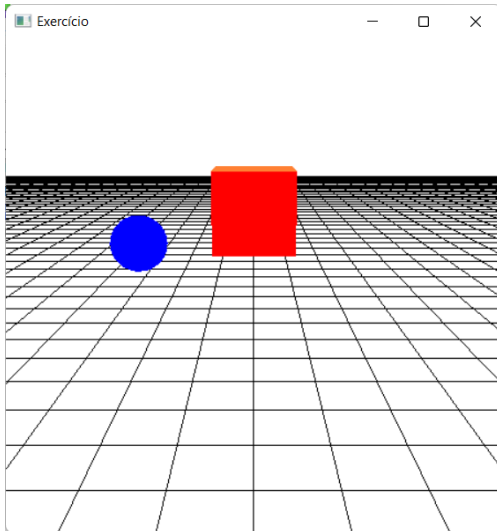


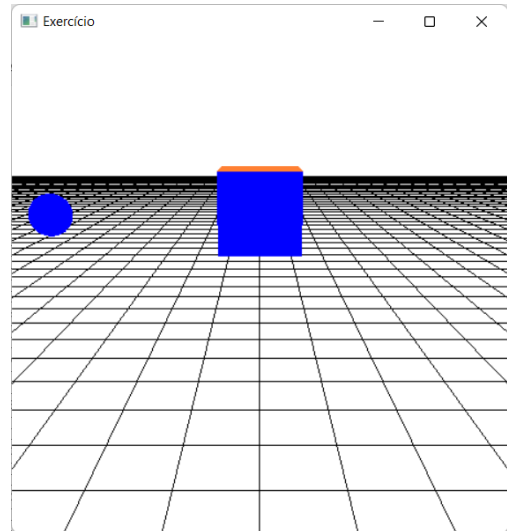
## OPENGL – PRÁTICA 08

### Exercício

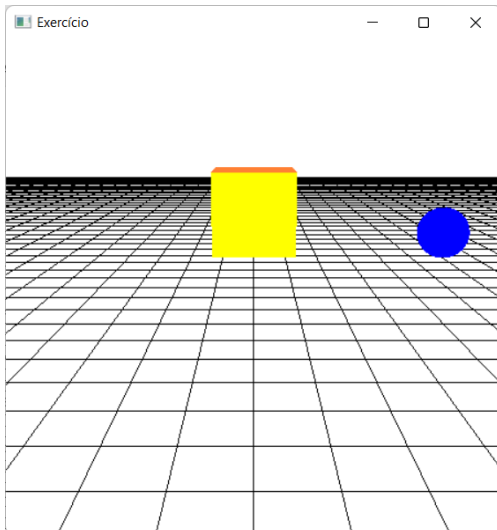
Implemente um programa em OpenGL com um cenário composto por um cubo, uma esfera e uma superfície (Figura 01).



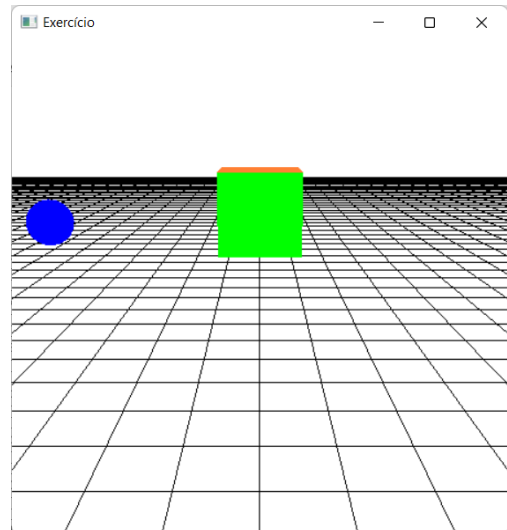
a) Vista anterior



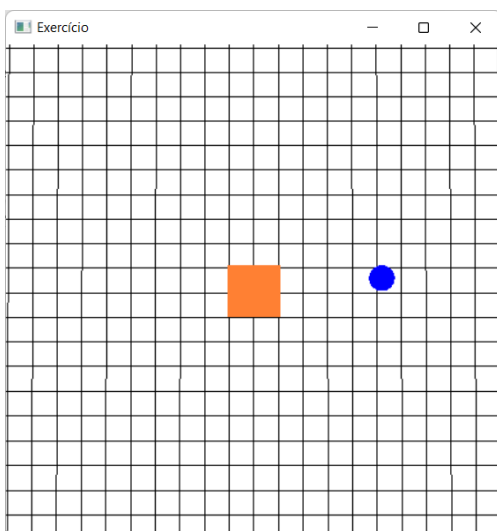
b) Vista lateral esquerda



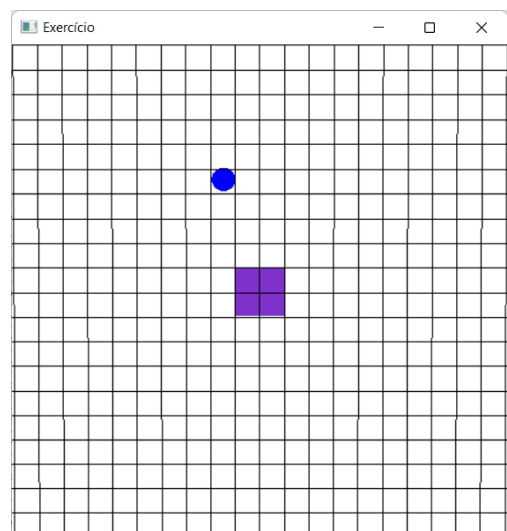
c) Vista posterior



d) Vista lateral direita









e) Vista superior



f) Vista inferior

**Figura 01** - Vistas do cenário

O programa deve atender aos seguintes requisitos:

Dificuldade	Requisito
	<b>Cenário</b> <ul style="list-style-type: none"><li>O cenário composto por uma superfície, o cubo e esfera é semelhante ao exercício da <b>Prática 07</b>.</li></ul>
	<b>Operação de zoom</b> <ul style="list-style-type: none"><li>A operação de zoom deve acontecer por meio dos botões do mouse. Esse recurso já foi implementado no exercício da <b>Prática 07</b>.</li></ul>
	<b>Operação de pan</b> <ul style="list-style-type: none"><li>A operação de pan deve acontecer por meio do teclado. Esse recurso já foi implementado no exercício da <b>Prática 07</b>.</li></ul>
	<b>Animação</b> <ul style="list-style-type: none"><li>A posição do cubo é fixa. A esfera deve realizar o movimento de translação em todo do cubo. Essa animação é semelhante ao exercício da <b>Prática 07</b>.</li></ul>
	<b>Controle da visão da câmera</b> <ul style="list-style-type: none"><li>O controle da visão da câmera deve acontecer por meio do teclado. Esse recurso deve permitir a visualização do cenário em relação às vistas anterior, lateral esquerda, posterior, lateral direita, superior e inferior.</li></ul>
	<b>Câmera orbital</b> <ul style="list-style-type: none"><li>O controle da câmera orbital deve acontecer por meio do teclado. Esse recurso deve mover a câmera em todo de um ponto de interesse e permitir a visualização de todos os elementos do cenário.</li></ul>

#### Dicas

- A função **glutSolidSphere** cria uma esfera
  - <https://www.opengl.org/resources/libraries/glut/spec3/node81.html>
- A superfície é um conjunto de linhas horizontais e verticais criadas em relação aos eixos **X** e **Z**
  - [https://www.youtube.com/watch?v=mW\\_LO1wMS3c](https://www.youtube.com/watch?v=mW_LO1wMS3c)
- A função **desenhaCubo** cria um cubo colorido
  - Anexo I

#### Observações

- O arquivo executável (**exercicio.exe**) do exercício foi disponibilizado para auxiliar a visualização e compreensão do funcionamento do exercício.
- Recursos do teclado que controlam a visão e movimento orbital da câmera:
  - Tecla **1**: exibe a vista anterior do cubo
  - Tecla **2**: exibe a vista lateral esquerda do cubo
  - Tecla **3**: exibe a vista posterior do cubo
  - Tecla **4**: exibe a vista lateral direita do cubo
  - Tecla **5**: exibe a vista superior do cubo
  - Tecla **6**: exibe a vista inferior do cubo
  - Tecla **0**: ativa e desativa a movimentação orbital da câmera

#### Referências

Todas as informações descritas neste roteiro foram retiradas do livro "OpenGL: Uma abordagem prática e objetiva", presente no plano didático da disciplina.

```

void desenhaCubo(GLfloat tamanho){
    //anterior - vermelho
    glColor3f(1.0f, 0.0f, 0.0f);
    glBegin(GL_QUADS);
        glVertex3f(-tamanho, tamanho, tamanho);
        glVertex3f(-tamanho, -tamanho, tamanho);
        glVertex3f( tamanho, -tamanho, tamanho);
        glVertex3f( tamanho, tamanho, tamanho);
    glEnd();

    //lateral esquerda - azul
    glColor3f(0.0f, 0.0f, 1.0f);
    glBegin(GL_QUADS);
        glVertex3f( tamanho, tamanho, tamanho);
        glVertex3f( tamanho, -tamanho, tamanho);
        glVertex3f( tamanho, -tamanho, -tamanho);
        glVertex3f( tamanho, tamanho, -tamanho);
    glEnd();

    //posterior - amarelo
    glColor3f(1.0f, 1.0f, 0.0f);
    glBegin(GL_QUADS);
        glVertex3f( tamanho, tamanho, -tamanho);
        glVertex3f(-tamanho, tamanho, -tamanho);
        glVertex3f(-tamanho, -tamanho, -tamanho);
        glVertex3f( tamanho, -tamanho, -tamanho);
    glEnd();

    //lateral direita - verde
    glColor3f(0.0f, 1.0f, 0.0f);
    glBegin(GL_QUADS);
        glVertex3f(-tamanho, tamanho, tamanho);
        glVertex3f(-tamanho, tamanho, -tamanho);
        glVertex3f(-tamanho, -tamanho, -tamanho);
        glVertex3f(-tamanho, -tamanho, tamanho);
    glEnd();

    //superior - laranja
    glColor3f(1.0f, 0.5f, 0.2f);
    glBegin(GL_QUADS);
        glVertex3f(-tamanho, tamanho, tamanho);
        glVertex3f( tamanho, tamanho, tamanho);
        glVertex3f( tamanho, tamanho, -tamanho);
        glVertex3f(-tamanho, tamanho, -tamanho);
    glEnd();

    //inferior - roxo
    glColor3f(0.5f, 0.2f, 0.8f);
    glBegin(GL_QUADS);
        glVertex3f(-tamanho, -tamanho, tamanho);
        glVertex3f(-tamanho, -tamanho, -tamanho);
        glVertex3f( tamanho, -tamanho, -tamanho);
        glVertex3f( tamanho, -tamanho, tamanho);
    glEnd();
}

```