Nome: Mirella Ribeiro Queiroz
Disciplina: Introdução à Computação Gráfica
Professor: Oscar

# Trabalho 1

Para compilar o código de cada arquivo é necessário executar o seguinte comando:

*gcc nome_arquivo.c -o nome_arquivo.exe -lglut -lGL -lGLU -lm*

Após compilar o código, é necessário rodar o seguinte comando para executar o arquivo compilado:

*./nome_arquivo.exe*

O “nome_arquivo” deve ser substituído pelo nome real do arquivo desejado.

**Arquivo base**

O arquivo chamado circulo.c é a base dos exercícios. Ao executar é possível ver um circulo de raio 1 a uma distância de 5 unidades da origem. Descrevendo o código temos:

Cada include é uma biblioteca que contém as funções necessárias para o programa. A stdlib é uma biblioteca do C que contém a função exit(), por exemplo. A math disponibiliza os cálculos necessários para o desenhos da circunferência. A glut contém as funções auxiliares do OpenGL.

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

Definição de um valor para a constante PI.

```
#define PI 3.141592
```

Definição dos valores das variáveis do raio e das coordenadas do círculo

```
GLfloat raio = 1;
GLfloat xcentro = 5;
GLfloat ycentro = 0;
```

Essa função desenha os eixo x, eixo y e o círculo pedido.

```
void Desenha(void){
```

Essa função especifica as intensidade de vermelho, verde e azul utilizadas para limpar as janelas. Pode variar de 0 a 1. O ultimo argumento serve pra compor superfícies transparentes.

```
    GlClearColor(1,1,1,0);
```

Essa função serve para limpar os buffers com valores pré definidos. O parâmetro diz a função que apenas o buffer de desenho deve ser limpo.

```
    glClear(GL_COLOR_BUFFER_BIT);
```

As funções glBegin e glEnd delimitam os vértices de desenho. O parâmetro GL_POINTS indica que os vértices devem ser tratados como pontos.

```
    glBegin(GL_POINTS);
```

Especifica a cor (R,G,B) que será o desenho.

```
    glColor3f(0,0,0);
    //eixos x e y
```

Desenha o eixo x.

```
    for(float x=-20;x<20;x+=0.01){
```

Essa função desenha as coordenadas.

```
            glVertex3f(x, 0, 0);
        }
```
Desenha o eixo y.
```
        for(float y=-20;y<20;y+=0.01){
            glVertex3f(0, y, 0);
        }
        //circulo
        glColor3f(0,0,1);
```
Desenha o circulo.
```
        for(float t=0;t<2*PI;t+=0.01){
            glVertex3f((raio*cos(t)+xcentro),(raio*sin(t)+ycentro),0);
        }

        glEnd();
```
Essa função faz com que qualquer comando, que ainda não tenha sido executado, seja executado o mais rápido possível.
```
        glFlush();
}
```

A função Keyboard serve para tratar eventos de teclado. Nesse caso, especifica que quando a tecla ESC (referente a key 27) for pressionada o programa deve ser finalizado.
```
void Keyboard(unsigned char key, int x, int y){
        if(key == 27)
        exit(0);
}

void Inicializa(void){
```
Essa função especifica a pilha de matrizes em que as operações matriciais ocorrerão. Nesse caso, será a pilha de projeção.
```
        glMatrixMode(GL_PROJECTION);
```
Essa função define o recorte das coordenadas.
```
        gluOrtho2D(-20,20,-20,20);
```
As pilhas de matrizes agora serão as usadas para definir translação, rotação e escalamento.
```
        glMatrixMode(GL_MODELVIEW);
}

int main(int argc, char *argv[]){
```
Inicializa a biblioteca GLUT.
```
        glutInit(&argc, argv);
```
Informa à biblioteca GLUT o modo do display quando a janela for criada. O GLU_SINGLE usa o buffer simples e o GLUT_RGB determina que o modelo de cor que será utilizado será o RGB.
```
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
```
Define o tamanho inicial da janela.
```
        glutInitWindowSize(500,500);
```
Cria uma janela e define o título.
```
        glutCreateWindow("Trabalho P1");
```
Define Desenha() como a função de desenho para a janela.
```
        glutDisplayFunc(Desenha);
```

Essa função diz que sempre que uma tecla for pressionada, a função Keyboard() vai ser chamada.

```
glutKeyboardFunc(Keyboard);
```

Chama a função Inicializa().

```
Inicializa();
```

Inicia o loop de processamento de desenhos com GLUT.

```
glutMainLoop();

return 0;
}
```

**1. Translação do centro da circunferência até a origem.**

O código a seguir, foi retirado do arquivo circulo_translado.c, faz o circulo transladar até a origem do eixo. Descrevendo o código que foi adicionado a base temos:

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

#define PI 3.141592
GLfloat raio = 1;
GLfloat xcentro = 5;
GLfloat ycentro = 0;

void Desenha(void){
	glClearColor(1,1,1,0);
	glClear(GL_COLOR_BUFFER_BIT);
	glBegin(GL_POINTS);
	glColor3f(0,0,0);
	//eixos x e y
	for(float x=-20;x<20;x+=0.01){
		glVertex3f(x, 0, 0);
	}
	for(float y=-20;y<20;y+=0.01){
		glVertex3f(0, y, 0);
	}
	//circulo
	glColor3f(0,0,1);
	for(float t=0;t<2*PI;t+=0.01){
		glVertex3f((raio*cos(t)+xcentro),(raio*sin(t)+ycentro),0);
	}
	glEnd();
	glFlush();
}

void Keyboard(unsigned char key, int x, int y){
	if(key == 27)
	exit(0);
```

```
}

void Inicializa(void){
	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(-20,20,-20,20);
	glMatrixMode(GL_MODELVIEW);
}

void Translacao(int valor){
```
	Condicionais que fazem a translação do circulo.
```
	if(xcentro>0){
		xcentro-=0.01;
	}
	if(xcentro<0){
		xcentro+=0.01;
	}
	if(ycentro>0){
		ycentro-=0.01;
	}
	if(ycentro<0){
		ycentro+=0.01;
	}
	glColor3f(1,0,0);
```
	Quando a função é executada, a função display é chamada novamente, fazendo com que a janela corrente seja redesenhada.
```
	GlutPostRedisplay();
```
	Essa função inicia um temporizador e após 10ms chama a função Translacao();
```
	glutTimerFunc(10,Translacao,valor);
}

int main(int argc, char *argv[]){
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(500,500);
	glutCreateWindow("Trabalho P1");
	glutDisplayFunc(Desenha);
	glutTimerFunc(10,Translacao,0.0);
	glutKeyboardFunc(Keyboard);
	Inicializa();
	glutMainLoop();

	return 0;
}
```

**2. A mesma translação do item anterior só que desta vez o círculo aumenta de tamanho até que o raio do círculo final tenha raio 2.**

O código a seguir, foi retirado do arquivo translado_raio.c, faz o circulo transladar até a origem do eixo aumento o raio para 2. Descrevendo o código que foi adicionado a base temos:

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

#define PI 3.141592
GLfloat raio = 1;
GLfloat xcentro = 5;
GLfloat ycentro = 0;

void Desenha(void){
	glClearColor(1,1,1,0);
	glClear(GL_COLOR_BUFFER_BIT);
	glBegin(GL_POINTS);
	glColor3f(0,0,0);
	//eixos x e y
	for(float x=-20;x<20;x+=0.01){
		glVertex3f(x, 0, 0);
	}
	for(float y=-20;y<20;y+=0.01){
		glVertex3f(0, y, 0);
	}
	//circulo
	glColor3f(0,0,1);
	for(float t=0;t<2*PI;t+=0.01){
		glVertex3f((raio*cos(t)+xcentro),(raio*sin(t)+ycentro),0);
	}
	glEnd();
	glFlush();
}

void Keyboard(unsigned char key, int x, int y){
	if(key == 27)
	exit(0);
}

void Inicializa(void){
	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(-20,20,-20,20);
	glMatrixMode(GL_MODELVIEW);
}

void Translacao(int valor){
	if(xcentro>0){
		xcentro-=0.01;
	}
```

```
	if(xcentro<0){
		xcentro+=0.01;
	}
	if(ycentro>0){
		ycentro-=0.01;
	}
	if(ycentro<0){
		ycentro+=0.01;
	}
	//aumenta o raio
```
Código que aumenta o raio do circulo enquanto ele translada pra origem do eixo.
```
	if(raio<2){
		raio+=0.002;
	}
	glutPostRedisplay();
	glutTimerFunc(10,Translacao,valor);
}

int main(int argc, char *argv[]){
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(500,500);
	glutCreateWindow("Trabalho P1");
	glutDisplayFunc(Desenha);
	glutTimerFunc(10,Translacao,0.0);
	glutKeyboardFunc(Keyboard);
	Inicializa();
	glutMainLoop();

	return 0;
}
```

**3. Rotação do círculo em torno da origem.**

O código a seguir foi retirado do arquivo rotacao_origem.c e faz o circulo rotacionar em torno da origem. Descrevendo o código que foi adicionado a base temos:

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

#define PI 3.141592
GLfloat raio = 1;
GLfloat xcentro = 5;
GLfloat ycentro = 0;

void Desenha(void){
	glClearColor(1,1,1,0);
	glClear(GL_COLOR_BUFFER_BIT);
	glBegin(GL_POINTS);
	glColor3f(0,0,0);
```

```
	//eixos x e y
	for(float x=-20;x<20;x+=0.01){
		glVertex3f(x, 0, 0);
	}
	for(float y=-20;y<20;y+=0.01){
		glVertex3f(0, y, 0);
	}
	//circulo
	glColor3f(0,0,1);
	for(float t=0;t<2*PI;t+=0.01){
		glVertex3f((raio*cos(t)+xcentro),(raio*sin(t)+ycentro),0);
	}
	glEnd();
	glFlush();
}

void Keyboard(unsigned char key, int x, int y){
	if(key == 27)
	exit(0);
}

void Inicializa(void){
	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(-20,20,-20,20);
	glMatrixMode(GL_MODELVIEW);
}

void Rotacao(int valor){
```
Faz com que o circulo rotacione ao redor da origem.
```
	float r = valor*0.01;
	xcentro = 5*sin(r);
	ycentro = 5*cos(r);
	valor++;
	glutPostRedisplay();
	glutTimerFunc(10,Rotacao,valor);
}

int main(int argc, char *argv[]){
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(500,500);
	glutCreateWindow("Trabalho P1");
	glutDisplayFunc(Desenha);
	glutTimerFunc(10,Rotacao,0.0);
	glutKeyboardFunc(Keyboard);
	Inicializa();
	glutMainLoop();

	return 0;
}
```

**4. Rotação combinada, círculo rotando sobre seu centro e este rotando em torno da origem.**

O código a seguir foi retirado do arquivo rotacao_centro.c e faz o circulo rotacionar a origem e sobre o próprio centro. Descrevendo o código que foi adicionado a base temos:

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

#define PI 3.141592
GLfloat raio = 1;
GLfloat xcentro = 5;
GLfloat ycentro = 0;
GLfloat coeficiente = 0.1;

void Desenha(void){
	glClearColor(1,1,1,0);
	glClear(GL_COLOR_BUFFER_BIT);
	glBegin(GL_POINTS);
	glColor3f(0,0,0);
	//eixos x e y
	for(float x=-20;x<20;x+=0.01){
		glVertex3f(x, 0, 0);
	}
	for(float y=-20;y<20;y+=0.01){
		glVertex3f(0, y, 0);
	}
	//circulo
	glColor3f(0,0,1);
	for(float t=0+coeficiente;t<2*PI+coeficiente;t+=0.01){
		glVertex3f((raio*cos(t)+xcentro),(raio*sin(t)+ycentro),0);
		if (t>=PI+coeficiente){
			glColor3f(1,0,0);
		}
	}
	glEnd();
	glFlush();
}

void Keyboard(unsigned char key, int x, int y){
	if(key == 27)
	exit(0);
}

void Inicializa(void){
	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(-20,20,-20,20);
	glMatrixMode(GL_MODELVIEW);
}
```

```
void Rotacao(int valor){
	float r = valor*0.01;
	xcentro = 5*sin(r);
	ycentro = 5*cos(r);
	valor++;
```
faz o circulo rotacionar sobre o próprio centro.
```
	coeficiente+=0.1;
	glutPostRedisplay();
	glutTimerFunc(10,Rotacao,valor);
}

int main(int argc, char *argv[]){
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(500,500);
	glutCreateWindow("Trabalho P1");
	glutDisplayFunc(Desenha);
	glutTimerFunc(10,Rotacao,0.0);
	glutKeyboardFunc(Keyboard);
	Inicializa();
	glutMainLoop();

	return 0;
}
```