# Metadados

- ***Autor: Raphael Lira dos Santos | RA: 223865***
- ***Autor: Matheus Percário Bruder | RA: 222327***
- *Criado em: 29/04/2021*
- *Arquivo: "main.c"*

# Projeto Aeroporto

## Descrição

Modele via OpenGL uma cena gráfica que contenha a simulação de um aeroporto. Pesquise referências antes de iniciar a modelagem: aeroportos modelados em desenhos animados infantis são bons exemplos devido à sua simplicidade. A cena deve utilizar ao menos 3 tipos diferentes de elementos geométricos.

Use objetos sólidos na modelagem. Use também teclas (ou mouse) para permitir a interação do usuário com a cena, permitindo a sua rotação com relação ao eixo vertical e o efeito zoom. Tente fazer em escala (escolha as dimensões). Deve ser entregue o código `fonte.c` e um `print` da cena (2 arquivos).

**Observação:** *Tenha também em mente que este exercício será a base para os exercícios seguintes, que consistem na inclusão de textura e iluminação na cena.*

### Explicação

Antes de mais nada, nossa cena gráfica tenta representar um aeroporto de grande porte, incluindo duas pistas, vários hangares (galpões em que os aviões são guardados), um portão de embarque com várias rampas, uma torre de controle e, por fim, um avião (teco-teco).

Desse modo, a fim de atender os requisitos propostos pelo professor, os diferentes tipos de **elementos geométricos** escolhidos, foram:

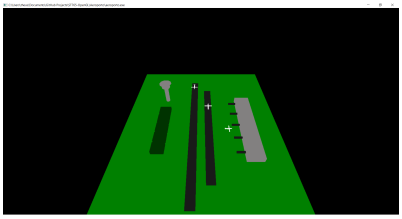
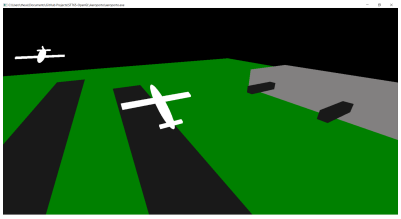
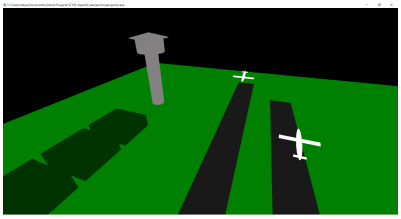
- **Cubo:** *Solo, pistas, galpões, portões de embarque, asas dos aviões;*
- **Esfera:** *Corpo dos aviões;*
- **Cilindro:** *Torre de controle;*
- **Cone:** *Cobertura da torre de controle.*

Além disso, o usuário pode interagir com a cena gráfica rotacionando no eixo X e no eixo Y. Isso pode ser um pouco confuso no início, mas ajuda muito na visualização do aeroporto.

Ademais, foi preciso também proporcionar ao usuário a possibilidade de dar "*zoom*" na cena gráfica. A seguir estão dispostos os comandos para interação com a cena gráfica:

- **Rotacionar no eixo X:** *W ou S;*
- **Rotacionar no eixo Y:** *A ou D;*
- **Zoom:** *Scroll do mouse.*

### Prints da cena:

| Zoom out | Zoom in | Torre e avião |
| --- | --- | --- |
|  |  |  |

## Código fonte

```
#include<stdlib.h>
#include<GL/glut.h>

/**
*  Medidas dadas em Metros
*/
#define CHAO_ALTURA 0.1

#define AVIAO_TRANSLACAO_X -100.0
#define AVIAO_TRANSLACAO_Y -50.0
#define AVIAO_TRANSLACAO_Z -700.0
#define CHAO_ESCALONAMENTO_X 2000.0
#define CHAO_ESCALONAMENTO_Z 2500.0
#define PISTA_TRANSLACAO_Y 0.0
#define PISTA_TRANSLACAO_Z 0.0
#define PISTA_ESCALONAMENTO_X 100.0
#define PISTA_ESCALONAMENTO_Y 0.2
#define GALPAO_ESCALONAMENTO_X 150.0
#define GALPAO_ESCALONAMENTO_Y 50.5
#define GALPAO_ESCALONAMENTO_Z 130.0
#define GALPAO_TRANSLACAO_X -500
#define GALPAO_ESPACAMENTO 20
#define PORTOES_ESCALONAMENTO_X 200.0
#define PORTOES_ESCALONAMENTO_Y 70.0
#define PORTOES_ESCALONAMENTO_Z 1000.0
#define PORTOES_TRANSLACAO_X 600.0
#define PORTAO_ESCALONAMENTO_X 100
#define PORTAO_ESCALONAMENTO_Y 20
#define PORTAO_ESCALONAMENTO_Z 20
```

```
#define PORTAO_TRANSLACAO_X PORTOES_TRANSLACAO_X - (PORTOES_ESCALONAMENTO_X/2) - (PORTAO
#define TORRE_CONTROLE_BASE_RAIO 30
#define TORRE_CONTROLE_BASE_ALTURA 250
#define TORRE_CONTROLE_BASE_SLICES 10
#define TORRE_CONTROLE_BASE_STACKS 10
#define TORRE_CONTROLE_BASE_TRANSLACAO_X GALPAO_TRANSLACAO_X
#define TORRE_CONTROLE_BASE_TRANSLACAO_Z 0.0
#define TORRE_CONTROLE_CABINE_RAIO 60
#define TORRE_CONTROLE_CABINE_ALTURA 60
#define TORRE_CONTROLE_CABINE_SLICES 10
#define TORRE_CONTROLE_CABINE_STACKS 10
#define TORRE_CONTROLE_CABINE_TRANSLACAO_X TORRE_CONTROLE_BASE_TRANSLACAO_X
#define TORRE_CONTROLE_CABINE_TRANSLACAO_Z 0.0
#define TORRE_CONTROLE_TELHADO_ALTURA 20.0
#define TORRE_CONTROLE_TELHADO_MARGEM 1.4


static int rotx = 45;
static int roty = 0;

static float zoom = -1500.0;

// TODO:
// 1) adicionar preenchimento
// 2) adicionar cores?
// **) tetos galpoes

/**
* Desenha cada port□a de embarque
*
* @param quantidades_portoes recebe a quantidade de port□es que ser□o criados para gara
* @param port□o o n□mero do port□o pelo qual ser□ impresso
*/
void desenhar_portao_embarque(int quantidades_portoes, int portao){
 glPushMatrix();

 glColor3f(0.1f, 0.1f, 0.1f);//Dark grey

 // transla□□o de Y
 float t_z =
    ((PORTOES_ESCALONAMENTO_Z/quantidades_portoes) * portao)
        - ((quantidades_portoes/2) * (PORTOES_ESCALONAMENTO_Z/quantidades_portoes));

 glTranslatef (PORTAO_TRANSLACAO_X, CHAO_ALTURA + (PORTOES_ESCALONAMENTO_Y/2), t_z);
 glScalef (PORTAO_ESCALONAMENTO_X, PORTAO_ESCALONAMENTO_Y, PORTAO_ESCALONAMENTO_Z);
 glutSolidCube (1.0);

 glPopMatrix();
}


/**
* Desenha a torre de controle
*
* @param quantidade_galpoes recebe a quantidade de galpoes criados para gerar espa□aame
```

```
*/
void desenhar_torre_controle(int quantidade_galpoes){

 // base da torre
 glPushMatrix();

 glColor3f(0.51f, 0.5f, 0.5f);//Grey

 float
    t_z = - ((quantidade_galpoes/2) * (GALPAO_ESCALONAMENTO_Z * 2));

 glTranslatef (
     (GLfloat) TORRE_CONTROLE_BASE_TRANSLACAO_X,
     (GLfloat) (CHAO_ALTURA + (TORRE_CONTROLE_BASE_ALTURA)),
     (GLfloat) t_z
 );
 glRotatef ((GLfloat) 90, 1.0, 0.0, 0.0);
 glutSolidCylinder(
      (GLdouble) TORRE_CONTROLE_BASE_RAIO,
     (GLdouble) TORRE_CONTROLE_BASE_ALTURA,
     (GLint) TORRE_CONTROLE_BASE_SLICES,
     (GLint) TORRE_CONTROLE_BASE_STACKS
 );

 glPopMatrix();

 // base onde fica os controladores
 glPushMatrix();
 glTranslatef (
     (GLfloat) TORRE_CONTROLE_CABINE_TRANSLACAO_X,
     (GLfloat) (CHAO_ALTURA + TORRE_CONTROLE_CABINE_ALTURA + TORRE_CONTROLE_BASE_ALTURA)
     (GLfloat) t_z
 );
 glRotatef ((GLfloat) 90, 1.0, 0.0, 0.0);
 glutSolidCylinder(
      (GLdouble) TORRE_CONTROLE_CABINE_RAIO,
     (GLdouble) TORRE_CONTROLE_CABINE_ALTURA,
     (GLint) TORRE_CONTROLE_CABINE_SLICES,
     (GLint) TORRE_CONTROLE_CABINE_STACKS
 );

 glPopMatrix();

 // telhado
 glPushMatrix();
 glTranslatef (
     (GLfloat) TORRE_CONTROLE_CABINE_TRANSLACAO_X,
     (GLfloat) (CHAO_ALTURA + TORRE_CONTROLE_CABINE_ALTURA + TORRE_CONTROLE_BASE_ALTURA)
     (GLfloat) t_z
 );
 glRotatef ((GLfloat) -90, 1.0, 0.0, 0.0);
 glutSolidCone(
      (GLdouble) TORRE_CONTROLE_CABINE_RAIO * TORRE_CONTROLE_TELHADO_MARGEM,
     (GLdouble) TORRE_CONTROLE_TELHADO_ALTURA,
     (GLint) TORRE_CONTROLE_CABINE_SLICES,
```

```
    (GLint) TORRE_CONTROLE_CABINE_STACKS
 );

 glPopMatrix();
}


/**
* Desenha os port�es onde haver� cada port�o de embarque
*/
void desenhar_portoes(){
 glPushMatrix();

 glColor3f(0.51f, 0.5f, 0.5f);//Grey

 glTranslatef (PORTOES_TRANSLACAO_X, CHAO_ALTURA + (PORTOES_ESCALONAMENTO_Y/2), 0.0);
 glScalef (PORTOES_ESCALONAMENTO_X, PORTOES_ESCALONAMENTO_Y, PORTOES_ESCALONAMENTO_Z);
 glutSolidCube(1.0);

 glPopMatrix();

 int
    i,
    quantidade_portoes = 5;// deixei 5 por default, mas poder� ser mudado para quantos

 for(i=0;i<quantidade_portoes;i++){
    desenhar_portao_embarque(quantidade_portoes,i);
 }
}

/**
* Desenha um galp�o de acordo com os par�metros
*
* @param t_x faz a transla��o do eixo X
* @param t_y faz a transla��o do eixo Y
* @param t_z faz a transla��o do eixo Z
*/
void desenhar_galpao(int quantidade_galpoes, int galpao){
 // transla��o de Y
 float t_z = ((GALPAO_ESCALONAMENTO_Z + GALPAO_ESPACAMENTO) * galpao) - ((quantidade_gal


 // Desenha teto do galp�o
 //  glPushMatrix();
 //  glTranslatef (GALPAO_TRANSLACAO_X - GALPAO_ESCALONAMENTO_X/2, CHAO_ALTURA + (GALPAO
 //  glRotatef (90, 0.0, 1.0, 0.0);
 //  glutSolidCylinder(GALPAO_ESCALONAMENTO_X/2 - 4, GALPAO_ESCALONAMENTO_X, 20, 20);
 //  glPopMatrix();


 // Desenha o galp�o
 glPushMatrix();

 glColor3f(0.0f, 0.2f, 0.0f);//Forest Green
```

```
 glTranslatef (GALPAO_TRANSLACAO_X, CHAO_ALTURA + (GALPAO_ESCALONAMENTO_Y/2), t_z);
 glScalef (GALPAO_ESCALONAMENTO_X, GALPAO_ESCALONAMENTO_Y, GALPAO_ESCALONAMENTO_Z);
 glutSolidCube (1.0);
 glPopMatrix();
}

/**
* Desenha uma pista de acordo com os par□metros
*
* @param t_x faz a transla□□o do eixo X
* @param e_z faz o escalonamento do eixto Z
*/
void desenhar_pista(float t_x, float e_z){
 glPushMatrix();

 glColor3f(0.1f, 0.1f, 0.1f);//Dark grey

 glTranslatef (t_x, PISTA_TRANSLACAO_Y + 0.5, PISTA_TRANSLACAO_Z);
 glScalef (PISTA_ESCALONAMENTO_X, PISTA_ESCALONAMENTO_Y, e_z);
 glutSolidCube (1.0);

 glPopMatrix();
}

/**
* Desenha o chao da cena de acordo com os par□metros
*/
void desenhar_chao(){
 glPushMatrix();

 glColor3f(0.0f, 0.5f, 0.0f);//Dark-Green

 glScalef (CHAO_ESCALONAMENTO_X, CHAO_ALTURA, CHAO_ESCALONAMENTO_Z);
 glutSolidCube (1.0);

 glPopMatrix();
}

/**
* Desenha o aviao
*
* @param rot_x faz a rotacao no eixo X (pousando / decolando)
* @param rot_y faz a rotacao no eixo Y ()
* @param t_x faz a transla□□o do eixo X
* @param t_y faz a transla□□o do eixo Y
* @param t_z faz a transla□□o do eixo Z
*/
void desenhar_aviao(int rot_x, int rot_y, float t_x, float t_y, float t_z){

 glPushMatrix();

 glColor4f(1.0f, 1.0f, 1.0f, 0.0f);//white

 // rotacao do aviao
 glRotatef (rot_x, 1.0, 0.0, 0.0);
```

```
 glRotatef (rot_y, 0.0, 1.0, 0.0);


 // Esfera -> corpo do corpo
 glPushMatrix();
 glTranslatef (t_x, t_y, t_z);
 glScalef (1.0, 1.0, 7.0);
 glutSolidSphere (7.0, 16, 16);
 glPopMatrix();

 // Asas tras
 glPushMatrix();
 glTranslatef (t_x, t_y, t_z - 40);
 glScalef (30.0, 3.0, 5.0);
 glutSolidCube (1.0);
 glPopMatrix();

 // Asas frente
 glPushMatrix();
 glTranslatef (t_x, t_y, t_z + 10);
 glScalef (100.0, 3.0, 10.0);
 glutSolidCube (1.0);
 glPopMatrix();

 // Asas (em pe)
 glPushMatrix();
 glTranslatef (t_x, t_y + 7, t_z - 40);
 glScalef (2.0, 7.0, 10.0);
 glutSolidCube (1.0);
 glPopMatrix();

 glPopMatrix();
}

void init(void){
 glClearColor (0.0, 0.0, 0.0, 0.0);
 glEnable(GL_DEPTH_TEST);
}

void display(void){
 // glClear (GL_COLOR_BUFFER_BIT);
 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
 glPushMatrix();

 glRotatef ((GLfloat) rotx, 1.0, 0.0, 0.0);
 glRotatef ((GLfloat) roty, 0.0, 1.0, 0.0);

 glTranslatef (0.0, zoom, zoom);

 // Chao da cena gr�fica
 desenhar_chao();

 // congonhas tem duas pistas:
 // 1940x45 e 1495x45
 desenhar_pista(-100.0,1940.0);
```

```
 desenhar_pista(100.0,1495.0);

 // galp?es/hagares
 int
    qtd_galpoes = 5,
    i;
 for(i= 0; i < qtd_galpoes; i++){
    desenhar_galpao(qtd_galpoes, i);
 }
 // Portoes onde ficarao os portoes de embarque
 desenhar_portoes();

 // Onde ficara a torre de controle
 desenhar_torre_controle(qtd_galpoes);

 // aviao pousando
 desenhar_aviao(15, 0, -100.0, -50.0, -700.0);

 // aviao decolando
 desenhar_aviao(20, 180, -100, -10.0, 300.0);

 // aviao portao de embarque
 desenhar_aviao(0, 90, -50.0, PORTAO_ESCALONAMENTO_Y, 350.0);

 glPopMatrix();
 glutSwapBuffers();
}

void reshape (int w, int h){
 glViewport (0, 0, (GLsizei) w, (GLsizei) h);
 glMatrixMode (GL_PROJECTION);
 glLoadIdentity ();
 gluPerspective(65.0, (GLfloat) w/(GLfloat) h, 1.0, 5000.0);
 glMatrixMode(GL_MODELVIEW);
 glLoadIdentity();
 glTranslatef (0.0, -200.0, 0.0);
}

void keyboard(unsigned char key, int x, int y){

 switch (key) {
 case 'a':
 case 'A':
   roty = (roty - 5) % 360;
   glutPostRedisplay();
   break;
 case 'd':
 case 'D':
   roty = (roty + 5) % 360;
   glutPostRedisplay();
   break;
  case 'w':
  case 'W':
   rotx = (rotx - 5) % 360;
   glutPostRedisplay();
```

```
    break;
  case 's':
  case 'S':
    rotx = (rotx + 5) % 360;
    glutPostRedisplay();
    break;
  case 27:                                          // tecla Esc (encerra o programa)
    exit(0);
    break;
  }
}

void mouse(int button, int state, int x, int y)
{
   if (state == GLUT_DOWN) {
       switch(button) {
    // Button 3 == SCROLL UP and Button 4  == SCROLL DOWN
       case 3:
         zoom += 30.0;
         glutPostRedisplay();
         break;
       case 4:
         zoom -= 30.0;
         glutPostRedisplay();
         break;
       default:
         break;
       }
     }
}

int main(int argc, char** argv){
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
  glutInitWindowSize (500, 500);
  glutInitWindowPosition (100, 100);
  glutCreateWindow (argv[0]);
  init ();
  glutDisplayFunc(display);
  glutReshapeFunc(reshape);
  glutKeyboardFunc(keyboard);
  glutMouseFunc(mouse);
  glutMainLoop();
  return 0;
}
```