UNIVERSIDADE FEDERAL DE OURO PRETO

Faculdade de Ciência da Computação

Computação Gráfica

Movimentação de um quadrado com teclado e mouse utilizando o OpenGL e GLUT

Marcos Vinicio Euzébio
Gabriel Araújo Saldanha

Ouro Preto
Junho/2025

Introdução

Nesta atividade, propõe-se a construção de uma cena 3D com cubos hierárquicos utilizando OpenGL e GLUT, explorando os seguintes conceitos: transformação geométrica hierárquica, controle de câmera, transparência, espessura de linha e manipulação de cores no modelo HSV, convertido para RGB em tempo de execução.

O desenvolvimento promove o domínio prático dos conceitos fundamentais de computação gráfica tridimensional, bem como o uso do pipeline de renderização da OpenGL clássica.

Objetivo

Este relatório possui os seguintes objetivos:

1. Implementar uma cena com dois cubos 3D utilizando OpenGL e C++, com controle hierárquico de transformações entre eles.

2. Aplicar cores geradas com base no modelo HSV, convertidas em tempo real para RGB.
3. Utilizar efeitos visuais como transparência (alpha blending) e arestas em destaque.
4. Permitir controle de câmera e interatividade por teclado.
5. Registrar as etapas de desenvolvimento e decisões técnicas tomadas.

Partes relevantes do código fonte

As partes do código fonte são muito intuitivas e os nomes das funções já dizem por si só o que elas significam; ademais, o código fonte em si não é nada complicado. Portanto, os comentários inseridos no programa que o grupo codificou já são mais do que suficientes para entender o fluxo de execução do código.

display()

A função display() é responsável por renderizar a cena a cada frame. Nela, a câmera é posicionada com gluLookAt, são aplicadas as transformações nos cubos pai e filho utilizando glPushMatrix e glPopMatrix, e os cubos são desenhados com transparência e arestas visíveis.

```
void display() {
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glLoadIdentity();

   gluLookAt(zoom * sin(cameraAngle), 0, zoom * cos(cameraAngle), 0, 0, 0, 0, 1, 0);

   glRotatef(angleX, 1, 0, 0);
   glRotatef(angleY, 0, 1, 0);

   glEnable(GL_BLEND);
   glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

   // Pai
   glPushMatrix();
   glRotatef(parentRot, 0, 1, 0);
   drawColoredCube(0.6f);

   // Filho
   glPushMatrix();
   glTranslatef(2.5f, 0, 0);
   glScalef(0.5, 0.5, 0.5);
   glRotatef(childRot, 0, 1, 0);
   drawColoredCube(0.8f);
   glPopMatrix();

   glPopMatrix();

   glutSwapBuffers();
}
```

drawColoredCube()

Função que desenha o cubo 3D usando 12 triângulos (2 por face). Cada face recebe uma cor definida no modelo HSV, convertida dinamicamente para RGB por meio da função HSVtoRGB.

Também são desenhadas as arestas com GL_LINES para destaque visual, usando glLineWidth(2).

HSVtoRGB()

Função de conversão baseada em segmentação do círculo cromático HSV. A implementação identifica o setor (intervalo de 60°) em que a cor se encontra e interpola as componentes RGB para gerar a cor correspondente com brilho e saturação controláveis.

```
void HSVtoRGB(float h, float s, float v, float &r, float &g, float &b) {
    int i = int(h * 6);
    float f = h * 6 - i;
    float p = v * (1 - s);
    float q = v * (1 - f * s);
    float t = v * (1 - (1 - f) * s);
    switch(i % 6) {
        case 0: r = v, g = t, b = p; break;
        case 1: r = q, g = v, b = p; break;
        case 2: r = p, g = v, b = t; break;
        case 3: r = p, g = q, b = v; break;
        case 4: r = t, g = p, b = v; break;
        case 5: r = v, g = p, b = q; break;
    }
}
```

Transformações Hierárquicas

O cubo pai é transformado (translação, rotação), e o cubo filho é posicionado em relação ao pai. As chamadas glPushMatrix() e glPopMatrix() garantem o correto empilhamento de transformações:

```
glPushMatrix();          // transforma o cubo pai
 drawCube();
 glPushMatrix();        // aplica transformação do cubo filho sobre a base do pai
    drawCube();
 glPopMatrix();
glPopMatrix();
```

Interação com Teclado

Por meio da função keyboard(), o usuário pode:

- Rotacionar a câmera (a, d)
- Aproximar ou afastar a visão (w, s)

- Rotacionar o cubo pai (j)
- Rotacionar o cubo filho (k)

```
void keyboard(unsigned char key, int, int) {
   switch (key) {
       case 'a': cameraAngle -= 0.1f; break;
       case 'd': cameraAngle += 0.1f; break;
       case 'w': zoom += 0.5f; break;
       case 's': zoom -= 0.5f; break;
       case 'j': parentRot += 5; break;
       case 'k': childRot += 5; break;
       case 27: exit(0); // ESC
   }
   glutPostRedisplay();
}
```

reshape()

Define o viewport e a matriz de projeção com perspectiva (campo de visão de 45°, proporção da janela e profundidade entre 1 e 100 unidades). Essa função garante a visualização correta mesmo quando a janela é redimensionada.

```
void reshape(int w, int h) {
   if (h == 0) h = 1;
   float ratio = 1.0f * w / h;
   glViewport(0, 0, w, h);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluPerspective(45, ratio, 1, 100);
   glMatrixMode(GL_MODELVIEW);
}
```

Diretivas de compilação e execução

O próprio código fonte enviado pelo grupo mostra como compilar e executar o código, porém, para fins elucidativos, mostramos como realizar ambos os processos.

Todo o processo de instalação das dependências, compilação e execução são feitos no terminal.

Instalando as dependências (sistemas baseados em Linux)

Como é certo quase toda distribuição baseada em Linux - *para uso pessoal, como o Ubuntu* - vir com o compilador g++, nos preocupamos somente com as bibliotecas principais que o sistema usa, as quais, neste caso, são o OpenGL e o GLUT, que podem ser obtidos a partir do seguinte comando via terminal:

```
sudo apt-get install build-essential freeglut3-dev
```

Compilando o projeto

Uma vez dentro do diretório do projeto, para compilar o programa basta executar o seguinte comando:

```
g++ Atividade03/atividade.cpp -o Atividade03/atividade -lGL -lGLU -lglut
```

Executando o projeto

Ainda dentro do diretório do projeto, nós o executamos a partir do seguinte comando:

```
./Atividade03/atividade
```

Conclusão

Esta atividade proporcionou uma compreensão aprofundada sobre a aplicação prática de transformações geométricas hierárquicas, manipulação de cores e interatividade em ambientes 3D.

Ao explorar o modelo HSV e sua conversão para RGB, foi possível criar efeitos visuais controlados e variados. A hierarquia de cubos permitiu entender como a pilha de matrizes da OpenGL influência na transformação relativa de objetos.

O uso do teclado como ferramenta de interação consolidou ainda mais a ideia de controle em tempo real de câmeras e objetos, aspectos fundamentais em sistemas de renderização interativos.