Projeto 3: Modelação hierárquica e aplicação de texturas

Change Log:

- 30/Nov/2018 @ 11h00 - Atualização da secção com os detalhes técnicos.
- 30/Nov/2018 @ 11h00 - Atualização da secção sobre a entrega do trabalho (grafo de cena em PDF como requisito).
- 26/Nov/2018 @ 20h10 - Publicação do enunciado.

Objetivo

Neste projeto vamos programar uma aplicação que nos permitirá visualizar e controlar um expositor de objetos giratório e elevatório (um modelo hierárquico). Sobre o expositor deverá estar colocada uma das primitivas (cubo, cilindo ou esfera), estando esta primitiva centrada na plataforma do expositor.

Uma representação esquemática do expositor (cuja especificação das dimensões fica a vosso cargo) , pode ser observada abaixo. A esfera visualizada na imagem é o objeto em exposição. As primitivas a usar para a modelação são:

- cubo
- esfera
- cilindro

Funcionalidades

Em termos de **manipulação do modelo**, o utilizador deverá poder efetuar as seguintes operações:

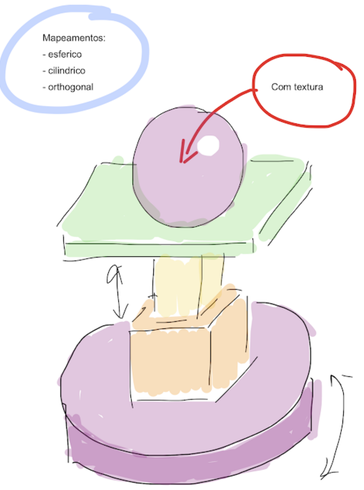
- elevar e recolher a plataforma do expositor
- sendo a base giratória, rodar o expositor nos dois sentidos.

Para qualquer destas interações implemente limites adequados que impeçam o modelo de alcançar configurações indesejáveis (por exemplo, separação completa dos dois componentes do pilar telescópico).

Outra funcionalidade pretendida é a da visualização do objeto exposto com **aplicação duma textura**, usando um dos seguintes **mapeamentos clássicos**:

- mapeamento ortogonal
- mapeamento cilindrico
- mapeamento esférico

Claro está que o utilizador deverá poder selecionar o mapeamento a usar num dado momento. Nenhum dos outros objectos deverá ser visualizado com mapeamento de texturas.



Projeção

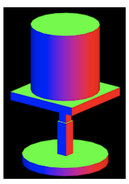
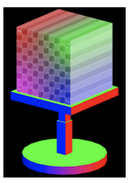
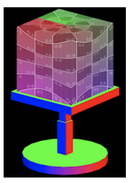
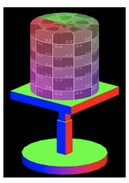
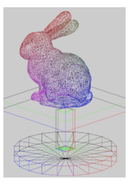
A cena deverá ser visualizada com uma projeção axonométrica, devendo o conjunto aparecer aproximadamente centrado no canvas. Os objetos não poderão apresentar deformação resultante do mapeamento para o canvas, nem deverão ser recortados.

Modos de visualização dos objetos

A visualização das primitivas será efetuada em dois modos distintos:

- Malha de arame — apenas são visíveis as arestas correspondentes às faces, interligado os vértices
- Superfícies pintadas — visualização os triângulos que formam das faces do objeto. Poderá usar uma adaptação dos shaders usados no 2º trabalho, visualizando-se o vetor normal sob a forma da cor do fragmento.

Eis alguns exemplos de visualização com diferentes objetos:

Interação com o utilizador

A aplicação deverá permitir as seguintes interações:

- Escolher o objeto a visualizar (esfera, cilindro e cubo);
- Escolher o modo de visualização (malha de arame ou superfícies pintadas);
- Alternar entre os diferentes mapeamentos de coordenadas de textura;
- Modificar qualquer dos parâmetros do grafo de cena, manipulando assim o modelo hierárquico.

Detalhes técnicos

Variáveis (conjunto mínimo) a passar para os shaders

Cada modelo enviará para o vertex shader os seguintes atributos:

- vPosition — posição do vértice;
- vNormal — `vec3` descrevendo o vetor perpendicular da superfície naquele vértice.

Para além destes atributos, o vertex shader deverá ainda receber as seguintes matrizes (`uniform mat4`):

- mProjection — tratará do volume de visualização da projeção axonométrica.
- mModelView —Composição das matrizes de modelação e de orientação da câmara na seguinte forma: mModelView = mView * mModel. A matriz mView tratará de orientar o objeto, de acordo com a projeção escolhida, de modo a que a direção de projeção seja a do eixo Z (com recurso à função lookAt ou M_axo). A matriz mModel será distinta para cada primitiva, de acordo com o percurso efetuado no grafo de cena.

Texturas

O mapeamento pode ser definido de 3 formas alternativas, cabendo a vocês escolher a melhor (menos trabalhosa e com maior qualidade):

- por via de atributos com as coordenadas de textura fornecidos pela aplicação ao vertex shader;
- por adição das coordenadas de textura ao nível do vertex shader;
- por adição das coordenadas de textura ao nível do fragment shader.

A textura usada nos exemplos está acessível aqui.

Regras e Informação Adicional

Composição dos grupos

Os trabalhos práticos deverão ser realizados por grupos de 2 alunos dum mesmo turno prático. A entrega do trabalho a título individual terá que ser devidamente justificada e autorizada pelo respectivo docente do turno prático.

Entrega

O trabalho será entregue via Google Classroom, usando a classe do respectivo turno prático até às **23h59 de 8 de Dezembro 2018**.

Ambos os alunos do grupo terão que efetuar a **submissão do trabalho**.

Na pasta Common apenas deverão estar ficheiros a que todos os alunos têm acesso, nomeadamente as bibliotecas fornecidas (MV.js, webgl-utils, initShaders), bem como os objetos fornecidos (sphere.js, cylinder.js, etc.). Os restantes ficheiros deverão estar todos localizados na pasta onde se encontra o .html e o .js principal da aplicação.

Assim, apenas deverão entregar os ficheiros localizados na mesma pasta onde se encontra o .html e o .js da aplicação. Na pasta deverá ainda constar um PDF com o grafo de cena, devendo ser usada a notação seguida nas aulas teóricas.

Avaliação

Os trabalhos serão avaliados pelo respetivo docente das aulas práticas e discutidos com os respetivos alunos em data a definir oportunamente.