## Trabalho do Grau A

### Visualizador de Modelos 3D

Implementar aplicativo para leitura de objetos 3D (formato OBJ) e renderização com shaders (OpenGL 4+). Deve ser seguida a estrutura de dados definida no material de modelagem e implementada a visualização de objetos 3D (com arquivos OBJ, MTL e texturas conforme definido no material sobre formato OBJ).

Características que devem estar presentes no software:

- As faces deve ser mantidas por triângulos, portanto, caso o modelo possua faces com 4 ou mais lados, elas devem ser "quebradas" em triângulos.
- Além do desenho das faces, devemos considerar a aplicação de mapeamento de texturas.
- Ajustes de escala, rotação e translação para exibir o objeto num tamanho e posição adequada na tela.
- Simular um *First Person Shooter*, onde com a barra de espaço (ou outra tecla apropriada), o usuário dá um tiro na direção da câmera. De início, o tiro segue a direção da câmera e tem sua posição "incrementada" (translação) até colidir com algum objeto da cena. Se o tiro colidir com algum objeto da cena:
  - caso este objeto seja do tipo "eliminável", o tiro e o objeto devem ser removidos da cena;
  - caso este objeto seja "não eliminável", o tiro deve ser refletido para outra direção.
- Se o tiro não colide em nada ou se ele passou o "tempo" de translação, deve ser removido da cena. O tempo pode ser ajustado pelo aluno, conforme observado em seus testes.
- Os objetos definidos no arquivo de configurações devem ter uma propriedade indicando se são "elimináveis".
- Definir o *bounding-box* do objeto para definir a sua caixa de colisão. Pode ser uma esfera, um cubo ou qualquer outra primitiva 3D para definir a área de colisão.
- Devem ser carregados diversos objetos OBJ e dispostos numa única cena "coerente".
- Deve ser possível navegar na cena com câmera virtual. Neste sentido, mapear as telas dos direcionais (ou WASD) do teclado para tratar o "movimento" da câmera:
  - → - rotaciona a câmera para a direita
  - ← - rotaciona a câmera para a esquerda
  - ↑ - move câmera para frente
  - ↓ - move câmera para trás.

### Dicas:

- Comece com programação da estrutura de classes.
- Os OBJs devem ser carregados com leitor próprio.
- Para representação do array de vértices (`vec3`), normais e mapeamento de texturas (`vec2`), utilizar a estrutura de dados para vértice (`vec2` e `vec3`) que é disponibilizada pela biblioteca `GLM`. O array pode ser implementado com `vector` da `STL`.
- Criar um VAO para cada grupo. Portanto, a cena deve ser desenhada por grupo de cada objeto 3D.
- O objeto da classe Obj3D (que representa os objetos da cena) deve ter atributo para manter a matriz de transformação que posição e ajusta o objeto na cena.
- Quando da criação dos VBOs, não esquecer resolver a indexação dos vértices de cada face. Desta forma, deve ser criado um array de float e que deve ser preenchido com as coordenadas dos vértices das faces de cada grupo.

### Entrega e Critérios de avaliação

- O código-fonte do trabalho bem como vídeo explicativo de captura de tela deve ser entregue no Moodle. O vídeo deve conter a explicação do programa executando e do que foi feito em código-fonte. Vídeo no formato MP4 e, preferencialmente, de no máximo 15 min.
- Critérios:

## Trabalho do Grau A

- Carga de arquivos e desenho de OBJ (**3 pts**)
- Carga de arquivo e representação da cena (**3 pts**)
- Reprodução do tiro, colisão e reflexão (**2 pts**)
- Avaliação geral sobre implementação, qualidade do código e funcionamento geral (**1 pt**)
- Exercício screen-saver (**1 pt**)