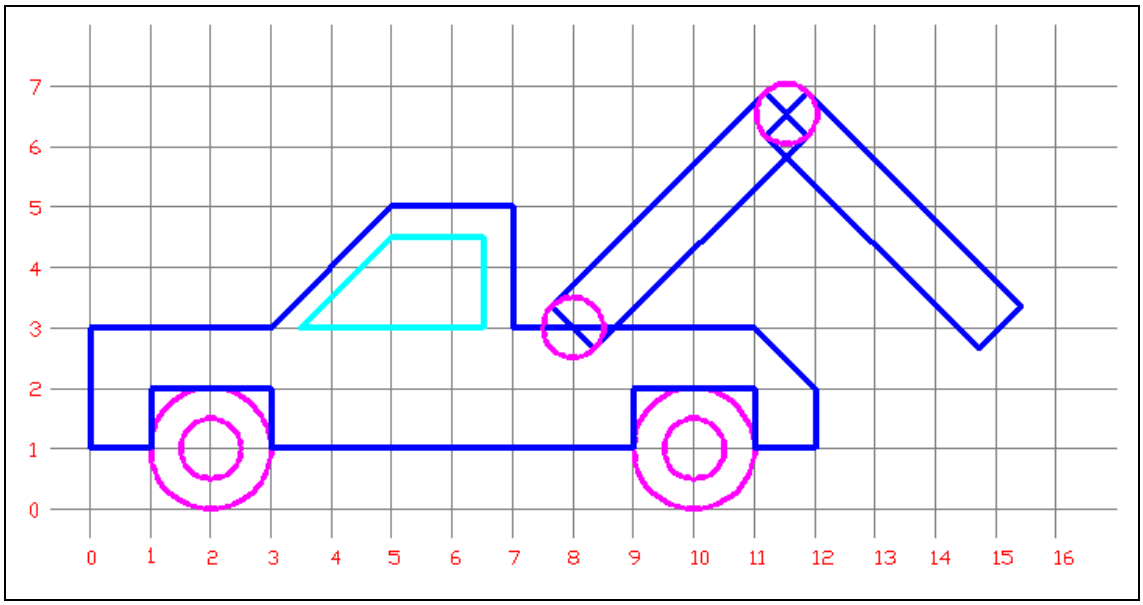
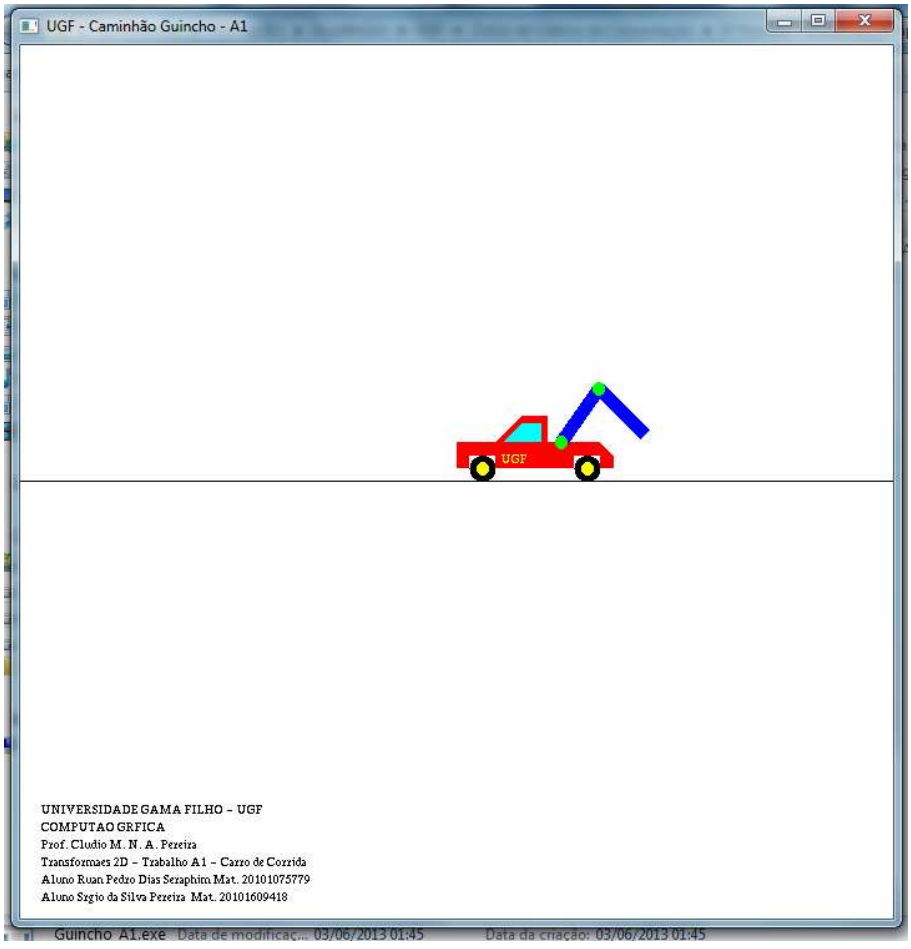


Figura 1 – Imagem do programa Caminhão Guincho



1.2 – Código fonte

```
//--------------------------------------------------------------------------
//
// UNIVERSIDADE GAMA FILHO -UGF
//
// COMPUTAÇÃO GRÁFICA
//
// Prof. Cláudio M. N. A. Pereira
//
// Transformações 2D - Trabalho A1 - Caminhão Guincho
//
// Aluno Ruan Pedro Dias Seraphim Mat. 20101075779
//
// Aluno Sérgio da Silva Pereira  Mat. 20101609418
//
//--------------------------------------------------------------------------

#include <gl/gl.h>
#include <gl/glut.h>
#include <stdlib.h>
#include <math.h>

//#include <iostream.h>

float Tx=0, Ty=0, J1=-35, J2=-100, S=3;

char texto1[52]="UNIVERSIDADE GAMA FILHO - UGF";
char texto2[52]="COMPUTAÇÃO GRÁFICA";
char texto3[52]="Prof. Cláudio M. N. A. Pereira";
char texto4[52]="Transformações 2D - Trabalho A1 - Carro de Corrida";
char texto5[52]="Aluno Ruan Pedro Dias Seraphim Mat. 20101075779";
char texto6[52]="Aluno Sérgio da Silva Pereira  Mat. 20101609418";
char texto7 [4]="UGF";

//--------------------------------------------------------------------------
// Desenha Texto
//--------------------------------------------------------------------------
void DesenhaTexto(float x, float y, char *string)
{
    // Posição no universo onde o texto será colocado
    glRasterPos2f(x,y);
    glPushMatrix();
    // Exibe caracter a caracter
    while(*string){
         glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_10,*string++);
    }
    glPopMatrix();
}

//--------------------------------------------------------------------------
// Desenha Circulo Preenchido
//--------------------------------------------------------------------------
void cirulo(float x, float y, float r) {
    int i=0;
    float a=0;
    int p = 1000;

    glBegin(GL_POLYGON);
        for(i=0; i<p; i++)
        {
            glVertex2f((cos(a)*r)+x,(sin(a)*r)+y);
            a+=(360.0/p);
        }
    glEnd();
}

//--------------------------------------------------------------------------
// Inicializa
//--------------------------------------------------------------------------
void Init () {

       glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
       glOrtho (-100, 100, -100, 100, -100, 100);
```

```
}

//-------------------------------------------------------------------------
// Display
//-------------------------------------------------------------------------
void Display () {
       //-------------------------------------------------------------------------
       // Inicializa matrizes
       //-------------------------------------------------------------------------
       Init();

       //-------------------------------------------------------------------------
       // Limpa janela
       //-------------------------------------------------------------------------
    glClearColor (1, 1, 1, 1);                                  // Cor de fundo
    glClear(GL_COLOR_BUFFER_BIT);                               // Limpa janela

       //-------------------------------------------------------------------------
       // Traça eixos
       //-------------------------------------------------------------------------

       //-------------------------------------------------------------------------
    // Desenha linha da pista
    //-------------------------------------------------------------------------
    glColor3f (0, 0, 0);
    glBegin (GL_LINES);
       glVertex2f (-100,0);
       glVertex2f (100,0);
    glEnd();


    //-------------------------------------------------------------------------
    // Label
    //-------------------------------------------------------------------------
    glColor3f (1, 1, 1);
    glBegin (GL_QUADS);
        glColor3f (1, 1, 1);
        glVertex2f (-99, -99);
       glVertex2f (-30, -99);
       glVertex2f (-30, -72);
       glVertex2f (-99, -72);
       glEnd();

    glColor3f (0, 0, 0);
    DesenhaTexto(-95, -76, texto1);
    DesenhaTexto(-95, -80, texto2);
    DesenhaTexto(-95, -84, texto3);
    DesenhaTexto(-95, -88, texto4);
    DesenhaTexto(-95, -92, texto5);
    DesenhaTexto(-95, -96, texto6);

       //-------------------------------------------------------------------------
       // Escala
       //-------------------------------------------------------------------------
       glScalef(S, S, S);

       //-------------------------------------------------------------------------
       // Caminhão - Empilha matriz - para "Double Buffering"
       //-------------------------------------------------------------------------
    glPushMatrix();
        //-----------------------------------------------------------------
       // Translação
       //-----------------------------------------------------------------
        glTranslatef(Tx, Ty, 0);
       //-----------------------------------------------------------------
       // Desenha Caminhão com Janela
       //-----------------------------------------------------------------
        glBegin (GL_QUADS);
            glColor3f (1, 0, 0);

               glVertex2f (0, 1);
               glVertex2f (1, 1);
               glVertex2f (1, 2);
               glVertex2f (0, 2);

            glVertex2f (3, 1);
               glVertex2f (9, 1);
```

```
            glVertex2f (9, 2);
            glVertex2f (3, 2);

         glVertex2f (11, 1);
            glVertex2f (12, 1);
            glVertex2f (12, 2);
            glVertex2f (11, 2);

            glVertex2f (0, 2);
            glVertex2f (12, 2);
            glVertex2f (11, 3);
            glVertex2f (0, 3);

            glVertex2f (3, 3);
            glVertex2f (3.5, 3);
            glVertex2f (5, 4.5);
            glVertex2f (5, 5);

            glVertex2f (5, 4.5);
            glVertex2f (6.5, 4.5);
            glVertex2f (6.5, 5);
            glVertex2f (5, 5);

            glVertex2f (6.5, 3);
            glVertex2f (7, 3);
            glVertex2f (7, 5);
            glVertex2f (6.5, 5);

            glColor3f (0, 1, 1);

            glVertex2f (3.5, 3);
            glVertex2f (6.5, 3);
            glVertex2f (6.5, 4.5);
            glVertex2f (5, 4.5);

    glEnd();

    //-------------------------------------------------------------------------
    // Desenha Testo na porta
    //-------------------------------------------------------------------------
    glColor3f (1, 1, 0);
    DesenhaTexto(3.5, 1.5, texto7);

    //-------------------------------------------------------------------------
    // Desenha Rodas
    //-------------------------------------------------------------------------
    glColor3f (0, 0, 0);
    cirulo(2,1,1);
    cirulo(10,1,1);

    //-------------------------------------------------------------------------
    // Desenha Calotas
    //-------------------------------------------------------------------------
    glColor3f (1, 1, 0);
    cirulo(2,1,0.5);
    cirulo(10,1,0.5);

//-----------------------------------------------------------------------------
// Caminhão - Desempilha matriz - para "Double Buffering"
//-----------------------------------------------------------------------------
glPopMatrix();

   //-------------------------------------------------------------------------
   // Rotação J1
   //-------------------------------------------------------------------------
glTranslatef((Tx+8), (Ty+3), 0);        // p/ Rotação (centro)
glRotatef(J1, 0, 0, 1);
   glTranslatef(-(Tx+8), -(Ty+3), 0);      // p/ Rotação (centro)

//-----------------------------------------------------------------------------
   // Braço 1 - Empilha matriz - para "Double Buffering"
   //-------------------------------------------------------------------------
glPushMatrix();
    //-------------------------------------------------------------------------
    // Translação
    //-------------------------------------------------------------------------
    glTranslatef(Tx, Ty, 0);
```

```
        //-------------------------------------------------------------------------
        // Desenha Braço 1
        //-------------------------------------------------------------------------
        glBegin (GL_QUADS);
            glColor3f (0, 0, 1);

               glVertex2f (7.5, 3);
               glVertex2f (8.5, 3);
               glVertex2f (8.5, 8);
               glVertex2f (7.5, 8);
        glEnd();

        //-------------------------------------------------------------------------
        // Desenha Junção 1
        //-------------------------------------------------------------------------
        glColor3f (0, 1, 0);
        cirulo(8,3,0.5);

    //-------------------------------------------------------------------------
    // Braço 1 - Desempilha matriz - para "Double Buffering"
    //-------------------------------------------------------------------------
    glPopMatrix();

    //-------------------------------------------------------------------------
        // Rotação J2
        //-------------------------------------------------------------------------
        glTranslatef((Tx+8), (Ty+8), 0);        // p/ Rotação (centro)
    glRotatef(J2, 0, 0, 1);
        glTranslatef(-(Tx+8), -(Ty+8), 0);      // p/ Rotação (centro)

    //-------------------------------------------------------------------------
        // Braço 2 - Empilha matriz - para "Double Buffering"
        //-------------------------------------------------------------------------
    glPushMatrix();
        //-------------------------------------------------------------------------
        // Translação
        //-------------------------------------------------------------------------
        glTranslatef(Tx, Ty, 0);
        //-------------------------------------------------------------------------
        // Desenha Braço 2
        //-------------------------------------------------------------------------

        glBegin (GL_QUADS);
            glColor3f (0, 0, 1);

               glVertex2f (7.5, 8);
               glVertex2f (8.5, 8);
               glVertex2f (8.5, 13);
               glVertex2f (7.5, 13);
        glEnd();

        //-------------------------------------------------------------------------
        // Desenha Junção 2
        //-------------------------------------------------------------------------
        glColor3f (0, 1, 0);
        cirulo(8,8,0.5);

    //-------------------------------------------------------------------------
    // Braço 2 - Desempilha matriz - para "Double Buffering"
    //-------------------------------------------------------------------------
    glPopMatrix();

    glutSwapBuffers();

    glFlush();            // Descarrega buffer
}

//-------------------------------------------------------------------------
// Controle de Teclado
//-------------------------------------------------------------------------
void Keyboard (unsigned char key, int x, int y) {

        switch (key) {
        case 27:
    {
         exit(0);
```

```
          break;
    }
       case 'd':
    {
        if(Tx < 100) Tx = Tx + 1;  // move caminhão para direita
        else Tx = -112;            // move caminhão para a esquerda da tela
        break;
    }
       case 'e':
       {
        if(Tx > -112) Tx = Tx - 1; // move caminhão para esquerda
        else Tx = 100;             // move caminhão para a direita da tela
        break;
    }
       case 'c':
       {
        //Ty = Ty + 1;
        break;
    }
       case 'b':
       {
        //Ty = Ty - 1;
        break;
    }
       case 'r':
       {
        // Limita Braço 1 para não amassar cabine do caminhão
        if(J1 < 13) J1 = J1 + 1;
        break;
    }
    case 'f':
       {
        // Limita Braço 1 para não amassar Chassi do caminhão
        if(J1 > -65)J1 = J1 - 1;
        break;
    }
    case 't':
       {
        // Limita Braço 2 para não quebrar o vidro da cabine do caminhão
        if(J2 < 120) J2 = J2 + 1;
        break;
    }
    case 'g':
       {
        // Limita Braço 2 para não amassar Chassi do caminhão
        if(J2 > -90) J2 = J2 - 1;
        break;
    }
       case '-':
       {
        S = S * 0.95; // Zoom Out
        break;
    }
       case '+':
       {
        S = S * 1.05; // Zomm In
        break;
    }
       case ' ':
       {
        Tx=0, Ty=0, S=3, J1=-35, J2=-100;
        break;
    }


   }

// cout << endl << "Tx=" << Tx <<  "  Ty=" << Ty <<  "  R=" << R << endl;

   glutPostRedisplay();
}

//------------------------------------------------------------------------
// Principal
//------------------------------------------------------------------------
int main (int argc, char** argv) {
```

```
   // Inicializa glut
   glutInit (&argc, argv);

   // Inicializa modos de display (single buffer e cores RGB)
   glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);

   // Tamanho da janela
   glutInitWindowSize (700,700);

   // Posição inicial da janela
   glutInitWindowPosition (50,50);

   // Cria a janela
   glutCreateWindow ("UGF - Caminhão Guincho - A1");

   Init();

   // Executa função Display
   glutDisplayFunc (Display);

   // Executa função Display
   glutKeyboardFunc (Keyboard);

   // Entra em loop
   glutMainLoop();
}

//---------------------------------------------------------------------------
// Fim
//---------------------------------------------------------------------------
```

2 – Carro de Corrida

2.1 – Desenho

Desenho 2 – Carro de Corrida

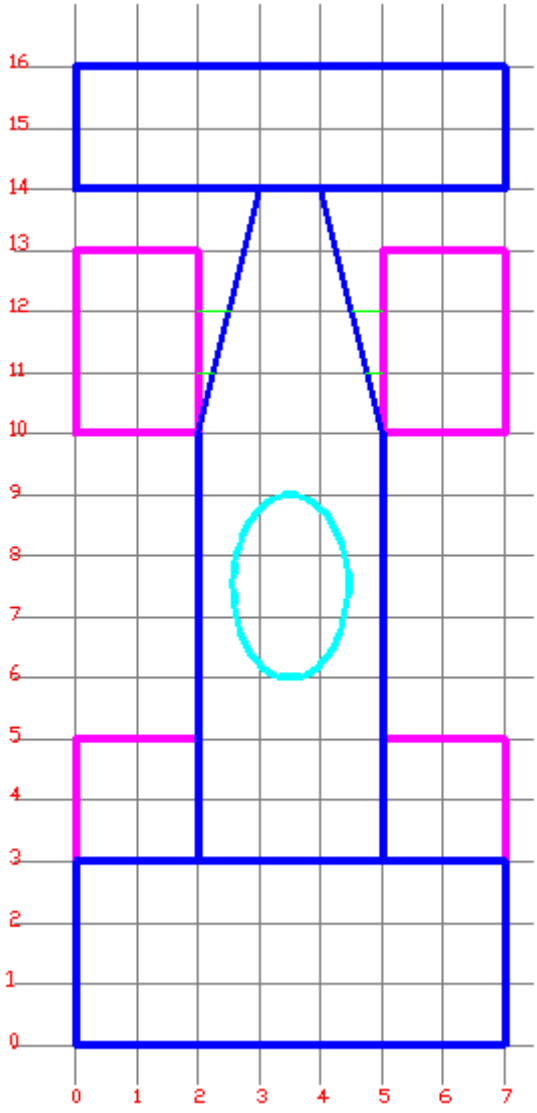
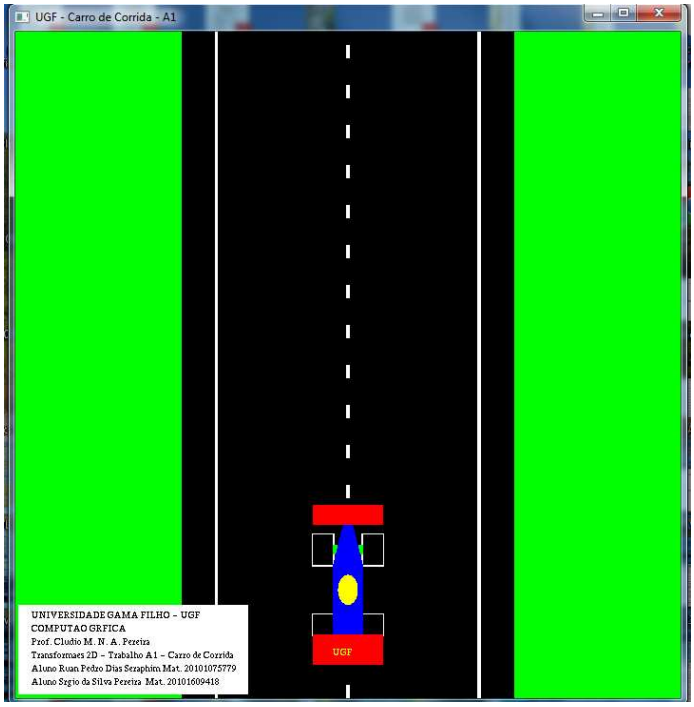


Figura 2 – Imagem do programa Carro de Corrida



2.2 – Código fonte

```
//--------------------------------------------------------------------------------
//
// UNIVERSIDADE GAMA FILHO - UGF
//
// COMPUTAÇÃO GRÁFICA
//
// Prof. Cláudio M. N. A. Pereira
//
// Transformações 2D - Trabalho A1 - Carro de Corrida
//
// Aluno Ruan Pedro Dias Seraphim Mat. 20101075779
//
// Aluno Sérgio da Silva Pereira  Mat. 20101609418
//
//--------------------------------------------------------------------------------

#include <windows.h>
#include <stdlib.h>
#include <math.h>
//#include <iostream.h>
#include <stdio.h>
#include <gl/gl.h>
#include <gl/glut.h>

#define PI 3.1415926535898

float Tx=-3.5, Ty=-30, R=0, S=3, z=0.15, m=0, k=0;
int v=0, vmax=20, vmin=-20, vscale=10, tm=0, tmin=1;

char texto1[52]="UNIVERSIDADE GAMA FILHO - UGF";
char texto2[52]="COMPUTAÇÃO GRÁFICA";
char texto3[52]="Prof. Cláudio M. N. A. Pereira";
char texto4[52]="Transformações 2D - Trabalho A1 - Carro de Corrida";
char texto5[52]="Aluno Ruan Pedro Dias Seraphim Mat. 20101075779";
char texto6[52]="Aluno Sérgio da Silva Pereira  Mat. 20101609418";
char texto7 [4]="UGF";

//--------------------------------------------------------------------------------
// Timer
//--------------------------------------------------------------------------------
void Timer(int value) {

   // Cálculo de aceleração
   tm=abs(abs(v*vscale)-(vmax*vscale));

   //Impede que se dê imprensão que inprensão que o sentido de movimento inverso
   if(tm<tmin)tm=tmin;

   // Configura o timer com tempo de rechamada
   glutTimerFunc(tm, Timer, 1);

   //Atualiza posição da faixa central da pista de rolamento simulando movimento
   if(v > 0) if(m >  0) m--; else m=12;
   if(v < 0) if(m < 12) m++; else m=0;

   // Redesenha a tela
   glutPostRedisplay();
}

//--------------------------------------------------------------------------------
// Desenha Texto
//--------------------------------------------------------------------------------
void DesenhaTexto(float x, float y, char *string)
{
    // Posição no universo onde o texto será colocado
    glRasterPos2f(x,y);
    glPushMatrix();
    // Exibe caracter a caracter
    while(*string){
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_10,*string++);
    }
    glPopMatrix();
}

//--------------------------------------------------------------------------------
```

```
// Desenha Circulo Preenchido
//-------------------------------------------------------------------------
void cirulo(float x, float y, float r) {
    int i=0;
    float a=0;
    int p = 1000;

    glBegin(GL_POLYGON);
        for(i=0; i<p; i++)
        {
            glVertex2f((cos(a)*r)+x,(sin(a)*r)+y);
            a+=(360.0/p);
        }
    glEnd();
}

//-------------------------------------------------------------------------
// Desenha Oval Preenchido
//-------------------------------------------------------------------------
void oval(float x, float y, float w, float h)
{
    float t, a;
    int n = 1000;
    a = PI / n;

    glBegin (GL_POLYGON);
        for (t = 0; t < 360; t += a)
        {
            glVertex2f (w/2 * cos (t)+x, h/2 * sin (t)+y);
        }
    glEnd ();
}

//-------------------------------------------------------------------------
// Inicializa
//-------------------------------------------------------------------------
void Init () {

       glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
       glOrtho (-100, 100, -100, 100, -100, 100);
}

//-------------------------------------------------------------------------
// Display
//-------------------------------------------------------------------------
void Display () {

       //-------------------------------------------------------------------------
       // Inicializa matrizes
       //-------------------------------------------------------------------------
       Init();

       //-------------------------------------------------------------------------
       // Limpa janela
       //-------------------------------------------------------------------------
    glClearColor (1, 1, 1, 1);                                  // Cor de fundo
    glClear(GL_COLOR_BUFFER_BIT);                               // Limpa janela

    //-------------------------------------------------------------------------
       // Desenha os canteiros laterais da Pista de Rolameto
       //-------------------------------------------------------------------------
       glColor3f (0, 1, 0);
    glBegin (GL_QUADS);
       glVertex2f (-100,-100);
       glVertex2f ( -50,-100);
       glVertex2f ( -50, 100);
       glVertex2f (-100, 100);

       glVertex2f (  50,-100);
       glVertex2f ( 100,-100);
       glVertex2f ( 100, 100);
       glVertex2f (  50, 100);
    glEnd();

       //-------------------------------------------------------------------------
       // Desenha da Pista de Rolameto
```

```
   //-------------------------------------------------------------------------
glColor3f (0, 0, 0);
glBegin (GL_QUADS);
   glVertex2f ( -50,-100);
   glVertex2f (  50,-100);
   glVertex2f (  50, 100);
   glVertex2f ( -50, 100);
glEnd();

//-------------------------------------------------------------------------
   // Desenho das faixas laterais da pista de rolamento
   //-------------------------------------------------------------------------
glColor3f (1, 1, 1);
glBegin (GL_QUADS);
   glVertex2f (-40,-100);
   glVertex2f (-39,-100);
   glVertex2f (-39, 100);
   glVertex2f (-40, 100);

   glVertex2f (40,-100);
   glVertex2f (39,-100);
   glVertex2f (39, 100);
   glVertex2f (40, 100);
glEnd();

//-------------------------------------------------------------------------
   // Desenha a dinâmica do movimento da faixa central da pista de rolamento
   //-------------------------------------------------------------------------
glBegin (GL_QUADS);
    for(k=-100; k<100; k=k+12)
    {
           glVertex2f (-0.5, k+m  );
           glVertex2f ( 0.5, k+m  );
           glVertex2f ( 0.5, k+m+4);
           glVertex2f (-0.5, k+m+4);
    }
glEnd();

//-------------------------------------------------------------------------
   // Label
   //-------------------------------------------------------------------------
glColor3f (1, 1, 1);
glBegin (GL_QUADS);
    glColor3f (1, 1, 1);
    glVertex2f (-99, -99);
   glVertex2f (-30, -99);
   glVertex2f (-30, -72);
   glVertex2f (-99, -72);
   glEnd();

glColor3f (0, 0, 0);
DesenhaTexto(-95, -76, texto1);
DesenhaTexto(-95, -80, texto2);
DesenhaTexto(-95, -84, texto3);
DesenhaTexto(-95, -88, texto4);
DesenhaTexto(-95, -92, texto5);
DesenhaTexto(-95, -96, texto6);

   //-------------------------------------------------------------------------
   // Escala
   //-------------------------------------------------------------------------
   glScalef(S, S, S);

   //-------------------------------------------------------------------------
   // Rotação Carro de Corrida
   //-------------------------------------------------------------------------
glTranslatef((Tx+3.5), (Ty+7.5), 0);        // p/ Rotação (centro)
glRotatef(R, 0, 0, 1);
   glTranslatef(-(Tx+3.5), -(Ty+7.5), 0);      // p/ Rotação (centro)

   //-------------------------------------------------------------------------
   // Carro de Corrida - Empilha matriz - para "Double Buffering"
   //-------------------------------------------------------------------------
glPushMatrix();
    //-----------------------------------------------------------------------
   // Translação
   //-----------------------------------------------------------------------
```

```
 glTranslatef(Tx, Ty, 0);
//-------------------------------------------------------------------------
// Desenha Carro de Corrida
//-------------------------------------------------------------------------
 glBegin (GL_QUADS);

     glColor3f (1, 1, 1);

     glVertex2f (0-z, 3-z);
        glVertex2f (2+z, 3-z);
        glVertex2f (2+z, 5+z);
        glVertex2f (0-z, 5+z);

        glVertex2f (5-z, 3-z);
        glVertex2f (7+z, 3-z);
        glVertex2f (7+z, 5+z);
        glVertex2f (5-z, 5+z);

        glVertex2f (0-z, 10-z);
        glVertex2f (2+z, 10-z);
        glVertex2f (2+z, 13+z);
        glVertex2f (0-z, 13+z);

        glVertex2f (5-z, 10-z);
        glVertex2f (7+z, 10-z);
        glVertex2f (7+z, 13+z);
        glVertex2f (5-z, 13+z);

     glColor3f (1, 0, 0);

        glVertex2f (0, 0);
        glVertex2f (7, 0);
        glVertex2f (7, 3);
        glVertex2f (0, 3);

     glVertex2f (0, 14);
        glVertex2f (7, 14);
        glVertex2f (7, 16);
        glVertex2f (0, 16);

        glColor3f (0, 1, 0);

        glVertex2f (2, 11);
        glVertex2f (3, 11);
        glVertex2f (3, 12);
        glVertex2f (2, 12);

        glVertex2f (4, 11);
        glVertex2f (5, 11);
        glVertex2f (5, 12);
        glVertex2f (4, 12);

        glColor3f (0, 0, 1);

     glVertex2f (2, 3);
        glVertex2f (5, 3);
        glVertex2f (5, 10);
        glVertex2f (2, 10);

        glVertex2f (2, 10);
        glVertex2f (5, 10);
        glVertex2f (4, 14);
        glVertex2f (3, 14);

     glColor3f (0, 0, 0);

        glVertex2f (0, 3);
        glVertex2f (2, 3);
        glVertex2f (2, 5);
        glVertex2f (0, 5);

        glVertex2f (5, 3);
        glVertex2f (7, 3);
        glVertex2f (7, 5);
        glVertex2f (5, 5);

        glVertex2f (0, 10);
```

```
                glVertex2f (2, 10);
                glVertex2f (2, 13);
                glVertex2f (0, 13);

                glVertex2f (5, 10);
                glVertex2f (7, 10);
                glVertex2f (7, 13);
                glVertex2f (5, 13);

            glEnd();

            //---------------------------------------------------------------------
            // Desenha Texto
            //---------------------------------------------------------------------
            glColor3f (1, 1, 0);
            DesenhaTexto(2, 1, texto7);

            //---------------------------------------------------------------------
            // Desenha Cockpit
            //---------------------------------------------------------------------
            glColor3f (1, 1, 0);
            oval(3.5, 7.5, 2, 3);

        //---------------------------------------------------------------------
        // Carro de Corrida - Desempilha matriz - para "Double Buffering"
        //---------------------------------------------------------------------
        glPopMatrix();

        glutSwapBuffers();

        glFlush();            // Descarrega buffer
}

//---------------------------------------------------------------------------
// Controle de Teclado
//---------------------------------------------------------------------------
void Keyboard (unsigned char key, int x, int y) {

            switch (key) {
            case 27:
        {
              exit(0);
              break;
        }
            case 'd':
        {
             if(Tx < 20) Tx = Tx + 1;
             break;
        }
            case 'e':
            {
             if(Tx > -27) Tx = Tx - 1;
             break;
        }
            case 'c':
            {
             if(Ty < 40)Ty = Ty + 1;
             else Ty = -50;
             break;
        }
            case 'b':
            {
             if(Ty > -50)Ty = Ty - 1;
             else Ty = 40;
             break;
        }
            case 'r':
            {
             R = R + 1;
             break;
        }
        case 'f':
            {
             R = R - 1;
             break;
        }
        case 't':
```

```
        {
         if(v < vmax) v = v + 1;
         break;
    }
    case 'g':
        {
         if(v > vmin) v = v - 1;
         break;
    }
        case '-':
        {
         S = S * 0.95; // Zoom Out
         break;
    }
        case '+':
        {
         S = S * 1.05; // Zomm In
         break;
    }
        case ' ':
        {
         Tx=-3.5, Ty=-30, S=3, R=0, v=0;
         break;
    }
   }

//   cout << endl << "Tx=" << Tx <<  "  Ty=" << Ty <<  "  R=" << R << endl;

   glutPostRedisplay();
}

//------------------------------------------------------------------------
// Principal
//------------------------------------------------------------------------
int main (int argc, char** argv) {
    tm=vscale*vmax;

        // Inicializa glut
        glutInit (&argc, argv);

        // Inicializa modos de display (single buffer e cores RGB)
   glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);

   // Tamanho da janela
   glutInitWindowSize (700,700);

   // Posição inicial da janela
   glutInitWindowPosition (50,50);

   // Cria a janela
        glutCreateWindow ("UGF - Carro de Corrida - A1");
   Init();

   // Executa função Display
   glutDisplayFunc (Display);

   // Executa função Display
   glutKeyboardFunc (Keyboard);

   //glutIdleFunc(Display);

   glutTimerFunc(tm, Timer, 1);

   // Entra em loop
   glutMainLoop();
}
//------------------------------------------------------------------------
// Fim
//------------------------------------------------------------------------
```

3 - Referências

Anotações do caderno feitas em aulas de Computação Gráfica ministradas pelo prof. Cláudio Márcio na Universidade Gama Filho e materiais fornecidos também em aula.