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 3 elementos**, deverão desenvolver um programa em C++ que, através do uso das bibliotecas OpenGL, GLM e GLFW, realize a renderização de um modelo 3D (disponibilizado em formato OBJ), aplicando as respetivas texturas, iluminação, e um efeito de deformação do modelo 3D. Este trabalho culminará na entrega, via Moodle, dos ficheiros com o código fonte, bem como numa apresentação oral do trabalho realizado.

Realização do trabalho prático

No ficheiro "**P3D-TP.zip**" é apresentada a respectiva descrição do trabalho e objectivos, contendo ainda:

- Enunciado do trabalho;
- Modelo 3D ("Iron_Man", com ficheiros .obj, .mtl e .tga).

O trabalho deverá ser desenvolvido em grupo, sendo que os **grupos deverão ser constituídos por 3 elementos**.

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

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[1].
- 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 colectiva da realização de um trabalho em grupo não prejudica o facto de a avaliação ser individual para cada um dos elementos do grupo.

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

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-xxxx-xxxx-xxxx.zip**" (em que **xxxx** deverá ser preenchido com o número de aluno de cada um dos elementos do grupo) e remetidos ao docente através da plataforma Moodle.
- Apenas 1 (um) elemento de cada grupo deverá submeter o trabalho.

O prazo de entrega termina no dia **4 de junho**, às **23:00**. **Não serão considerados trabalhos entregues após esta data**. A **defesa dos trabalhos** será realizada durante as aulas de **6 e 7 de junho**. Qualquer alteração à data de entrega e/ou apresentação será indicada a todos os alunos via Moodle.

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 (onde por exemplo se tenha alterado apenas os nomes das variáveis de um outro código) entre grupos, todos os trabalhos similares serão anulados.

Regulamento Disciplinar dos Estudantes do IPCA:

https://ipca.pt/wp-content/uploads/2016/04/PR-91-AprovRegDiscEstudantes-1.pdf

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 do modelo 3D disponibilizado, aplicando as respetivas texturas e iluminação, e que permita ainda a possibilidade de aplicar um efeito de deformação ao modelo original.

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;
- ([2]) Implementar a renderização de um cubo através da biblioteca OpenGL (incluindo a GLEW);
- ([2]) A coloração dos fragmentos do cubo 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 modelo 3D, através de movimentos do rato.

[2] 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 do ficheiro .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.
- A aplicação deverá fazer uso dessa biblioteca C++ para carregar e renderizar o modelo 3D fornecido. O modelo deverá ser aplicado a 3 objetos, localizados nas seguintes coordenadas: (0,0,0) , (-2,-1,-5) e (2,-1,-5). A câmara deverá permanecer imóvel na posição (0,2,5), estando orientada para a coordenada (0,2,0).

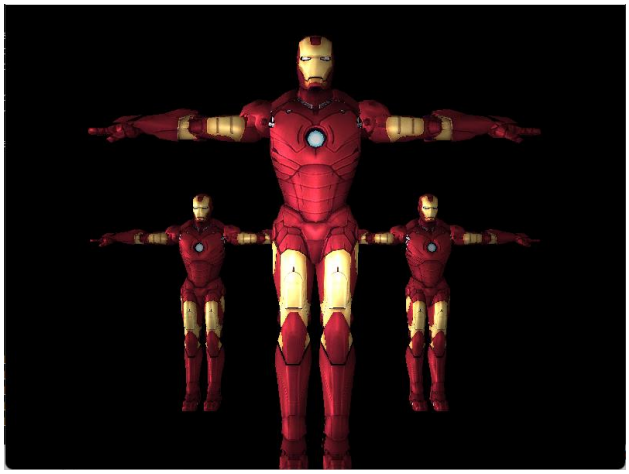

Figura 1: Exemplo com a renderização dos 3 objetos nas respetivas coordenadas.

- 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:
    - **`Read(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.
    - **`Send(void)`**
        - Gera os VAO e VBO necessários e envia os dados do modelo (vértices, coordenadas de textura e normais) para a memória do GPU.
    - **`Draw(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 **`Send()`**) 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:

- Dotar o vertex shader da possibilidade de ativar/desativar um efeito de deformação do modelo 3D carregado. Esta deformação deverá variar em função do tempo.

Formato do Modelo 3D

O formato OBJ permite representar objetos 3D em ficheiros de texto que sigam uma determinada estrutura. Para este trabalho, o modelo 3D encontra-se definido em dois de ficheiros de texto (.obj e .mtl) e uma imagem de textura (.tga):

- Iron_Man.obj → Informação relativa aos vértices, coordenadas de textura, e normais.
- Iron_Man.mtl → Informação relativa ao material.
- Iron_Man.tga → Imagem de textura do modelo 3D.

A biblioteca deverá receber o caminho para o ficheiro Iron_Man.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

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

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