**Exercicio Helloworld, el quadrado**

**#include "glm.h"**

**void displayMe(void)**

**{**

**glClear(GL\_COLOR\_BUFFER\_BIT);**

**glColor3f(1.0, 0.0, 0.0);**

**glBegin(GL\_POLYGON);**

**glVertex3f(0.0, 0.0, 0.0);**

**glVertex3f(0.7, 0.0, 0.0);**

**glVertex3f(0.7, 0.7, 0.0);**

**glVertex3f(0.0, 0.7, 0.0);**

**glEnd();**

**glFlush();**

**}**

**int main(int argc, char\*\* argv)**

**{**

**glutInit(&argc, argv);**

**glutInitDisplayMode(GLUT\_SINGLE);**

**glutInitWindowSize(1000, 300);**

**glutInitWindowPosition(100, 100);**

**glutCreateWindow("Hello world");**

**glClearColor(0.5, 0.5, 1.0, 1.0);**

**glutDisplayFunc(displayMe);**

**glutMainLoop();**

**return 0;**

**}**

**Ficha 0 – ex 1 (inserir números e ver o maior)**

**#include <GL/glut.h>**

**#include <stdio.h>**

**float num1, num2, result;**

**char buffer[50];**

**void display() {**

**glClear(GL\_COLOR\_BUFFER\_BIT);**

**glColor3f(1.0, 1.0, 1.0);**

**// Exibe a soma dos números**

**sprintf(buffer, "Soma: %.2f", result);**

**glRasterPos2f(-0.8, 0.2); //define a posiçao atual do desenho de pixeis**

**for (int i = 0; buffer[i] != '\0'; i++) {**

**glutBitmapCharacter(GLUT\_BITMAP\_9\_BY\_15, buffer[i]);**

**}**

**// Exibe o maior número**

**if (num1 > num2) {**

**sprintf(buffer, "Maior: %.2f", num1);**

**} else {**

**sprintf(buffer, "Maior: %.2f", num2);**

**}**

**glRasterPos2f(-0.8, 0.0);**

**for (int i = 0; buffer[i] != '\0'; i++) {**

**glutBitmapCharacter(GLUT\_BITMAP\_9\_BY\_15, buffer[i]);**

**}**

**glFlush();**

**}**

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

**if (key == 27) { // Tecla Esc para sair**

**exit(0);**

**}**

**}**

**void specialKeys(int key, int x, int y) {**

**if (key == GLUT\_KEY\_F1) {**

**result = num1 + num2;**

**glutPostRedisplay();**

**}**

**}**

**int main(int argc, char\*\* argv) {**

**printf("Digite o primeiro número: ");**

**scanf("%f", &num1);**

**printf("Digite o segundo número: ");**

**scanf("%f", &num2);**

**result = num1 + num2;**

**glutInit(&argc, argv);**

**glutInitWindowSize(400, 300);**

**glutCreateWindow("Soma e Maior Número");**

**glClearColor(0.0, 0.0, 0.0, 0.0);**

**glutDisplayFunc(display);**

**glutKeyboardFunc(keyboard);**

**glutSpecialFunc(specialKeys);**

**glutMainLoop();**

**return 0;**

**}**

**Ficha 0 – ex 2 (pede n números ao user e ordena)**

**#include <GL/glut.h>**

**#include <stdio.h>**

**#define MAX\_SIZE 100**

**float num1;**

**char buffer[50];**

**float arr[MAX\_SIZE];**

**void display() {**

**glClear(GL\_COLOR\_BUFFER\_BIT);**

**glColor3f(1.0, 1.0, 1.0);**

**//order arr in ascending order**

**for(int i = 0; i < num1; i++){**

**for(int j = i + 1; j < num1; j++){**

**if(arr[i] > arr[j]){**

**float temp = arr[i];**

**arr[i] = arr[j];**

**arr[j] = temp;**

**}**

**}**

**}**

**// Exibe os valores de arr dos números**

**for (int i = 0; i < num1; i++) {**

**sprintf(buffer, "%.2f", arr[i]);**

**glRasterPos2f(-0.8, 0.8 - i \* 0.2);**

**for (int i = 0; buffer[i] != '\0'; i++) {**

**glutBitmapCharacter(GLUT\_BITMAP\_9\_BY\_15, buffer[i]);**

**}**

**}**

**glFlush();**

**}**

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

**if (key == 27) { // Tecla Esc para sair**

**exit(0);**

**}**

**}**

**void specialKeys(int key, int x, int y) {**

**if (key == GLUT\_KEY\_F1) {**

**glutPostRedisplay();**

**}**

**}**

**int main(int argc, char\*\* argv) {**

**printf("Digite um número: ");**

**scanf("%f", &num1);**

**for(int i = 0; i < num1; i++){**

**printf("Digite um número: ");**

**scanf("%f", &arr[i]);**

**}**

**glutInit(&argc, argv);**

**glutInitWindowSize(400, 300);**

**glutCreateWindow("Soma e Maior Número");**

**glClearColor(0.0, 0.0, 0.0, 0.0);**

**glutDisplayFunc(display);**

**glutKeyboardFunc(keyboard);**

**glutSpecialFunc(specialKeys);**

**glutMainLoop();**

**return 0;**

**}**

**Ficha 0 – ex 3 (verifica se numero e primo)**

**#include <GL/glut.h>**

**#include <stdio.h>**

**#define MAX\_SIZE 100**

**int num1;**

**char buffer[50];**

**float arr[MAX\_SIZE];**

**void display() {**

**glClear(GL\_COLOR\_BUFFER\_BIT);**

**glColor3f(1.0, 1.0, 1.0);**

**//verifica se num1 é primo e escreve no glut primo se for e não primo se não for**

**int i, j, primo = 1;**

**for (i = 2; i < num1; i++) {**

**if (num1 % i == 0) {**

**primo = 0;**

**break;**

**}**

**}**

**if (primo == 1) {**

**sprintf(buffer, "O numero %d primo", num1);**

**} else {**

**sprintf(buffer, "O numero %d nao e primo", num1);**

**}**

**glRasterPos2f(-0.5, 0.5);**

**for (j = 0; buffer[j] != '\0'; j++) {**

**glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24, buffer[j]);**

**}**

**glFlush();**

**}**

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

**if (key == 27) { // Tecla Esc para sair**

**exit(0);**

**}**

**}**

**void specialKeys(int key, int x, int y) {**

**if (key == GLUT\_KEY\_F1) {**

**glutPostRedisplay();**

**}**

**}**

**int main(int argc, char\*\* argv) {**

**printf("Digite um número: ");**

**scanf("%d", &num1);**

**glutInit(&argc, argv);**

**glutInitWindowSize(400, 300);**

**glutCreateWindow("Soma e Maior Número");**

**glClearColor(0.0, 0.0, 0.0, 0.0);**

**glutDisplayFunc(display);**

**glutKeyboardFunc(keyboard);**

**glutSpecialFunc(specialKeys);**

**glutMainLoop();**

**return 0;**

**}**

**Ficha 0/Ficha1 – ex 4.. (exercício para desenhar qualquer poligno, apenas muda o ‘n’)**

**#include <GL/glut.h>**

**#include <stdio.h>**

**#include <math.h>**

**struct ponto**

**{**

**float x, y;**

**};**

**void desenhar\_circunferencia (){**

**glClear(GL\_COLOR\_BUFFER\_BIT);**

**glColor3f(1.0, 1.0, 1.0);**

**int n = 4;**

**struct ponto p0, p1;**

**p0.x = p0.y = 0.0;**

**p1.x = p1.y = 0.8;**

**//calcular ponto médio de p0 e p1**

**struct ponto c;**

**c.x = (p0.x + p1.x) / 2;**

**c.y = (p0.y + p1.y) / 2;**

**//calcular diametro de p0 e p1**

**float diametro = sqrt(pow(p1.x - p0.x, 2) + pow(p1.y - p0.y, 2));**

**//calcular raio**

**float raio = diametro / 2;**

**struct ponto arr[20];**

**//calcular pontos da circunferencia**

**float aDim = 360/n;**

**float angulo = 360;**

**for (int i = 0; i < n; i++) {**

**float radianos = (angulo\*3.14f/180);**

**arr[i].x = c.x + raio \* cos(radianos);**

**arr[i].y = c.y + raio \* sin(radianos);**

**angulo -= aDim;**

**}**

**//desenhar circunferencia**

**glBegin(GL\_POLYGON);**

**for (int i = 0; i < n; i++) {**

**glVertex3f(arr[i].x, arr[i].y, 0.0);**

**}**

**glEnd();**

**glFlush();**

**}**

**int main(int argc, char\*\* argv) {**

**glutInit(&argc, argv);**

**glutInitWindowSize(800, 500);**

**glutCreateWindow("Soma e Maior Número");**

**glClearColor(0.0, 0.0, 0.0, 0.0);**

**glutDisplayFunc(desenhar\_circunferencia);**

**glutMainLoop();**

**return 0;**

**}**

**Ficha 2 – ex relogio**

**#include <stdio.h>**

**#include <string.h>**

**#include <stdlib.h>**

**#include <math.h>**

**#include <time.h>**

**#if defined(\_\_APPLE\_\_) || defined(MACOSX)**

**#include <GLUT/glut.h>**

**#else**

**#include <GL/glut.h>**

**#endif**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\* CONSTANTE PI \*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**#ifndef M\_PI**

**#define M\_PI 3.1415926535897932384626433832795**

**#endif**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* AUXILIARES CONVERSÃO GRAUS-RADIANOS \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**#define rtd(x) (180 \* (x) / M\_PI)**

**#define dtr(x) (M\_PI \* (x) / 180)**

**#define DEBUG 1**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\* VARIÁVEIS GLOBAIS \*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**typedef struct**

**{**

**GLboolean doubleBuffer;**

**GLint delay;**

**} Estado;**

**typedef struct**

**{**

**GLint hor, min, seg;**

**} Horas;**

**typedef struct**

**{**

**GLint numLados;**

**GLfloat raio;**

**GLfloat tamLado;**

**Horas hora;**

**} Modelo;**

**Estado estado;**

**Modelo modelo;**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\* FUNÇÕES \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**void poligono(GLint n, GLfloat x0, GLfloat y0, GLfloat r){ //FIZEMOS PRO EXERCICIO**

**glColor3f(1.0f, 1.0f, 1.0f);**

**GLfloat aDim = 360/n;**

**GLfloat angulo = 45;**

**GLfloat x, y;**

**glBegin(GL\_POLYGON);**

**for (int i = 0; i < n; i++)**

**{**

**GLfloat radianos = (angulo\*3.14f/180);**

**x = x0 + r \* cos(radianos);**

**y = y0 + r \* sin(radianos);**

**glVertex2f(x, y);**

**angulo += aDim;**

**}**

**glEnd();**

**}**

**void mostrador(){ // FIZEMOS PRO EXERCICIO**

**glColor3f(0.0f, 0.0f, 0.0f);**

**GLfloat aDim = 360/60;**

**GLfloat angulo = 0;**

**GLfloat x, y, x2, y2;**

**glLineWidth(1.0);**

**for (int i = 0; i < 60; i++)**

**{**

**GLfloat comprimento = (i%5 == 0) ? 0.05 : 0.02;**

**GLfloat radianos = (angulo\*M\_PI/180);**

**x = modelo.raio \* cos(radianos);**

**y = modelo.raio \* sin(radianos);**

**x2 = (modelo.raio - comprimento) \* cos(radianos);**

**y2 = (modelo.raio - comprimento) \* sin(radianos);**

**glBegin(GL\_LINES);**

**glVertex2f(x, y);**

**glVertex2f(x2, y2);**

**glEnd();**

**angulo += aDim;**

**}**

**}**

**void ponteiros(){ // FIZEMOS PRO EXERCICIO**

**GLfloat angulo\_segundos = 360 + 90.0 - modelo.hora.seg \* 6; // 360 graus divididos por 60 segundos**

**GLfloat angulo\_minutos = 360 + 90.0 - (modelo.hora.min6); // 360 graus divididos por 60 minutos**

**GLfloat angulo\_horas = 360 + 90.0 - (modelo.hora.hor30 + (modelo.hora.min \* 0.5)); // 360 graus divididos por 12 horas glLineWidth(1.0);**

**//segundos**

**glBegin(GL\_LINES);**

**glColor3f(1.0f, 0.0f, 0.0f);**

**glVertex2f(0, 0);**

**glVertex2f((0.6) \* cos(angulo\_segundos \* 3.14f / 180), (0.6) \* sin(angulo\_segundos \* 3.14f / 180));**

**glEnd();**

**//minutos**

**glBegin(GL\_LINES);**

**glColor3f(0.0f, 1.0f, 0.0f);**

**glVertex2f(0, 0);**

**glVertex2f((0.4) \* cos(angulo\_minutos \* 3.14f / 180), (0.4) \* sin(angulo\_minutos \* 3.14f / 180));**

**glEnd();**

**//horas**

**glBegin(GL\_LINES);**

**glColor3f(0.0f, 0.0f, 1.0f);**

**glVertex2f(0, 0);**

**glVertex2f((0.2) \* cos(angulo\_horas \* 3.14f / 180), (0.2) \* sin(angulo\_horas \* 3.14f / 180));**

**glEnd();**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\* INICIALIZAÇÃO DO AMBIENTE OPENGL \*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**void init(void)**

**{**

**struct tm \*current\_time;**

**time\_t timer = time(0);**

**/\* Delay para o timer \*/**

**estado.delay = 1000;**

**modelo.tamLado = 1;**

**modelo.numLados = 60;**

**modelo.raio = 0.75;**

**/\* Ler hora do Sistema \*/**

**current\_time = localtime(&timer);**

**modelo.hora.hor = current\_time->tm\_hour;**

**modelo.hora.min = current\_time->tm\_min;**

**modelo.hora.seg = current\_time->tm\_sec;**

**glClearColor(0.0, 0.0, 0.0, 0.0);**

**glEnable(GL\_POINT\_SMOOTH);**

**glEnable(GL\_LINE\_SMOOTH);**

**glEnable(GL\_POLYGON\_SMOOTH);**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\* CALL BACKS DE JANELA/DESENHO \*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* Callback para redimensionar janela \*/**

**void reshape(int width, int height)**

**{**

**GLint size;**

**if (width < height)**

**size = width;**

**else**

**size = height;**

**/\* glViewport(botom, left, width, height)**

**Define parte da janela a ser utilizada pelo OpenGL \*/**

**glViewport(0, 0, (GLint)size, (GLint)size);**

**/\* Matriz Projeção**

**Matriz onde se define como o mundo e apresentado na janela \*/**

**glMatrixMode(GL\_PROJECTION);**

**glLoadIdentity();**

**/\* gluOrtho(left,right,bottom,top,near,far);**

**Projeção ortogonal 3D, com profundidade (Z) entre -1 e 1 \*/**

**glOrtho(-1.0, 1.0, -1.0, 1.0, 1.0, -1.0);**

**/\* Matriz Modelview**

**Matriz onde são realizadas as tranformações dos modelos desenhados \*/**

**glMatrixMode(GL\_MODELVIEW);**

**glLoadIdentity();**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\* ESPAÇO PARA DEFINIÇÃO DAS ROTINAS \*\***

**\*\*\*\*\*\* AUXILIARES DE DESENHO ... \*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* Callback de desenho \*/**

**void draw(void){**

**glClear(GL\_COLOR\_BUFFER\_BIT);**

**/\* Espaço para chamada das rotinas auxiliares de desenho ... \*/**

**poligono(modelo.numLados,0,0,modelo.raio); //ADICIONAMOS A ESTA FUNC**

**mostrador(); //ADICIONAMOS A ESTA FUNC**

**ponteiros(); //ADICIONAMOS A ESTA FUNC**

**/\***

**glBegin(GL\_POLYGON);**

**glColor3f(1.0f, 0.0f, 0.0f);**

**glVertex2f(modelo.tamLado / 2, modelo.tamLado / 2);**

**glColor3f(0.0f, 0.0f, 1.0f);**

**glVertex2f(-modelo.tamLado / 2, modelo.tamLado / 2);**

**glColor3f(0.0f, 1.0f, 0.0f);**

**glVertex2f(-modelo.tamLado / 2, -modelo.tamLado / 2);**

**glColor3f(1.0f, 0.0f, 1.0f);**

**glVertex2f(modelo.tamLado / 2, -modelo.tamLado / 2);**

**glEnd();**

**\*/**

**glFlush();**

**if (estado.doubleBuffer)**

**glutSwapBuffers();**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\* CALLBACKS TIME/IDLE \*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* Callback Idle \*/**

**void idle(void)**

**{**

**/\* Ações a tomar quando o GLUT está idle \*/**

**/\* Redesenhar o ecrã \*/**

**glutPostRedisplay();**

**}**

**/\* Callback de temporizador \*/**

**void timer(int value){ //REFIZEMOS ESTA FUNC**

**glutTimerFunc(estado.delay, timer, 0);**

**/\* Acções do temporizador ...**

**Não colocar aqui primitivas OpenGL de desenho glBegin, glEnd, etc.**

**Simplesmente alterar os valores de modelo.hora.hor, modelo.hora.min e modelo.hora.seg \*/**

**/\* Ler hora do Sistema \*/**

**modelo.hora.seg++;**

**if(modelo.hora.seg == 60){**

**modelo.hora.min+=1;**

**modelo.hora.seg=0;**

**if(modelo.hora.min == 60){**

**modelo.hora.hor+=1;**

**modelo.hora.min = 0;**

**if(modelo.hora.hor == 12){**

**modelo.hora.hor = 0;**

**}**

**}**

**}**

**/\* Redesenhar o ecrã (invoca o callback de desenho) \*/**

**glutPostRedisplay(); }**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\* FUNÇÃO AJUDA \*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**void imprime\_ajuda(void)**

**{**

**printf("\n\nDesenho de um quadrado\n");**

**printf("h,H - Ajuda \n");**

**printf("+ - Aumentar tamanho do Lado\n");**

**printf("- - Diminuir tamanho do Lado\n");**

**printf("ESC - Sair\n");**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\* CALLBACKS TECLADO \*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* Callback para interação via teclado (carregar na tecla) \*/**

**void key(unsigned char key, int x, int y)**

**{**

**switch (key)**

**{**

**case 27: // Tecla Escape**

**exit(1);**

**/\* Ações sobre outras teclas \*/**

**case 'h':**

**case 'H':**

**imprime\_ajuda();**

**break;**

**case '+':**

**if (modelo.tamLado < 1.8)**

**{**

**modelo.tamLado += 0.05;**

**glutPostRedisplay();**

**}**

**break;**

**case '-':**

**if (modelo.tamLado > 0.2)**

**{**

**modelo.tamLado -= 0.05;**

**glutPostRedisplay();**

**}**

**break;**

**}**

**if (DEBUG)**

**printf("Carregou na tecla %c\n", key);**

**}**

**/\* Callback para interação via teclado (largar a tecla) \*/**

**void keyUp(unsigned char key, int x, int y)**

**{**

**if (DEBUG)**

**printf("Largou a tecla %c\n", key);**

**}**

**/\* Callback para interacção via teclas especiais (carregar na tecla) \*/**

**void specialKey(int key, int x, int y)**

**{**

**/\* Ações sobre outras teclas especiais**

**GLUT\_KEY\_F1 ... GLUT\_KEY\_F12**

**GLUT\_KEY\_UP**

**GLUT\_KEY\_DOWN**

**GLUT\_KEY\_LEFT**

**GLUT\_KEY\_RIGHT**

**GLUT\_KEY\_PAGE\_UP**

**GLUT\_KEY\_PAGE\_DOWN**

**GLUT\_KEY\_HOME**

**GLUT\_KEY\_END**

**GLUT\_KEY\_INSERT \*/**

**switch (key)**

**{**

**case GLUT\_KEY\_UP:**

**estado.delay \*= 2;**

**break;**

**case GLUT\_KEY\_DOWN:**

**estado.delay /= 2;**

**break;**

**/\* Redesenhar o ecrã \*/**

**glutPostRedisplay();**

**}**

**if (DEBUG)**

**printf("Carregou na tecla especial %d\n", key);**

**}**

**/\* Callback para interação via teclas especiais (largar a tecla) \*/**

**void specialKeyUp(int key, int x, int y)**

**{**

**if (DEBUG)**

**printf("Largou a tecla especial %d\n", key);**

**printf("Delay %d\n", estado.delay);**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\* FUNÇÃO MAIN \*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**int main(int argc, char \*\*argv)**

**{**

**estado.doubleBuffer = GL\_FALSE; // Colocar GL\_TRUE para ligar o Double Buffer**

**glutInit(&argc, argv);**

**glutInitWindowPosition(0, 0);**

**glutInitWindowSize(300, 300);**

**glutInitDisplayMode(((estado.doubleBuffer) ? GLUT\_DOUBLE : GLUT\_SINGLE) | GLUT\_RGB);**

**if (glutCreateWindow("Exemplo") == GL\_FALSE)**

**exit(1);**

**init();**

**imprime\_ajuda();**

**/\* Registar callbacks do GLUT \*/**

**/\* callbacks de janelas/desenho \*/**

**glutReshapeFunc(reshape);**

**glutDisplayFunc(draw);**

**/\* Callbacks de teclado \*/**

**glutKeyboardFunc(key);**

**glutKeyboardUpFunc(keyUp);**

**glutSpecialFunc(specialKey);**

**glutSpecialUpFunc(specialKeyUp);**

**/\* Callbacks timer/idle \*/**

**glutTimerFunc(estado.delay, timer, 0);**

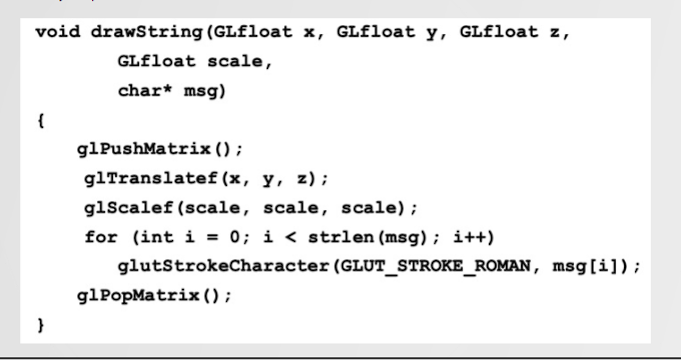
**glutIdleFunc(idle);**

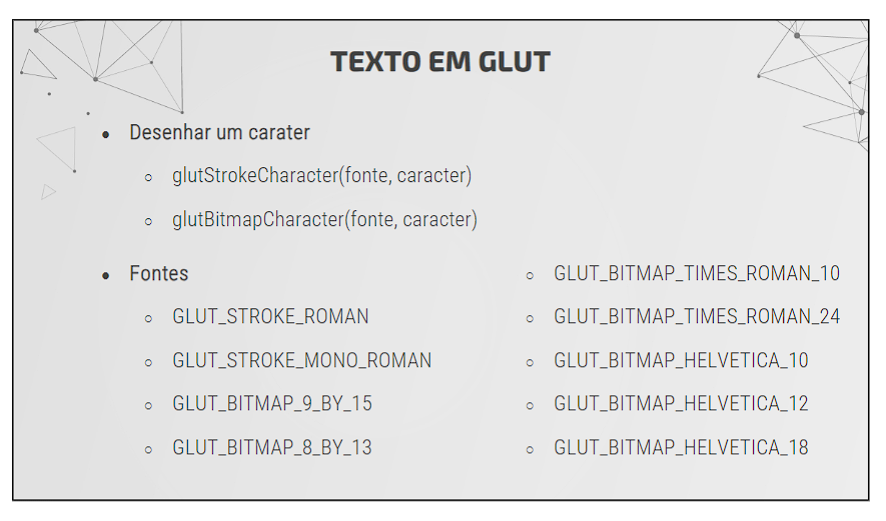
**/\* Começar loop \*/**

**glutMainLoop();**

**return 0;**

**}**

****

****

**Outras Funções Importantes**

**int glutCreateWindow (char\* nome)**

**//Cria uma nova janela primária (top-level)**

**//Nome é tipicamente usado para rotular a janela**

**//O número inteiro devolvido é usado pelo GLUT para identificar a janela**

**glutPostRedisplay()**

**//Indica ao GLUT que deve “chamar” a callback responsável pelo desenho (glutDisplayFunc) assim que possível**

**glScalef(2.0, 2.0, 1.0);**

**// controla a escala dos vertices de X,Y,Z (2 -> \*2 // 0.5 -> /2)**

**// Aumenta o tamanho do quadrado**

**glTranslatef(2.0, 2.0, 0.0);**

**// controla a posição do poligono nos eixos X,Y,Z**

**// Move o quadrado para a posição (2, 2)**

**Exercicio que tem 1 quadrado, 1 losango , 1 triangulo e 1 trapezio**

**void displayMe(void)**

**{**

**glClear(GL\_COLOR\_BUFFER\_BIT);**

**glColor3f(1.0,1.0,0.0);**

**//calcular pontos da circunferencia**

**float aDim = 360/4;**

**float angulo = 0;**

**float raio = 0.2;**

**// este é o losango**

**// /\**

**// \/**

**glBegin(GL\_POLYGON);**

**for (int i = 0; i < 4; i++) {**

**float radianos = (angulo\*3.14f/180);**

**float x = 0.0 + raio \* cos(radianos);**

**float y = 0.0 + raio \* sin(radianos);**

**glVertex3f(x, y, 0.0);**

**angulo -= aDim;**

**}**

**glEnd();**

**// este é o quadrado normal**

**glColor3f(1.0,0.0,0.0);**

**glBegin(GL\_POLYGON);**

**angulo = 45;**

**for (int i = 0; i < 4; i++) {**

**float radianos = (angulo\*3.14f/180);**

**float x = -0.7 + raio \* cos(radianos);**

**float y = 0.5 + raio \* sin(radianos);**

**glVertex3f(x, y, 0.0);**

**angulo -= aDim;**

**}**

**glEnd();**

**// este é o triangulo (uma linha de cada cor)**

**angulo = 180;**

**glBegin(GL\_LINE\_LOOP);**

**for (int i = 0; i < 3; i++) {**

**if(i == 0) glColor3f(0.0,1.0,0.0);**

**if(i == 1) glColor3f(0.0,1.0,1.0);**

**if(i == 2) glColor3f(1.0,0.7,1.0);**

**float radianos = (angulo\*3.14f/180);**

**float x = 0.0 + raio \* cos(radianos);**

**float y = 0.5 + raio \* sin(radianos);**

**glVertex3f(x, y, 0.0);**

**angulo -= aDim; }**

**glEnd();**

**// este é o trapezio**

**glBegin(GL\_LINE\_LOOP);**

**float C = -0.5;**

**glVertex3f(C,C,0.0);**

**glVertex3f(0.0,C,0.0);**

**glVertex3f(0.25,C-0.25,0.0);**

**glVertex3f(-0.75,C-0.25,0.0);**

**glEnd();**

**glFlush();**

**}**