Universidade Presbiteriana Mackenzie
Eduardo Veit Ferrão RA: 10389961
Leonardo Pinheiro de Souza RA: 10388961
Lucas Paulo da Rocha RA: 10391076
Luiz Octavio Tassinari Saraiva RA: 10374379

Atividade em grupo de até 4 pessoas.

Analise o exemplo de código C que usa SDL, OpenGL e GLSL para renderizar um triângulo, disponível em: https://github.com/profkishimoto/HelloGL.

O que deve ser alterado no código original para que o programa exiba um cubo de cores que fica girando em um ou mais eixos sem parar?

No "cubo de cores", a posição (x, y, z) de um vértice é mapeada para uma cor (r, g, b). Assim, um vértice na posição xyz(0, 0, 0) tem a cor preta rgb(0, 0, 0), um vértice na posição xyz(1, 0, 0) tem a cor vermelha rgb(1, 0, 0), e assim por diante.

Como referência do que o programa alterado deve fazer, veja o vídeo disponível no repositório acima.

Descreva as alterações que devem ser realizadas e publique o resultado dessa atividade no blog.

Lembre-se de incluir todas as referências consultadas (livros, links de artigos, vídeos, etc.) e identificar todas as pessoas do grupo.

O que deve ser alterado no código original para que o programa exiba um cubo de cores que fica girando em um ou mais eixos sem parar?

Um cubo tem seis faces quadradas. Como o OpenGL só conhece triângulos, teremos que desenhar 12 triângulos: dois para cada face. Apenas definimos nossos vértices da mesma forma que fizemos para o triângulo. Uma cor é, conceitualmente, exatamente igual a uma posição: são apenas dados. Assim, declaramos as cores como um trio RGB por vértice:

```
static const GLfloat g_color_buffer_data[] = {
    0.583f,  0.771f,  0.014f,
    0.609f,  0.115f,  0.436f,
    etc…
}
```

Agora, o restante das configurações (criação do buffer, preenchimento, bound e etc) continuam iguais, apenas precisamos atualizar o shader de fragmentos para utilizar as cores dos vértices.

No "cubo de cores", a posição (x, y, z) de um vértice é mapeada para uma cor (r, g, b). Assim, um vértice na posição xyz(0, 0, 0) tem a cor preta rgb(0, 0, 0), um vértice na posição xyz(1, 0, 0) tem a cor vermelha rgb(1, 0, 0), e assim por diante.

Para exibir um cubo de cores que gira continuamente, devemos substituir os vértices do triângulo pelos vértices do cubo, onde cada vértice é mapeado para uma cor com base na sua posição `(x, y, z)`, por exemplo, `(0, 0, 0)` será mapeado para a cor preta `(0, 0, 0)` e `(1, 0, 0)` para a cor vermelha.
Faremos isso adicionando um buffer de cores correspondente a esses vértices e modificando o shader de fragmentos para usar as cores atribuídas. Além disso, incluímos uma matriz de modelo (`model matrix`) que será atualizada continuamente para aplicar a rotação ao cubo dentro do loop de renderização.
No exemplo que encontramos, o teste de profundidade foi habilitado (`glEnable(GL_DEPTH_TEST)`) para garantir a renderização correta dos fragmentos. No loop principal a matriz de modelo aplica uma rotação com `glm.rotate`, enviando para o shader e desenhando o cubo com `glDrawArrays(GL_TRIANGLES, 0, 36)`, garantindo assim que o cubo gire continuamente nos eixos desejados.

Fontes:
https://www.opengl-tutorial.org/beginners-tutorials/tutorial-4-a-colored-cube/
https://stackoverflow.com/questions/22636069/cube-rotation-opengl
https://shankarrajagopal.github.io/DOWNLOAD/CG_LAB_P3.pdf