

TRABALHO DA 3ª UNIDADE – COMPUTAÇÃO GRÁFICA.

Relatório do trabalho de implementação de com OpenGL.

Marcos André Azevedo de Assis

Departamento de Computação – Universidade do Estado do Rio Grande do Norte (UERN)
Av. Dr. João Medeiros Filho, nº 3419 – Natal – RN – Brasil.

`marcosassis@alu.uern.br`

Resumo

Este relatório tem o objetivo de apresentar o processo de modelagem de um Robô Articulado virtual com a utilização da Application Programming Interface (API) Open Graphics Library (OpenGL). A proposta dessa API é simplificar a síntese de imagens virtuais. Sendo assim, neste trabalho, com o auxílio da API OpenGL, foi gerado um braço mecânico também conhecido como Robô Articulado, e este, será controlado por meio do teclado do computador, utilizando teclas específicas. Para isso, inicialmente, foi realizado um estudo na documentação da API, com a finalidade de compreender as funções que foram aplicadas no desenvolvimento do trabalho. Como resultado obtido, foi possível desenvolver formas geométricas que simulam um braço mecânico.

1. INTRODUÇÃO

OpenGL é o principal ambiente para o desenvolvimento de aplicativos gráficos 2D e 3D portáteis e interativos. Desde a sua introdução em 1992, o OpenGL se tornou a interface de programação de aplicativos gráficos 2D e 3D mais amplamente usada e suportada do setor, trazendo milhares de aplicativos para uma ampla variedade de plataformas de computador. O OpenGL promove a inovação e acelera o desenvolvimento de aplicativos ao incorporar um amplo conjunto de renderização, mapeamento de textura, efeitos especiais e outras funções de visualização poderosas. Os desenvolvedores podem aproveitar o poder do OpenGL em todas as plataformas populares de desktops e estações de trabalho, garantindo ampla implantação de aplicativos.

Como descrito, OpenGL não é uma linguagem de computação como C, Java ou Python, é uma API ou biblioteca para desenvolvimento de aplicações gráficas 3D renderizadas em tempo real.

2. MODELAGEM

Para o desenvolvimento do trabalho, foi necessário instalar a biblioteca OpenGL Utility Toolkit (GLUT).

A biblioteca GLUT disponibiliza funcionalidades para OpenGL e o principal objetivo é a abstração do sistema operacional fazendo com que os aplicativos sejam multiplataforma.

A biblioteca possui funcionalidades para criação e controle de janelas, e também tratamento de eventos de dispositivos de entrada (mouse e teclado). Também existem rotinas para o desenho de formas tridimensionais pré-definidas como cubo, esfera, bule, etc.

3. INSTALAÇÃO E CONFIGURAÇÃO

Para o desenvolvimento do projeto foi usado o Sistema Operacional Linux Ubuntu 20.04, e a instalação das bibliotecas foram feitas da seguinte forma:

Instalação

1. `sudo apt update && sudo apt upgrade`

- Comando para atualização da lista das versões dos pacotes disponíveis e atualiza os pacotes instalados no Sistema Operacional.

2. `sudo apt-get install libgl1-mesa-dev`

- Instala os arquivos .h das bibliotecas do OpenGL.

3. `sudo apt-get install build-essential`

- Instala um compilador GCC C/C++ associado com a ferramenta *make*.

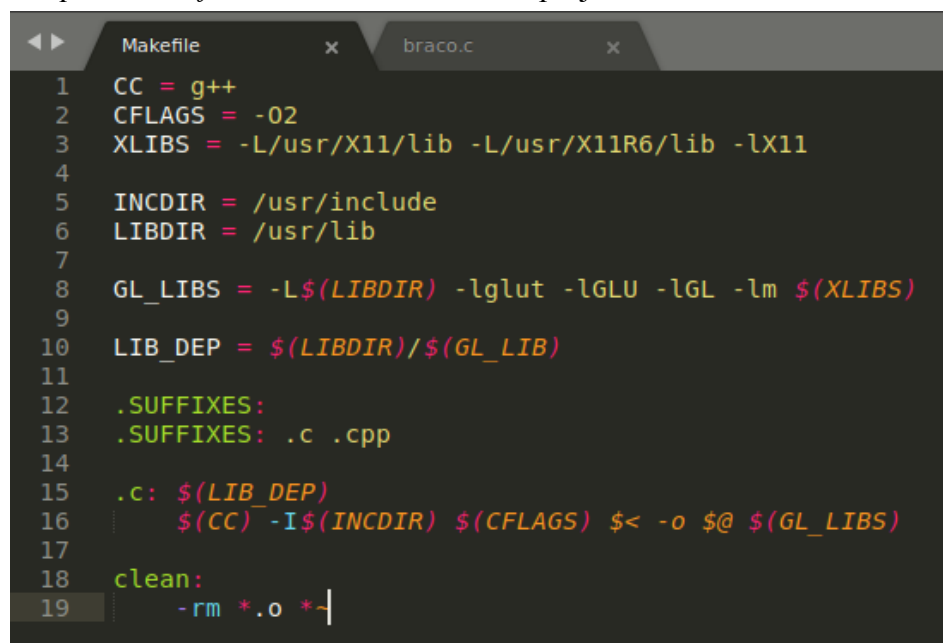
4. `sudo apt-get install libglew1.5-dev freeglut3-dev libglm-dev`

- Instalação de bibliotecas GLEW, FreeGLUT e GLM.

As tarefas de compilação foram automatizadas com o utilitário *make*. O *make* determina automaticamente que partes de um grande programa necessitam ser compiladas e os comandos necessários para compilar, a partir da leitura das regras definidas em um arquivo *Makefile*. Assim, para efetuar a compilação do programa, basta executar o comando `$ make nome_do_arquivo.c`.

Configuração

1. Criar o arquivo *Makefile* dentro do diretório do projeto.



```
1 CC = g++
2 CFLAGS = -O2
3 XLIBS = -L/usr/X11/lib -L/usr/X11R6/lib -lX11
4
5 INCDIR = /usr/include
6 LIBDIR = /usr/lib
7
8 GL_LIBS = -L$(LIBDIR) -lglut -lGLU -lGL -lm $(XLIBS)
9
10 LIB_DEP = $(LIBDIR)/$(GL_LIB)
11
12 .SUFFIXES:
13 .SUFFIXES: .c .cpp
14
15 .c: $(LIB_DEP)
16     $(CC) -I$(INCDIR) $(CFLAGS) $< -o $@ $(GL_LIBS)
17
18 clean:
19     -rm *.o *~
```

2. Agora para compilar o código basta ir até o diretório do projeto e digitar o comando:
 - `$ make nome_do_arquivo`
3. Se tudo estiver nos conformes, agora é só executar o projeto usando o comando:
 - `$ make ./nome_do_arquivo`

4. CÓDIGO

Inicialmente foi importado a biblioteca GLUT, logo em seguida declaradas e inicializadas as variáveis de movimentação do Robô Articulado. Depois foi instanciado um objeto chamado *base* no modelo quadrático que irá ser usado para criar formas geométricas.

```
Makefile x braco.c x
1 #include <GL/glut.h>
2
3 float movimentoOmbro = 0, movimentoCotovelo = 0, movimentoMao = 0, dedoDireito = -1, dedoEsquerdo = 1;
4 GLUquadricObj *base;
5
```

Dando continuidade, temos a função **montarRobo()**. Aqui foram implementadas todas as formas geométricas, suas posições dentro da cena, suas respectivas cores, formatos e tudo que for relacionado a características visuais do Robô. Foram utilizadas com grande frequência as definições da biblioteca GLUT e elas podem ser identificadas facilmente dentro do código, pois são iniciadas com a sigla *gl*.

```
6 void montarRobo(void){
7     glClear(GL_COLOR_BUFFER_BIT); // Superfície plana
8     glColor3f(0, 1, 1);
9     glBegin(GL_QUADS);
10         glVertex2f(-12,-2);
11         glVertex2f(-12, 0);
12         glVertex2f( 12, 0);
13         glVertex2f( 12,-2);
14     glEnd();
15
16     glPushMatrix();
17     glRotatef(movimentoOmbro, 0, 0, 1);
18     glColor3f(1, 1, 0);
19
20     gluDisk(base, 0, 2, 16, 1); // Ombro
21
22     glBegin(GL_QUADS); // Elo 1 - braço
23         glVertex2f(0,-0.5);
24         glVertex2f(0, 0.5);
25         glVertex2f(6, 0.5);
26         glVertex2f(6,-0.5);
27     glEnd();
28
29     glPushMatrix();
30     glTranslatef(6, 0, 0);
31     glRotatef(movimentoCotovelo, 0, 0, 1);
32     glColor3f(1, 0, 0);
33     gluDisk(base, 0, 0.8, 16, 1); // Cotovelo
34     glBegin(GL_QUADS); // Elo 2 - antebraço
35         glVertex2f(0,-0.3);
36         glVertex2f(0, 0.3);
37         glVertex2f(3, 0.3);
38         glVertex2f(3,-0.3);
39     glEnd();
40 }
```

```

41     glPushMatrix();
42     glTranslatef(3, 0, 0);
43     glRotatef(movimentoMao, 0, 0, 1);
44     glColor3f(0, 0, 1);
45     gluDisk(base, 0, 0.8, 10, 1); // Munheca
46     glBegin(GL_QUADS); // Elo 3 - Mão
47         glVertex2f(0, -0.3);
48         glVertex2f(0, 0.3);
49         glVertex2f(2, 0.3);
50         glVertex2f(2, -0.3);
51     glEnd();
52
53     glPushMatrix();
54     glTranslatef(1, 0, 0);
55     glColor3f(0, 0, 1);
56     glBegin(GL_QUADS); // Palma Mão
57         glVertex2f(1, -1);
58         glVertex2f(1, 1);
59         glVertex2f(1.5, 1);
60         glVertex2f(1.5, -1);
61     glEnd();
62
63     glBegin(GL_QUADS); // Dedo Esquerdo
64         glVertex2f(1.5, dedoEsquerdo - 0.5);
65         glVertex2f(1.5, dedoEsquerdo);
66         glVertex2f(3, dedoEsquerdo);
67         glVertex2f(3, dedoEsquerdo - 0.5);
68     glEnd();
69
70     glBegin(GL_QUADS); // Dedo Direito
71         glVertex2f(1.5, dedoDireito);
72         glVertex2f(1.5, dedoDireito + 0.5);
73         glVertex2f(3, dedoDireito + 0.5);
74         glVertex2f(3, dedoDireito);
75     glEnd();
76     glPopMatrix();
77     glPopMatrix();
78     glPopMatrix();
79     glFlush();
80
81 }

```

Foi usado **glDisk()** para gerar os círculos que simulavam as juntas do braço robótico, **glBegin()** e **glVertex2f()** para montar quadrados que serviram para simular o braço, antebraço, dedos e palma da mão.

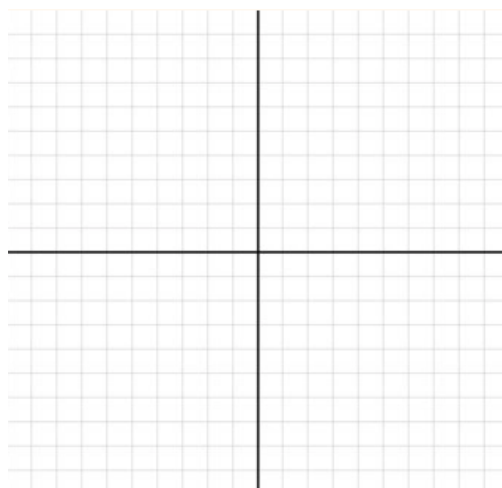
Em seguida foi criada a função de **controlarBraco()** que recebe como parâmetro uma tecla que foi pressionada no teclado e atualiza a variável de controle de movimentos do objeto a qual aquela tecla foi atribuída a operação, e por fim atualiza a matriz de projeção com as variáveis com os novos valores usando a função **glutPostRedisplay()**.

```

83 void controlarBraco(unsigned char botaoPressionado, int x, int y){
84     switch(botaoPressionado){
85         case 'n':
86             if(dedoDireito >= -1.1 && dedoDireito <= -0.5)
87                 dedoDireito += 0.1;
88             if(dedoEsquerdo >= 0.5 && dedoEsquerdo <= 1.1)
89                 dedoEsquerdo -= 0.1;
90             break;
91
92         case 'm':
93             if(dedoDireito >= -0.9 && dedoDireito <= -0.4)
94                 dedoDireito -= 0.1;
95             if(dedoEsquerdo >= 0.4 && dedoEsquerdo <= 0.9)
96                 dedoEsquerdo += 0.1;
97             break;
98
99         case 'z': movimentoOmbro += 5;
100        break;
101
102        case 'x': movimentoOmbro -= 5;
103        break;
104
105        case 'a': movimentoCotovelo += 7;
106        break;
107
108        case 's': movimentoCotovelo -= 7;
109        break;
110
111        case 'q': movimentoMao += 9;
112        break;
113
114        case 'w': movimentoMao -= 9;
115        break;
116    }
117    glutPostRedisplay(); // Atualiza a janela
118 }

```

Em seguida temos as funções **main()** e **abrirJanela()**, onde essa última função é chamada para inicialização da janela de projeções usando **glClearColor()** para definir a cor de background da janela e **gluOrtho2D()** para definir a matriz de projeções dentro da janela. Matriz essa que pode ser entendida com a imagem abaixo.

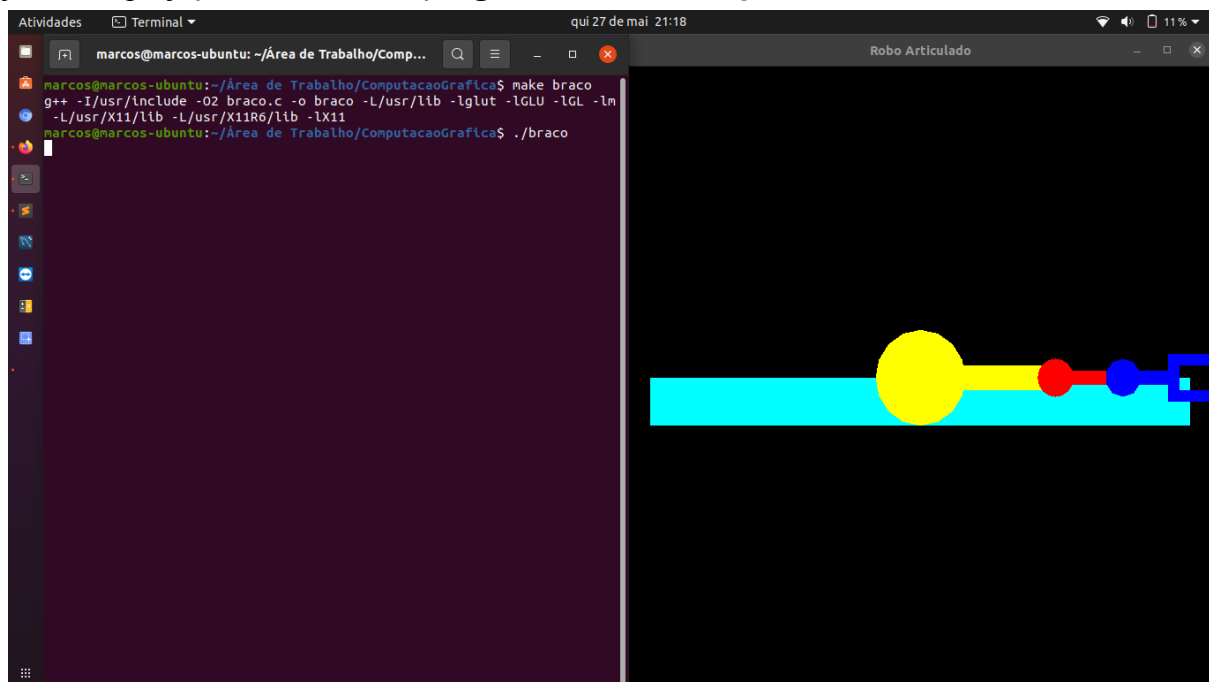


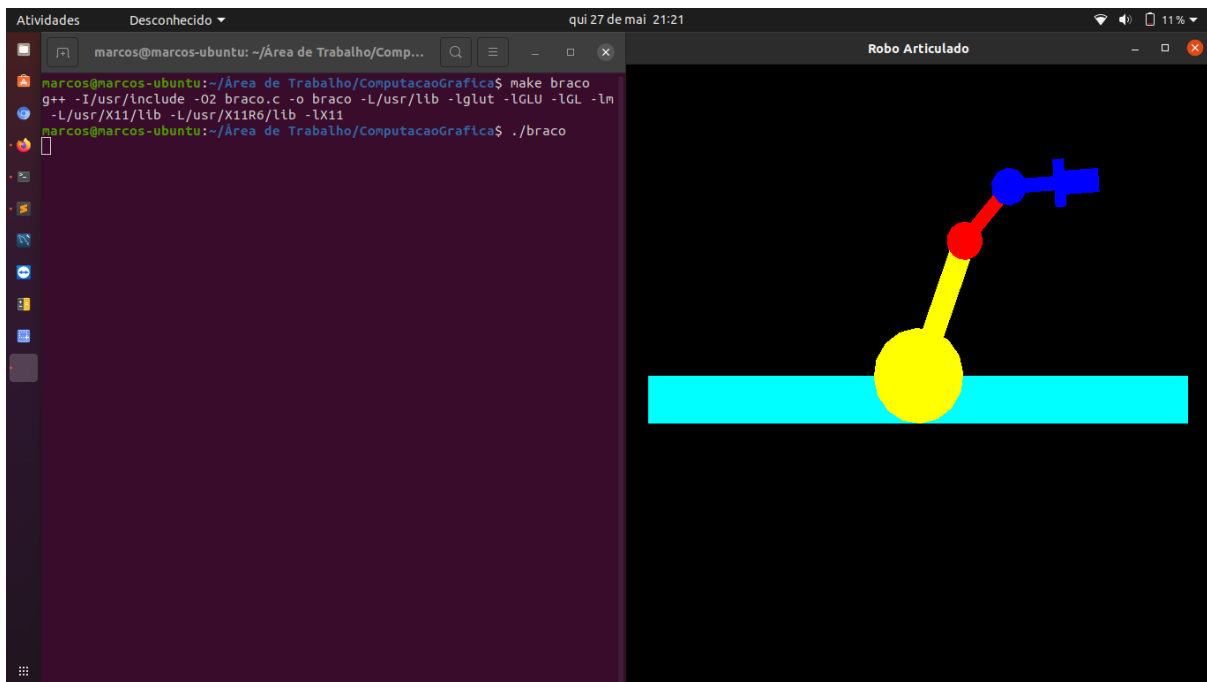
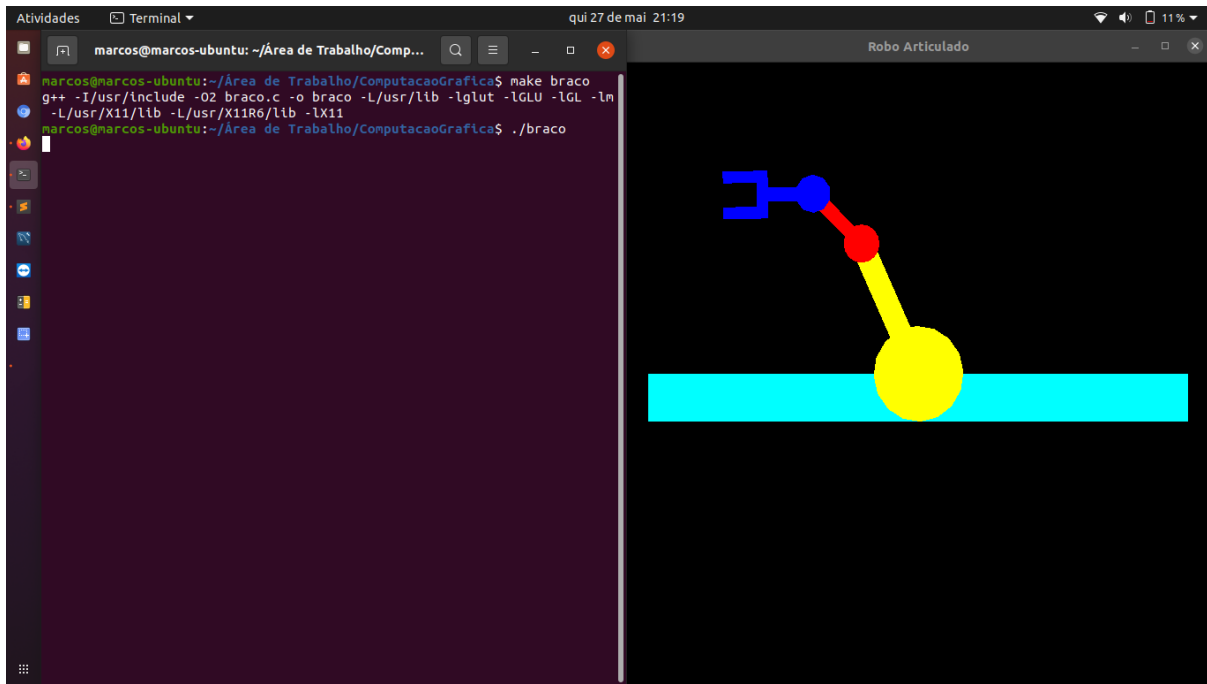
```

120 void abrirJanela(void){
121     glClearColor(0, 0, 0, 1); // RGBA
122     gluOrtho2D(-13, 13, -13, 13); // Matriz 26 x 26
123 }
124
125 int main(int argc, char** argv){
126     glutInit(&argc, argv);
127
128     glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
129
130     glutCreateWindow("Robo Articulado");
131
132     base = gluNewQuadric();
133
134     glutDisplayFunc(montarRobo);
135
136     glutKeyboardFunc(controlarBraco);
137
138     abrirJanela();
139
140     glutMainLoop();
141 }

```

Na função **main()** é chamado a função de **abrirJanela()** e gera um loop com a função **glutMainLoop()** para que o programa fique sempre executando e na escuta de novos comandos. A função **glutKeyboardFunc()** recebe as teclas digitadas para serem passadas para a função de **controlarBraco()**. A função **glutDisplayFunc()** usa a função **montarRobo()** para desenhar todas as partes do robô. Também é adicionado um título à janela de projeções usando a função **glutCreateWindow()**.





REFERÊNCIAS

<https://www.opengl.org/>

<https://agostinhobritojr.github.io/tutorial/opengl/>

https://pt.wikibooks.org/wiki/Programa%C3%A7%C3%A3o_com_OpenGL/Instala%C3%A7%C3%A3o/Linux