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

O OpenGL é uma ferramenta poderosíssima para a criação de aplicações gráficas. A linguagem C++ possui suporte à ferramenta e ao *framework* GLUT, que serão utilizados na própria linguagem pelo grupo responsável por redigir este relatório para a criação de um programa cuja função é renderizar um quadrado 2D na tela e permitir que o usuário interaja com ele por meio do teclado e do mouse.

Objetivo

Este relatório possui os seguintes objetivos:

1. Exibir um pequeno manual acerca do processo de compilação e execução do programa.

2. Passar por partes relevantes do código fonte que explicam a lógica do que é visto sendo executado graficamente.

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()* nada mais faz do que renderizar a tela e o quadrado que está contido nela.

```
void display() {
    // Os dois códigos abaixo resetam os buffers de dados que são preenchidos a cada execução
    glClear(GL_COLOR_BUFFER_BIT);
```

```
    glLoadIdentity();

    // Atribui a cor branca ao quadrado
    glColor3f(1.0f, 1.0f, 1.0f);

    // Desenha o quadrado definindo a coordenada de cada um dos 4 vértices
    glBegin(GL_QUADS);
        glVertex2f(squareX - squareSize / 2.0f, squareY - squareSize / 2.0f);
        glVertex2f(squareX + squareSize / 2.0f, squareY - squareSize / 2.0f);
        glVertex2f(squareX + squareSize / 2.0f, squareY + squareSize / 2.0f);
        glVertex2f(squareX - squareSize / 2.0f, squareY + squareSize / 2.0f);
    glEnd();

    // Faz um "swap" dos buffers para exibir o quadrado
    glutSwapBuffers();
}
```

reshape()

A função *reshape()* determina como a janela OpenGL deve se comportar quando for redimensionada ou criada inicialmente.

```
void reshape(int width, int height) {
    // Define o viewport
    glViewport(0, 0, width, height);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    // Define uma janela logica entre -1 e 1
    gluOrtho2D(-1.0, 1.0, -1.0, 1.0);
    glMatrixMode(GL_MODELVIEW);
}
```

specialKeys()

A função da *specialKeys()* captura qual tecla está sendo pressionada pelo o usuário, age na alteração da coordenada do quadrado, dependendo da tecla, e renderiza novamente para atualizar as informações.

```
void specialKeys(int key, int x, int y) {
    const float moveStep = 0.05f;

    // Detecta a tecla e acrescenta/decrementa o valor de X ou Y na coordenada.
    switch (key) {
        case GLUT_KEY_UP:    squareY += moveStep; break;
        case GLUT_KEY_DOWN:  squareY -= moveStep; break;
        case GLUT_KEY_LEFT:  squareX -= moveStep; break;
        case GLUT_KEY_RIGHT: squareX += moveStep; break;
    }

    // Apenas verifica se a atualização do valor de alguma coordenada provoca overflow do
```

```
quadrado na tela. Se sim, readapta a coordenada para ficar no limite
    if (squareX + squareSize / 2.0f > 1.0f) squareX = 1.0f - squareSize / 2.0f;
    if (squareX - squareSize / 2.0f < -1.0f) squareX = -1.0f + squareSize / 2.0f;
    if (squareY + squareSize / 2.0f > 1.0f) squareY = 1.0f - squareSize / 2.0f;
    if (squareY - squareSize / 2.0f < -1.0f) squareY = -1.0f + squareSize / 2.0f;

    // Redesenha a tela
    glutPostRedisplay();
}
```

mouseClick()

Similar ao *pressedKey()*, a função *mouseClick()* verifica se o botão esquerdo do *mouse* foi pressionado sobre a tecla e imprime a coordenada onde se clicou no terminal.

```
void mouseClick(int button, int state, int x, int y) {
    // Verifica se o botao esquerdo do mouse foi pressionado
    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
        // O if irá basicamente converter o pixel em coordenada OpenGL e printar o ponto
clicado na tela
        GLdouble objX, objY, objZ;
        GLdouble modelview[16], projection[16];
        GLint viewport[4];

        glGetDoublev(GL_MODELVIEW_MATRIX, modelview);
        glGetDoublev(GL_PROJECTION_MATRIX, projection);
        glGetIntegerv(GL_VIEWPORT, viewport);

        gluUnProject(x, viewport[3] - y, 0.0, modelview, projection, viewport, &objX, &objY,
&objZ);

        std::cout << "Clique em: X = " << objX << ", Y = " << objY << std::endl;
    }
}
```

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++ Atividade02/atividade.cpp -o Atividade02/atividade -lGL -lGLU -lglut
```

Executando o projeto

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

```
./Atividade02/atividade
```

Conclusão

Concluímos que, com a execução deste trabalho, foi possível obter um entendimento melhor das ferramentas gráficas OpenGL e GLUT, o que fortificou e edificou o conhecimento dos alunos responsáveis pela criação do código no que tange à computação gráfica.