Trabalho Prático

Resumo

Este trabalho tem como objetivo aplicar os principais conceitos de programação 3D abordados nas aulas. Assim, os estudantes, organizados em **grupos de 4 elementos**, deverão desenvolver um programa em C++ que, através do uso das bibliotecas OpenGL, GLEW, GLM e GLFW, realize a renderização de um cenário 3D, aplicando as respetivas texturas, iluminação, e realize uma animação em resposta a um evento de teclado. Este trabalho culminará na entrega, via Moodle, dos ficheiros com o código fonte, bem como numa apresentação/defesa oral do trabalho realizado.

Realização do trabalho prático

O ficheiro "**P3D-TP.zip**" contém:

- Enunciado do trabalho;
- Modelos 3D de bolas de bilhar (ficheiros .obj e .mtl).
- Imagens de texturas para as 15 bolas (em formato .jpg).

O trabalho deverá ser desenvolvido em grupo, sendo que os **grupos deverão ser constituídos por 4 elementos**.

O trabalho deverá ainda ser "defendido", pelos alunos, numa apresentação oral a realizar em aula.

Objetivo

Este trabalho tem como objetivo o desenvolvimento de um programa em **C++** (respeitando o paradigma de Programação Orientada a Objetos) que realize a renderização de um cenário 3D, aplicando as respetivas texturas e iluminação, e realize uma animação (movimento de uma das bolas) em resposta a um evento de teclado.

Os requisitos são aqui agrupados em 4 passos, de modo a orientar os estudantes para a resolução do trabalho. Note, no entanto, que **só deve fazer a entrega de uma única versão do seu programa (a versão final), que responda a todos os requisitos**.

- Passo 1:

- Implementar a gestão de janelas e interface com o utilizador através da biblioteca GLFW;
- Implementar a manipulação de matrizes e vetores através da biblioteca GLM;
- ([1]) Implementar a renderização de um paralelepípedo (para simular a mesa de bilhar) através da biblioteca OpenGL (incluindo a GLEW);
- ([2]) A coloração dos fragmentos do paralelepípedo deve ser realizada de modo que cada face apresente uma cor distinta (nota que não é esperado que nesta fase estejam implementados os efeitos de iluminação);
- A aplicação deverá permitir ao utilizador realizar zoom através da *scroll wheel* do rato;
- A aplicação deverá permitir navegar (rodar) em torno do centro do paralelepípedo, através de movimentos do rato.
- Implementar um minimapa da mesa de bilhar, a ser exibido no canto superior direito da janela da aplicação. Este minimapa deve apresentar uma perspetiva *top view* da mesa. O minimapa não deve ser afetado pela iluminação.

[1] Não é um requisito para avaliação. Serve apenas de guia para o desenvolvimento do trabalho.

- Passo 2:

- Os alunos deverão desenvolver uma biblioteca C++ (respeitando o paradigma de Programação Orientada a Objetos) capaz de:
    - Carregar os dados de posição dos vértices, normais, e coordenadas de textura dos ficheiros .obj;
    - Reorganizar a informação dos vértices de acordo com a informação das faces;
    - Enviar para a memória de GPU (VAO e respetivos VBO) os dados dos vértices;
    - Identificar no ficheiro .obj o nome do ficheiro .mtl;
    - Carregar as propriedades do material (ficheiro .mtl), incluindo a imagem de textura;
    - Carregar a imagem de textura para uma unidade de textura do OpenGL;
    - Disponibilizar função para renderização do objeto.

    Embora o formato .mtl suporte múltiplos materiais, para este trabalho assuma que um ficheiro .mtl terá apenas um único material a aplicar a todo o objeto.
- A aplicação deverá fazer uso dessa biblioteca C++ para carregar os modelos, compor e renderizar um cenário 3D que apresente 15 bolas dispostas sobre uma mesa de snooker.

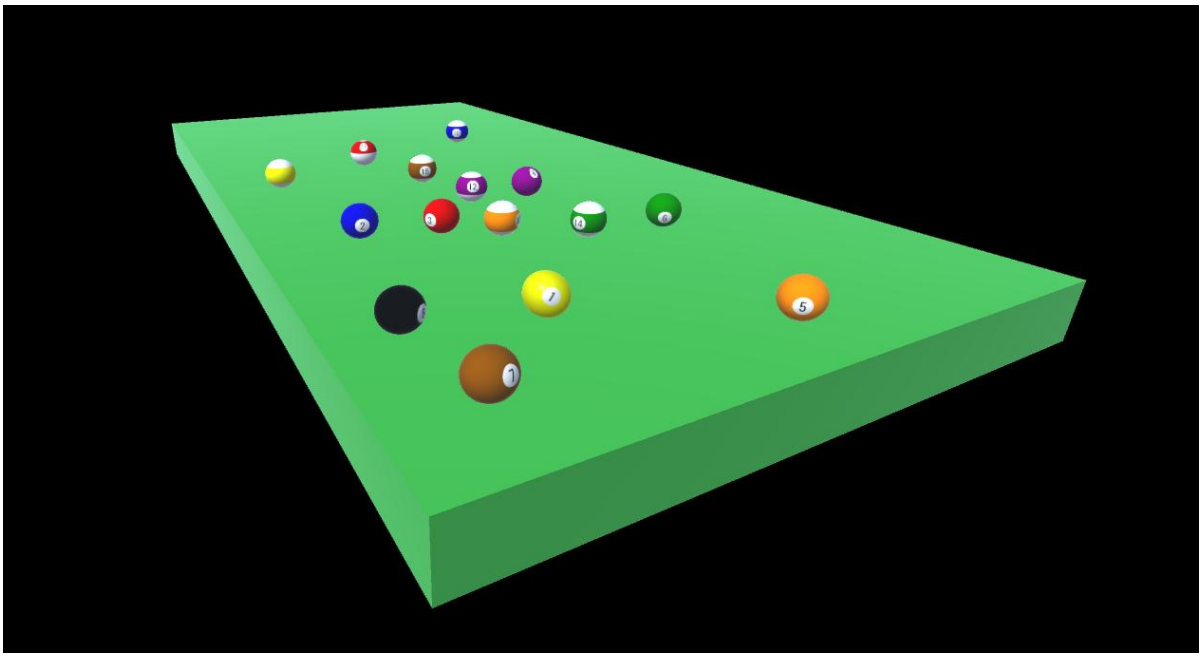

Figura 1: Exemplo de composição com mesa e bolas de bilhar.

- A câmara deverá permanecer imóvel numa posição que permita visualizar toda a mesa, estando orientada para o centro da mesa.
- A biblioteca deverá possuir um namespace próprio no qual será definida uma classe (e suas respetivas funções-membro) para gestão do processo de renderização. São de implementação obrigatória as seguintes funções-membro:
    - `Load(const std::string obj_model_filepath)`
        - Esta função tem como objetivo carregar para a memória do CPU os dados do ficheiro .obj, cujo caminho é passado como argumento.
        - Deverá ainda carregar para a memória do CPU os materiais e texturas associados ao modelo.

- `Install(void)`
  - Gera os VAO e VBO necessários e envia os dados do modelo (vértices, coordenadas de textura e normais) para a memória do GPU.
- `Render(glm::vec3 position, glm::vec3 orientation)`
  - Função para renderizar o modelo na posição definida por **`position`** e orientação definida por **`orientation`**.
- Deverão ainda ser criadas funções que permitam a associação entre os dados do modelo (previamente carregados para o GPU através da função **`Install()`**) e os atributos de um programa shader.
- Para além destas funções obrigatórias, poderão ser criadas outras que se afigurem necessárias.

- Passo 3:

- Deverá ser possível aplicar uma qualquer combinação de 4 fontes de luz que incidam sobre o objeto:
  - Luz ambiente;
  - Luz direcional;
  - Luz pontual;
  - Luz cónica.
- Os parâmetros de cada uma das fontes de luz ficam ao critério de cada grupo;
- A aplicação deverá permitir ao utilizador ativar/desativar cada fonte de luz, através de uma tecla:
  - '1' – Ativar/desativar fonte de luz ambiente;
  - '2' – Ativar/desativar fonte de luz direcional;
  - '3' – Ativar/desativar fonte de luz pontual;
  - '4' – Ativar/desativar fonte de luz cónica.

- Passo 4:

- Implemente a animação de movimento (translação e rotação) de uma das bolas de bilhar. Essa animação deverá ser desencadeada assim que o utilizador premir a tecla "Espaço". A animação deverá parar quando a bola atingir uma outra bola de bilhar ou os limites da mesa.

Formato dos Modelos 3D

O formato OBJ permite representar objetos 3D em ficheiros de texto que sigam uma determinada estrutura. Para este trabalho, os modelos 3D de cada bola encontram-se definidos em dois de ficheiros de texto (.obj e .mtl) e uma imagem de textura (.jpg). Exemplo:

- Ball1.obj → Informação relativa aos vértices, coordenadas de textura, normais e faces.
- Ball1.mtl → Informação relativa ao material.
- PoolBalluv1.jpg → Imagem de textura do modelo 3D.

A biblioteca deverá receber o caminho para o ficheiro .obj e extrair a informação relativa a:

- Nome do ficheiro com extensão .mtl
- Dados de posição dos vértices (v) → Em cada linha representa-se a coordenada em **x**, **y** e **z**.
- Dados das coordenadas de textura (vt) → Em cada linha representa-se a coordenada em **s** e **t**.
- Dados das normais (vn) → Em cada linha representam-se os valores do vetor em **x**, **y** e **z**.
- Faces (f)

Com o nome do ficheiro .mtl, a biblioteca deverá realizar a leitura dos parâmetros que definem o material:

- Ka – Coeficiente de reflexão da luz ambiente
- Kd – Coeficiente de reflexão da luz difusa
- Ks – Coeficiente de reflexão da luz especular
- Ns – Expoente especular (representa o brilho do objeto. O mesmo que *material shininess*.)

Deve ainda ler o nome do ficheiro da imagem de textura:

- map_Kd

Ignore os restantes parâmetros que o ficheiro possa conter.

Para mais informações sobre o formato OBJ, aconselha-se a consulta de:

`https://en.wikipedia.org/wiki/Wavefront_.obj_file`

Avaliação

Serão tomados como critérios de avaliação os seguintes factores:

- Respeito pelas regras de entrega do trabalho;
- Qualidade do programa:
  - organização, clareza e qualidade do código fonte;
  - utilização da linguagem C++ e respeito pelo paradigma de POO;
  - desenvolvimento das funcionalidades descritas no enunciado do trabalho;
  - nível de otimização das funcionalidades implementadas;
  - funcionamento correcto do programa;
  - valor acrescentado[2].
- Qualidade do código e respetivos comentários, bem como da apresentação oral:
  - descrição correta e completa da estrutura do programa;
  - descrição das técnicas aplicadas no desenvolvimento das funcionalidades.
- Conhecimento que cada estudante demonstra relativamente ao código apresentado.

A natureza coletiva da realização de um trabalho em grupo não prejudica o facto de a avaliação ser individual para cada um dos elementos do grupo.

Prazos

A realização do trabalho pressupõe a entrega de um ficheiro ZIP contendo os ficheiros com o código fonte.

O trabalho deverá ser remetido ao docente, via Moodle, até à data e hora definida (também disponível na página Moodle da UC). O docente reserva o direito de não avaliar os trabalhos entregues após aquela data e hora.

A entrega do trabalho prático deverá respeitar **obrigatoriamente** os seguintes requisitos:

- Os ficheiros com o código fonte (**não** inclua o projeto do Visual Studio, nem o executável) deverão ser colocados num ficheiro **ZIP** com o nome "**P3D-TP-GY-xxxx-xxxx-xxxx-xxxx.zip**", em que:

[2] Por valor acrescentado entende-se a forma como o trabalho se destaca (positivamente) dos restantes.

- **xxxx** deverá ser preenchido com o número de aluno de cada um dos elementos do grupo;
- **GY** deverá ser preenchido com **G1** para grupo 1, **G2** para grupo 2, etc.

- No nome do ficheiro zip só deverão constar os números dos alunos que efetivamente realizaram o trabalho, e que, portanto, se irão submeter à avaliação.
- Apenas 1 (um) elemento de cada grupo deverá submeter o trabalho.

O prazo de entrega termina no dia **27 de maio**, às **23:00**. **Não serão considerados trabalhos entregues após esta data**. A **defesa dos trabalhos** será realizada durante as aulas de **30 de maio**.

Conduta ética

A falta de transparência em avaliações, presenciais ou não, é naturalmente ilegal e imoral. Todas as fontes utilizadas para suporte a trabalhos devem ser obrigatoriamente e claramente referenciadas. Qualquer plágio, cópia ou conduta académica imprópria será penalizada com a anulação do trabalho. Caso se verifique a existência de trabalhos notoriamente similares entre grupos, todos os trabalhos similares serão anulados.