# Relatório Sobre Realismo em um Objeto Tridimensional

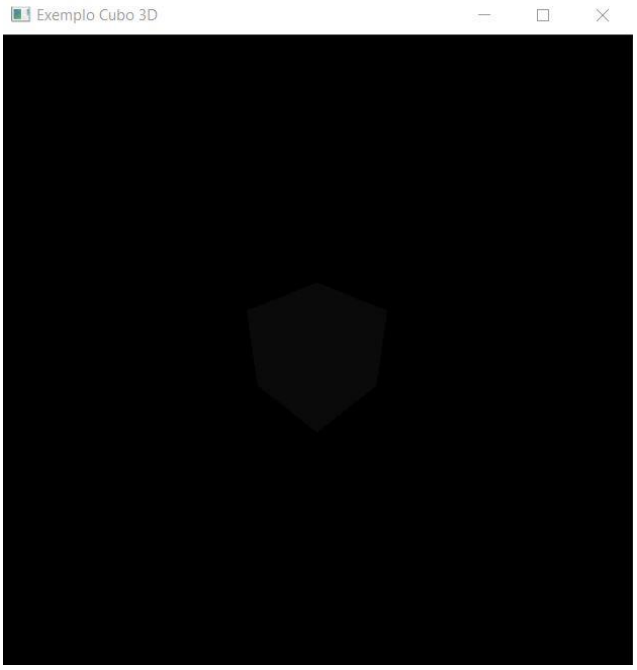
Bruno de Souza Cruz

**Resumo**

Implementar um programa para aplicar as transformações em um objeto tridimensional, são elas a cor, iluminação e textura. O OpenGL é uma API livre utilizada na computação gráfica, para desenvolvimento de aplicativos gráficos ambientes 3D, ela foi utilizada no programa, o GLUT é uma biblioteca de funcionalidades para OpenGL cujo principal objetivo é a abstração do sistema operacional fazendo com que os aplicativos sejam multiplataforma, a ide utilizada foi o Code::Blocks.

## 1.1. Aplicando a iluminação

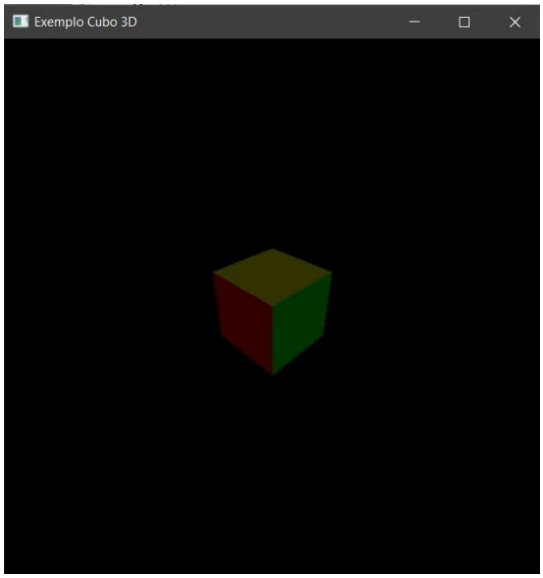
Continuando com o trabalho da avaliação II, a aplicação de cor já foi feito na avaliação II, agora será feito a iluminação do objeto, dentro do método inicializar foi chamado o comando GL_LIGHTING, o problema é que quando esse comando é habilitado ele não reconhece as cores e o resultado inicial se ver na (Figura 1.0).



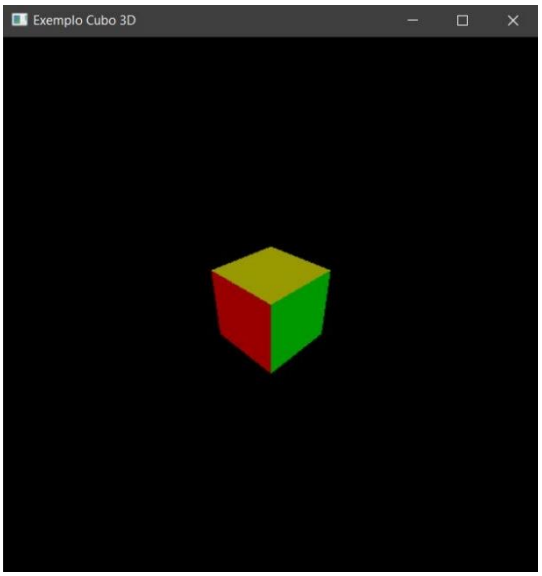

**(Figura 1.0)**

Pra resolver isso tem que passar a informação que a cor é material, habilitando o GL_COLOR_MATERIAL, perceba que na (Figura 1.1) agora dá pra ver as cores do cubo só que com a quantidade de luz baixa. Havia uma luz na cena, essa é a luz ambiente, ela é pequena, que por padrão ela é 0.2, pra modificar a luz do ambiente inicial basta criar uma variável e passar novos valores do ambiente que inicialmente eram (0.2, 0.2, 02), o último valor é o alpha, por fim chama o lgLightModelfv e passa como parâmetro a luz do ambiente, e a modificação feita, com a modificação o resultado da iluminação se ver na (Figura 1.2).

```
1   #include <GL/freeglut.h>
2
3   void inicializar(){
4       glClearColor(0.0, 0.0, 0.0, 0.0);
5       glEnable(GL_DEPTH_TEST);
6       glMatrixMode(GL_MODELVIEW);
7       glLoadIdentity();
8
9       glEnable(GL_LIGHTING);
10      glEnable(GL_COLOR_MATERIAL);
11      float global_ambiente[] = {0.6, 0.6, 0.6, 1.0};
12      glLightModelfv(GL_LIGHT_MODEL_AMBIENT, global_ambiente);
13
14      gluLookAt(2.0,2.0,2.0,
15                0.0,0.0,0.0,
16                0.0,1.0,0.0);
17  }
18
```




**(Figura 1.1)**




**(Figura 1.2)**

## 1.2. Aplicando a textura

A textura é o aspecto de uma superfície ou seja, quando olhamos para um objeto ou uma superfície e vimos se ela é lisa ou rugosa por exemplo, a textura é por isso, uma sensação visual ou tocável. Após muitas pesquisas e tentativas consegui implementar um código que colocava a textura nas faces do cubo.

Primeiramente algumas alterações no programa foram feitas para que se possa trabalhar com a textura, o método transformar ele continua o mesmo, o método inicializar agora ele especifica a função usada para comparar cada valor de profundidade de pixel, no método desenhar_cubo é habilitado o recurso gl, foi criado uma classes com extensão ".h" para que possa carregar o arquivo bitmap.

main.cpp | BmpLoader.h | src\BmpLoader.cpp

```
#ifndef BMPLOADER_H
#define BMPLOADER_H

#include <windows.h>

class BmpLoader
{
    public:
        unsigned char* textureData;
        int iWidth, iHeight;
        BmpLoader(const char*);
        ~BmpLoader();
    private:
        BITMAPFILEHEADER bfh;
        BITMAPINFOHEADER bih;
};

#endif // BMPLOADER_H

```

O glTextCoord2f define as coordenadas de textura atuais, essa função especifica as coordenadas de textura em uma, duas, três ou quatro dimensões.

```
glBegin(GL_QUADS);
    glTexCoord2f(1.0,1.0);  glVertex3f(1.0,1.0,0.0);
    glTexCoord2f(0.0,1.0);  glVertex3f(-1.0,1.0,0.0);
    glTexCoord2f(0.0,0.0);  glVertex3f(-1.0,-1.0,0.0);
    glTexCoord2f(1.0,0.0);  glVertex3f(1.0,-1.0,0.0);
glEnd();
```

Foi criado um método chamado carregar_textura, dentro dele basicamente acontece, a definição dos parâmetros de textura, acontece também a vinculação da textura nomeada a um alvo texturizado, o carregamento do arquivo .h, a imagem utilizada no programa é no formato (.bmp).

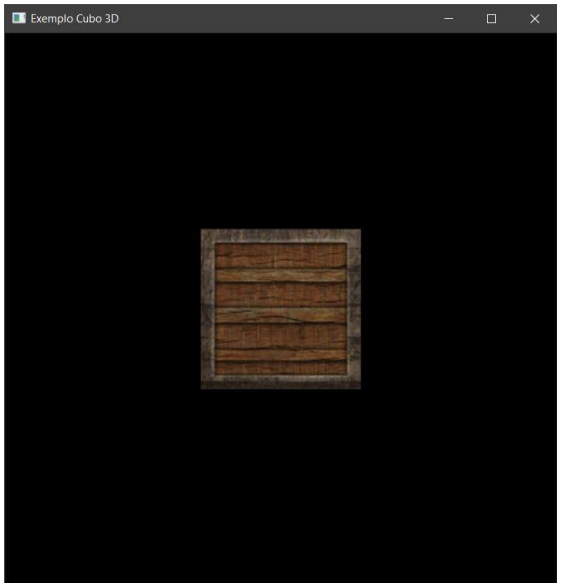
```
void carregar_textura(const char*filename)
{
    BmpLoader bl(filename);
    glGenTextures(1,&ID);
    glBindTexture(GL_TEXTURE_2D, ID);
    glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    gluBuild2DMipmaps(GL_TEXTURE_2D, GL_RGB, bl.iWidth, bl.iHeight, GL_RGB, GL_UNSIGNED_BYTE, bl.textureData);
}
```

No método principal acontece a chamada da imagem a ser colocada no objeto.

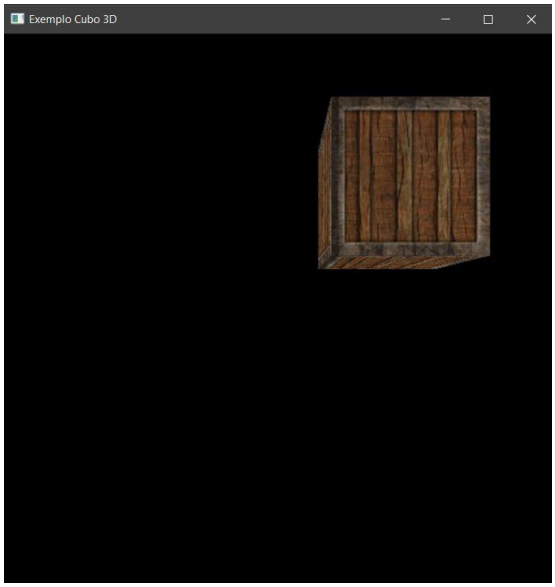
```
int main(int argc, char**argv)
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);
    glutInitWindowSize(600,600);
    glutCreateWindow("Exemplo Cubo 3D");
    carregar_textura("imagem-1.bmp");
    glutDisplayFunc(desenhar_cubo);
    glutReshapeFunc(renderizar);
    inicializar();
    glutMainLoop();
    return 0;
}

```

## 1.3 Resultados

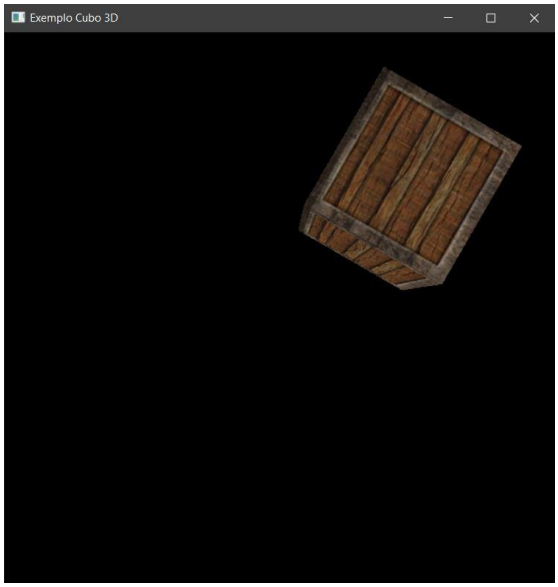
Veja na (Figura 1.3) o primeiro resultado da aplicação de textura, veja na (Figura 1.4) e (Figura 1.5) o resultado da textura com a translação e rotação e na (Figura 1.6) uma outra textura.



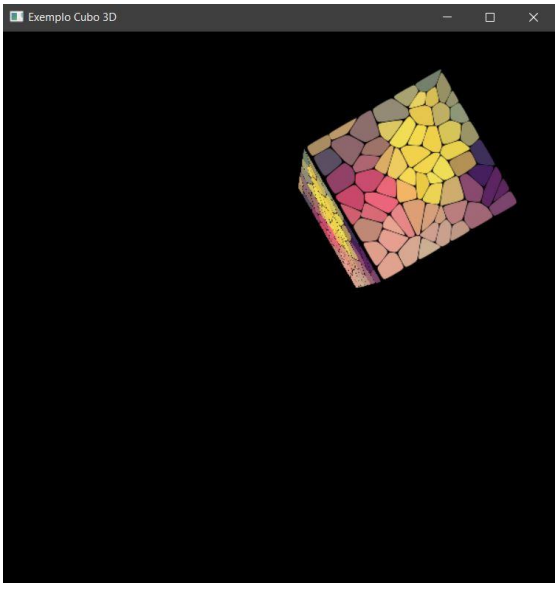

**(Figura 1.3)**




**(Figura 1.4)**




**(Figura 1.5)**




**(Figura 1.6)**

Link para ver o código no GitHub:

brunnu-sc/trabalho-objetos-tridimensionais: Computação Gráfica (github.com)

### 1.5 Referências

Computação Gráfica 05 – Prof. Jorge Cavalcanti (UNIVASF)

Computação Gráfica 07 – Prof. Jorge Cavalcanti (UNIVASF)

http://www.univasf.edu.br/~jorge.cavalcanti/configcb.html

Introdução à Computação Gráfica OpenGL Básico (João Paulo & Claudio Esperança)