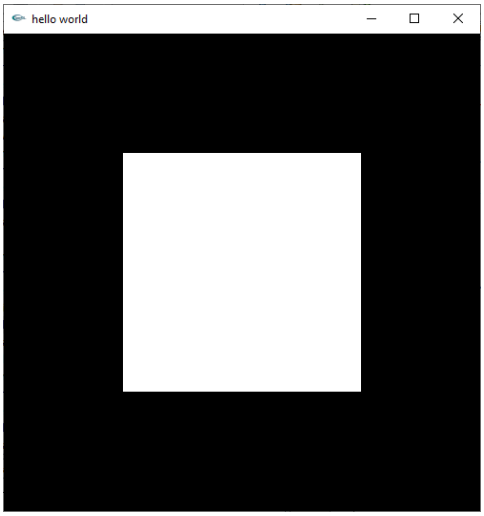
Roteiro de Laboratório - OpenGL

Objetivo: Trabalhar os conceitos básicos do OpenGL e do GLUT incluindo a construção de uma aplicação gráfica que trate interações básicas com o usuário.

Material: Sistema operacional Linux (recomendação Ubuntu 20.04) com g++, OpenGL e GLUT instalados (geralmente só são necessários dois comandos para a instalação, https://gist.github.com/AbdullahKady/f2782157991df652c2baee0bba05b788).

1. Tarefa:

Criar uma aplicação “Hello World!” de computação gráfica. A aplicação cria uma janela com um quadrado branco na tela, como na figura abaixo.



1.1. Passo: criação da janela base

- Compilar o arquivo “quadrado.cpp” fornecido com este roteiro com o comando “g++ -o test quadrado.cpp -lGL -lGLU -lglut”. As bibliotecas OpenGL, GLU e GLUT são referenciadas respectivamente com os comandos -lGL -lGLU -lglut.
- Executar o arquivo compilado com o comando “./test”
- Verificar se uma janela foi criada como descrito para esta tarefa

1.2. Passo: modificação do objeto

- Alterar a posição dos vértices do quadrado no código e ver o que acontece
- Alterar a cor do objeto no código e ver o que acontece
- Alterar o tipo do objeto no código e ver o que acontece

1.3. Passo: modificação de comandos gerais

- Comentar o comando “glClear (GL_COLOR_BUFFER_BIT);”, verificar o que acontece e depois retorne com o comando.
    - O comando é responsável por inicializar o buffer de cores.
    - Se ele não é inicializado, o valor é lixo
    - Em computação gráfica, lixo geralmente é o que estava lá antes na tela

- Teste outros tamanhos e posições de janela no código
- Teste o redimensionamento da janela

2. Tarefa:

Introduzir interatividade via teclado na aplicação anterior.

2.1. Passo: Interatividade via evento direto do teclado

- Incluir duas variáveis globais no programa anterior para controlar a posições x e y do quadrado (float gX = 0 e float gY = 0)
- Adicionar essas variáveis às respectivas coordenadas x e y dos vértices do quadrado da função *display* para permitir incrementar a posição dele quando a variáveis mudarem de valor (pode testar se funcionou colocando diferentes valores de inicialização no item anterior)
- Implementar a call-back de teclado com cabeçalho "void keyPress(unsigned char key, int x, int y)"
    - Utilizar as teclas *w*, *s*, *a* e *d* para controlar respectivamente incrementos para cima, para baixo, para esquerda e para a direita.
    - Fazer incrementos de ±0.01 nas variáveis gX e gY (testar outros valores se estiver pequeno ou grande para sua máquina), de acordo com a tecla pressionada
    - O parâmetro *key* contém caractere da tecla pressionada (ex. *if (key == 'a')* para saber se *a* foi pressionado)
    - Finalizar a função com o comando "glutPostRedisplay();" para indicar que a tela deve ser renderizada novamente
- Registrar a rotina de call-back do teclado com o comando "glutKeyboardFunc(keyPress);"
- Compilar o programa, executar e verificar se consegue mover o quadrado pressionando as teclas descritas acima
- Tentar mover na diagonal
    - Não vai conseguir de forma fácil, pois um evento cancela o outro

2.2. Passo: Interatividade via evento do teclado mais idle

- Incluir uma variável global "int keyStatus[256];" que será responsável por armazenar o estado (pressionado = 1 ou não pressionado = 0) de cada uma das teclas.
- Utilizar eventos do teclado para alterar o estado das teclas na variável *keyStatus* e não mais para fazer incrementos
    - Alterar a função *keyPress* para ao invés de fazer incrementos em gX e gY, marcar uma tecla como pressionada em *keyStatus* (ex. "keyStatus[(int)('a')] = 1;" para marcar *a* como pressionada)
    - A função *keyPress* e disparada quando uma tecla do teclado é pressionada, portanto precisamos também tratar o evento de quando a tecla é solta com call-back de cabeçalho "void keyUp(unsigned char key, int x, int y)". Essa função deve marcar a tecla *key* como não pressionada (ex. "keyStatus[(int)(key)] = 0;") e depois pedir para redesenhar a tela com "glutPostRedisplay();".
    - Registrar *keyUp* com comando "glutKeyboardUpFunc(keyUp);"
- Agora, com as teclas marcadas, é necessário atualizar o estado do mundo virtual (o mundo virtual dessa aplicação é composto apenas por um quadrado) continuamente, ou seja, enquanto alguma tecla de interesse estiver pressionada a sua respectiva variável gX e/ou

gY deve(m) receber o incremento apropriado (utilizar um valor menor do que o anterior, ex. 0.001, e alterar se necessário).
    - Para isso, implementar a call-back de cabeçalho “void idle(void)” que incrementará continuamente o estado das variáveis gX e gY sempre que as teclas respectivas estiverem pressionadas (ex. “if(keyStatus[(int)('a')]) gX -= 0.001;”). Este evento atualizará continuamente o estado interno do mundo virtual.
    - Pedir para redesenhar a tela no final da função com o comando “glutPostRedisplay();”
- Compilar, rodar e verificar se o movimento está mais suave e permite diagonal
- Testar o que ocorre quando o evento de soltar a tecla é perdido
    - Modificar o valor do incremento para mover bem devagar
    - Pressionar uma tecla para mover
    - Enquanto estiver movendo, clique em outra janela diferente da sua e depois solte a tecla que estava pressionada
    - Retorne para a janela da aplicação e veja que o quadrado vai continuar movendo, pois quem capturou o evento de soltar a tecla foi a outra janela

3. Tarefa:

Introduzir interatividade via mouse na aplicação anterior.

3.1. Passo: Testar coordenadas retornadas pelo evento do mouse

- Implementar a função de call-back do evento de clique do mouse com o cabeçalho “void mouse(int button, int state, int x, int y)” e fazer um *printf* para o terminal das posições retornadas pelos parâmetros *x* e *y*.
- Registrar o call-back com “glutMouseFunc(mouse);”
- Compilar, rodar e clicar próximo aos cantos da janela para verificar que o (0, 0) está no canto superior esquerdo
- Inverter o *y* logo no início do evento para garantir que o (0, 0) esteja como estamos acostumados, isto é, no canto inferior esquerdo
    - Para inverter, basta subtrair y do tamanho da janela
- Compilar, rodar e clicar próximo aos cantos da janela para verificar se funcionou

3.2. Passo: Atualizar posição do quadrado para a posição clicada pelo mouse

- Fazer gX e gY receberem a posição do clique do mouse
- Compilar, rodar e clicar em um ponto na tela e verificar que o quadrado some.
    - Isso ocorre, pois o sistema de coordenadas está em pixel, porém a janela de desenho do quadrado foi definida de 0.0 a 1.0
    - Para corrigir, deve-se dividir as coordenadas pelo tamanho da janela, fazendo variar de 0.0 a 1.0 (cuidado com a divisão inteira)
- Compilar, rodar e clicar em um ponto na tela e verificar que o quadrado foi desenhado deslocado para a direita a para cima do seu clique.
    - Isso ocorreu, pois o quadrado é sempre desenhado com o canto inferior esquerdo na posição gX+0.25 e gY+0.25
    - Portanto, se for de interesse, isso deveria ser corrigido no desenho original do quadrado na display, ou no clique do mouse

3.3. Passo: Exercício de arrastar o quadrado com o mouse

- Alterar a aplicação para permitir fazer o *drag and drop* do quadrado, ou seja, se clicar dentro do quadrado e arrastar com o botão pressionado, o quadrado deve se mover com o mouse até o botão ser solto. Tente melhorar para que o arraste seja feito mantendo a posição clicada inicialmente.
- Teste seu programa como se fosse um usuário malvado!
- Pode desfazer o passo 3.2 para fazer este exercício