Link para o repositório do GitHub com o código fonte:
https://github.com/VictorC98/abcg/tree/main/AtividadesPraticas/Atividade2
Link para o Github Pages com a aplicação:
https://victorc98.github.io/abcg/ShootingRange/

Técnicas utilizadas

A aplicação consiste em um campo de tiro com 3 alvos vermelhos, 4 alvos azuis e um alvo amarelo, todos usam um modelo obtido no site free3d.com. O usuário tem controle sobre a câmera e sobre cada grupo de alvos, as instruções de controle estão no canto inferior esquerdo da tela, usando a fonte Raleway. Foi utilizado a estrutura SDL_Event da SDL, divididos em pressionamento de teclas (SDL_KEYDOWN) e liberação de teclas (SDL_KEYUP), para controlar os alvos, decidindo quais alvos devem ficar orientados para cima, com rotação 0 e quais devem ficar "deitados", com rotação 90°. O alvo amarelo pode ser movimentado para a esquerda e para a direita (eixo x), desaparecendo e reaparecendo do lado oposto caso o alvo se desloque demais para um lado.

Foi utilizado o SDL_MOUSEMOTION para capturar os movimentos relativos do mouse, junto com o SDL_SetRelativeMouseMode(SDL_TRUE) para prender o mouse no centro da tela e escondê-lo, assim permitindo o controle da câmera através do mouse. Como isso dificulta fechar a aplicação, foi adicionado o botão Escape para liberar o mouse. Também foi usado o SDL_MOUSEBUTTONDOWN para aplicar um zoom na câmera e o SDL_MOUSEBUTTONUP para cancelar o efeito. O efeito de zoom foi feito alterando o Field of View da câmera.

A câmera usada é uma câmera LookAt que utiliza as seguintes funções para movimentação:

- dolly: faz a câmera ir para a frente e para trás ao longo da direção de visão .
- truck: faz a câmera deslizar para os lados.
- pan: faz a câmera girar em torno de seu eixo y, permitindo olhar para os lados.
- rotatex: faz a câmera girar em torno de seu eixo x, permitindo olhar para cima e para baixo.

A função dolly computa o vetor "para frente" subtraindo a posição da câmera do ponto LookAt. Depois muda a posição da câmera e do ponto LookAt

multiplicando o vetor “para frente” por uma velocidade predeterminada, baseada em deltaTime.

A função truck computa o vetor para a esquerda fazendo o produto vetorial dos vetores para frente e para cima, depois muda a posição da câmera e do ponto LookAt da mesma maneira que a função dolly.

A função pan faz o movimento de girar a câmera em torno de seu eixo y. Isso é feito alterando apenas o ponto LookAt por meio de uma série de transformações de matrizes.

A função rotatex faz o movimento de girar a câmera em torno de seu eixo x. Ela funciona da mesma forma que a função pan, mas ela usa o vetor de direção para direita (m_right) na transformação de rotação, enquanto a pan usa o vetor para cima. Isso faz com que, ao olhar para trás, a câmera inverta os controles e mexer o mouse para cima faz a câmera olhar para baixo, e vice-versa. Para consertar esse problema, a câmera inverte a velocidade de rotação se o usuário olhar para trás.

Depois de cada função da câmera, a matriz de visão é recalculada através da função computeViewMatrix.

Os alvos utilizam o identificador GL_TRIANGLES na função de renderização glDrawArrays para criar o conjunto de triângulos que compõem cada modelo, enquanto o chão e a parede utilizam o identificador GL_TRIANGLE_STRIP para formar uma polilinha. Todos os modelos usam geometria indexada, ou seja, usam um arranjo ordenado de posições de vértices armazenado no VBO, e um arranjo de números inteiros que representam os índices para esses vértices armazenados no EBO. Todos os modelos têm suas cores definidas por uma variável uniforme de cor RGBA.

O arquivo lookat.vert é o vertex shader, que cuida do processamento dos vértices, incluindo a posição do vértice no espaço da câmera e a intensidade da cor RGB dos objetos. O arquivo lookat.frag é o fragment shader, que processa cada fragmento individualmente e multiplica a cor RGB por 0.5 se a câmera estiver dentro de um objeto.