Projeto 3: Projeções de objectos 3D, iluminação (e texturas)

ChangeLog

- 2020/11/23 @ 15h00 - Adicionada a secção de Detalhes técnicos.
- 2020/11/23 @ 13h50 - Publicação do enunciado.

Descrição Geral

Pretende-se desenvolver uma aplicação que permita ao utilizador visualizar um conjunto de objectos (um de cada vez), sob o efeito de iluminação e segundo uma projeção à sua escolha (dum leque disponível) e cujos parâmetros poderão ser manipulados interativamente.

A visualização do objecto corrente deverá ser efetuada sem deformação do mesmo, num visor que deverá ocupar toda a largura da janela, ficando uma área por baixo desse mesmo visor reservada para os elementos da interface com o utilizador.

A visualização do objecto poderá ser efetuada em dois modos:

- Malha de arame (tecla 'W')
- Preenchido (tecla 'F')

O resultado final será ainda afetado pela ativação ou desativação de duas técnicas de remoção de superfícies ocultas:

- Algoritmo z-buffer (tecla 'Z')
- Método do produto interno - back face culling (tecla 'B')

O programa deverá permitir a utilização das seguintes projeções:

- Projeção ortogonal (alçado principal, planta, alçado lateral direito)
- Projeção axonométrica (isometria, dimetria (A=42º, B=7º), trimetria (A=54° 16', B=23° 16') e livre (o utilizador controla diretamente os valores de gamma e de theta).
- Projeção perspetiva (com centro de projeção em (0,0,d))

Sempre que a opção relativa à projeção caia numa projeção com valores pré-definidos, o controlo dos parâmetros da projeção deverão estar bloqueados, sendo desbloqueados quando se escolhe a opção livre.

Para a realização deste trabalho não poderá usar diretamente as matrizes de projeção deduzidas nas aulas teóricas, mas apenas as funções ortho(), perspective() e lookAt().

Objectos Suportados

Os objetos a suportar pela aplicação são:

- Esfera de raio 0.5, centrada na origem;
- Cubo de aresta 1, centrado na origem;
- Torus centrado com plano y=0 (R=0.5 , r=0.2);
- Cilindro centrado na origem, base e topo paralelos ao plano y=0 e de altura 1.
- parabolóide desenvolvido no trabalho prático 2

Os objetos fornecidos encontram-se na secção **Recursos** na página do google classroom (turno teórico).

Valores iniciais

Quando o programa arranca, o objecto a visualizar deverá ser o cubo, sendo para tal usada a projeção axonométrica dimétrica identificada atrás. A visualização será feita em modo de malha de arame, com o algoritmo de culling de faces e de z-buffer desativados.

Interação

Interface reduzida

O programa deverá apresentar apenas os elementos da sua interface com o utilizador que farão sentido num dado momento. Por exemplo, se a projeção escolhida é uma projeção axonométrica, não fará sentido apresentar os elementos que permitem controlar outras projeções que não a axonométrica.

Interface com memória

O programa deverá ser, também, capaz de memorizar o estado de cada tipo de projeção, permitindo a alteração do tipo de projeção sem que, ao retornar a uma projeção onde já se efetuaram alterações aos parâmetros, se tenha que recomeçar de novo.

Ampliação/Redução

A aplicação deverá permitir a ampliação do objeto projetado, por redução do volume de visualiação. Em sentido oposto a redução deverá ser igualmente suportada através dum aumento do volume de visualização. Tais funcionalidades deverão ser implementadas em resposta ao evento da janela *onwheel*, ativado quando se faz scroll usando a roda de scroll do rato.

Iluminação

O programa deverá permitir a visualização dos objetos sob a ação duma única fonte de luz, escolhendo o utilizador se ela deverá ser direcional ou pontual. A posição ou direção da fonte de luz são também controladas pelo utilizador. A cor/intensidade da fonte de luz é um parâmetro também possível de ser controlado pelo utilizador.

As caracteristicas do material que compõe o objeto também poderão ser manipuladas pelo utilizador (coeficientes de reflexão difusa, especular e *shininess).*

Opcional (apenas 1 será contabilizado)

Controlo livre da projeção perspetiva

O utilizador poderá controlar a direção do eixo central do volume de visão, de forma livre, usando o rato. A ideia é controlar de forma independente a orientação desse eixo e a distância do centro de projeção à origem.

Aplicação de texturas

A visualização poderá permitir a visualização dos objetos com mapeamento de texturas. Os mapeamentos suportados deverão ser os seguintes:

- esférico
- cilíndrico
- ortogonal

Detalhes técnicos

Z-Buffer

A ativação do algoritmo de z-buffer em WebGL, consiste na ativação do teste de profundidade:

```
gl.enable(gl.DEPTH_TEST);
```

Sendo também necessário limpar o conteúdo do z-buffer a cada geração da imagem dum quadro (frame):

```
gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
```

Para desativar o z-buffer bastará chamar:

```
gl.disable(gl.DEPTH_TEST);
```

Backface Culling

A ativaçao da remoção de faces "viradas de costas" é conseguida da seguinte forma:

```
gl.enable(gl.CULL_FACE);
```

Ao ativar a remoção de faces, podemos ainda especificar que faces pretendemos remover:

- `gl.FRONT` - As faces orientadas positivamente no ecrã
- `gl.BACK` - As faces orientadas negativamente no ecrã
- `gl.FRONT_AND_BACK` - Ambas as faces (com orientação qualquer)

Para tal usa-se a função:

```
gl.cullFace();
```

A determinação da "positividade" duma face é controlada pela função:

```
gl.frontFace(mode)
```

Sendo *mode* ou `gl.CW` (clock-wise) ou `gl.CCW` (counter-clock-wise).

Para desligar a funcionalidade de *face culling* usa-se:

```
gl.disable(gl.CULL_FACE);
```

Shaders

Neste trabalho, os *shaders* são relativamente simples. No caso do vertex shader é importante que o mesmo possa receber quer a matriz **mView**, quer a matriz **mProjection**, efetuando a determinação do vértice no espaço de recorte (volume de visualização normalizado) usando a expressão:

```
gl_Position = mProjection * mView * mModel * vPosition;
```

Note-se que, apesar de na expressão acima estar presente a matriz **mModel**, a mesma deverá ser sempre a matriz identidade, pelo que poderá ser omitida, caso queiram.

Quanto ao atributo **vNormal**, embora não seja usado/necessário neste trabalho, poderemos visualizar o seu valor mapeando-o para um varying que será usado pelo fragment shader para afetar a cor do fragmento.

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. O prazo limite de entrega é às **23h59m do dia 13 de Dezembro de 2020**. Cada grupo deverá submeter um ficheiro .zip, cujo conteúdo será o seguinte:

- Ficheiro .html
- Ficheiros .js da aplicação (excluindo os ficheiros fornecidos)

Todos os scripts adicionais (residentes na directoria Common) não deverão ser incluídos na entrega.

Avaliação

O trabalho, sem a parte opcional será cotado para **17 valores**, sendo os restantes **3 valores** atribuídos à realização de um dos opcionais. Os trabalhos serão avaliados pelo respetivo docente das aulas práticas e discutidos com os respetivos alunos em data a definir oportunamente.