Aula 9

- Interação usando o rato e o teclado
- Estrutura de dados simples para armazenar malhas triangulares
- Utilização de texturas
- Transparência

1.1 Interação usando o rato e o teclado

Analise o exemplo **WebGL_example_26.html**.

O objetivo deste exemplo é ilustrar o modo como podem ser usados o teclado e o rato para permitir uma interação intuitiva com o(s) modelo(s) representado(s) na cena.

Identifique as novas funcionalidades de interação disponibilizadas e analise as funções que permitem:

- Usar as teclas do cursor para incrementar / decrementar a velocidade de rotação em torno dos eixos XX e YY.

- Usar as teclas *Page Up* e *Page Down* para fazer *zoom out* / *zoom in*.

- Usar o rato para alterar a orientação do(s) modelo(s) da cena, por seleção de um ponto do *canvas* e deslocamento do cursor, mantendo premido um dos botões do rato.

Tarefa:

- Escolha duas teclas (p.ex., Z e X) e controle a velocidade de rotação em torno do eixo ZZ.

1.2 Estrutura de dados simples para armazenar malhas triangulares

Analise o exemplo **WebGL_example_27.html**.

O objetivo deste exemplo é ilustrar a utilização de uma estrutura de dados simples para armazenamento da malha triangular representando a superfície de um modelo.

É usado um *array* armazenando as coordenadas dos vários vértices, um outro *array* armazenando a cor (RGB) associada a cada um dos vértices, e um *array* armazenando os índices dos vértices (*CCW*) definindo cada um dos triângulos da malha.

A utilização desta estrutura de dados implica algumas alterações às funções **initBuffers()** e **drawModel()**.

Tarefa:

No ficheiro **WebGL_example_27.js** está definido um cubo usando essa simples estrutura de dados.

Analise:

- O *array* dos vértices – Como aparecem os vértices definindo cada uma das faces do cubo?
- O *array* das cores – Que cor está associada a cada face do cubo?
- O *array* dos índices definindo cada um dos triângulos que compõem a malha – Como é subdividida cada uma das faces do cubo?

Tarefa:

Analise as alterações efetuadas nas funções **initBuffers()** e **drawModel()**.

Note o modo como se desencadeia o desenho dos triângulos usando a função **gl.drawElements()**.

Questão:

O que aconteceria à representação do cubo se não existissem vértices repetidos na estrutura de dados?

Tarefa:

Faça essa alteração na estrutura de dados e verifique o que acontece.

1.3 Utilização de texturas

Analise o exemplo **WebGL_example_28.html**.

Atenção:

Alguns *browsers* só permitem o carregamento de imagens de textura a partir de um servidor (p.ex., Chrome), enquanto outros não apresentam essa restrição (p.ex., Edge ou Firefox).

O objetivo deste exemplo é ilustrar a utilização de texturas, o que implica significativas alterações ao código dos exemplos anteriores.

Note que deixa de haver uma cor associada a cada vértice de um modelo, uma vez que cada face será representada usando uma textura.

Tarefa:

Identifique as alterações mais significativas realizadas:

- Mapeamento dos pontos extremos da textura nos quatro vértices definindo cada face do cubo.
- Alterações ao código dos ***shaders***.
- Correspondentes alterações na função **initShaders()**.
- Funções para carregamento de uma imagem de textura e estabelecimento das características da sua representação.
- Alterações ao código das funções **initBuffers()** e **drawModel()**.

Tarefa:

Usando uma imagem à sua escolha, efetue o seu carregamento e verifique se é corretamente representada nas várias faces do cubo.

Note que deve usar uma imagem quadrada, em que o número de linhas / colunas deve ser uma potência de 2.

Sugestão:

Modifique o código de modo a poder representar uma imagem diferente em cada uma das faces do cubo.

E mesmo imagens diferentes para cada uma das instâncias do cubo.

1.4 Transparência – Blending

Analise o exemplo **WebGL_example_29.html**.

Atenção:

Alguns *browsers* só permitem o carregamento de imagens de textura a partir de um servidor (p.ex., Chrome), enquanto outros não apresentam essa restrição (p.ex., Edge ou Firefox).

O objetivo deste exemplo é ilustrar a simulação de materiais transparentes.

Isso é conseguido usando uma alternativa ao algoritmo do *Z-Buffer* para representar corretamente os *pixels* no *frame buffer* – veja como isso efetuado.

Quando se usa *blending*, uma função de ponderação combina a informação de cor de vários fragmentos que se sobrepõem para determinar a cor dos diferentes *pixels*.

Tarefa:

Identifique as alterações mais significativas realizadas:

- Alterações ao código dos ***shaders*** – Analise o seu significado.
- Correspondentes alterações na função **initShaders()**.
- Alterações ao código da função **drawModel()**, para estabelecer a função de ponderação e o valor do parâmetro **alpha**.

Tarefa:

Acrescente a possibilidade de o utilizador incrementar / decrementar o valor de **alpha**, entre 0 e 1, usando o teclado.

Verifique o que acontece à representação do cubo.