

Computação Gráfica (L.EIC)



Tópico 4

Aplicação de texturas

Objetivos

- Definir coordenadas de textura de forma adequada.
- Explorar os diferentes modos de aplicação de textura.
- Combinar o uso de materiais com texturas para obter uma aparência realista.

Trabalho prático

À semelhança do trabalho anterior, será necessário fazerem capturas de ecrã em alguns pontos do enunciado, bem como assinalar versões do código no *Git* com *Tags*. Os pontos onde tal deve ser feito estão assinalados ao longo do documento e listados numa check list no final deste enunciado, sempre assinalados com os ícones  (captura de uma imagem) e  (tags).

Preparação do Ambiente de Trabalho

Devem descarregar o código disponibilizado para este trabalho no Moodle, e colocar o conteúdo da pasta **tp4** contida no ficheiro .zip na pasta correspondente do repositório. Devem também copiar o ficheiro da classe **MyUnitCubeQuad**, criado no **TP2**, para a pasta deste trabalho. A classe **MyQuad** utilizada para os planos do cubo composto é fornecida no código base desta aula prática, pelo que poderá optar por usar esse ficheiro em substituição do **MyQuad** do **TP2** (verifique se as classes são consistentes e compatíveis).

1. Aplicação de texturas

O mapeamento de texturas é uma forma de atribuir informação armazenada em formato *bitmap* a diferentes zonas das superfícies 3D desenhadas. Um dos seus usos mais comum é o de mapear partes ou a totalidade de uma imagem a uma geometria, de forma a acrescentar detalhe visual sem aumentar o número de vértices e sem acrescentar

complexidade à geometria (outros tipos de mapeamento incluem, por exemplo, *bump mapping* e *normal mapping*, mas que não serão explorados neste trabalho).

No contexto de OpenGL/WebGL, uma textura de duas dimensões pode resultar do carregamento de uma imagem *bitmap*, e que é carregada para um buffer, que posteriormente pode ser acedido usando duas dimensões vulgarmente identificadas como s e t (ou noutros contextos como u e v), e cujas coordenadas são normalizadas entre 0 e 1 (ver fig. 1).

Nota: É importante reparar que a representação dos eixos de coordenadas de textura pode ser diferente em diferentes contextos. No caso dos nossos projetos em WebGL/WebCGF, a origem (0,0) corresponde ao *canto superior esquerdo*.

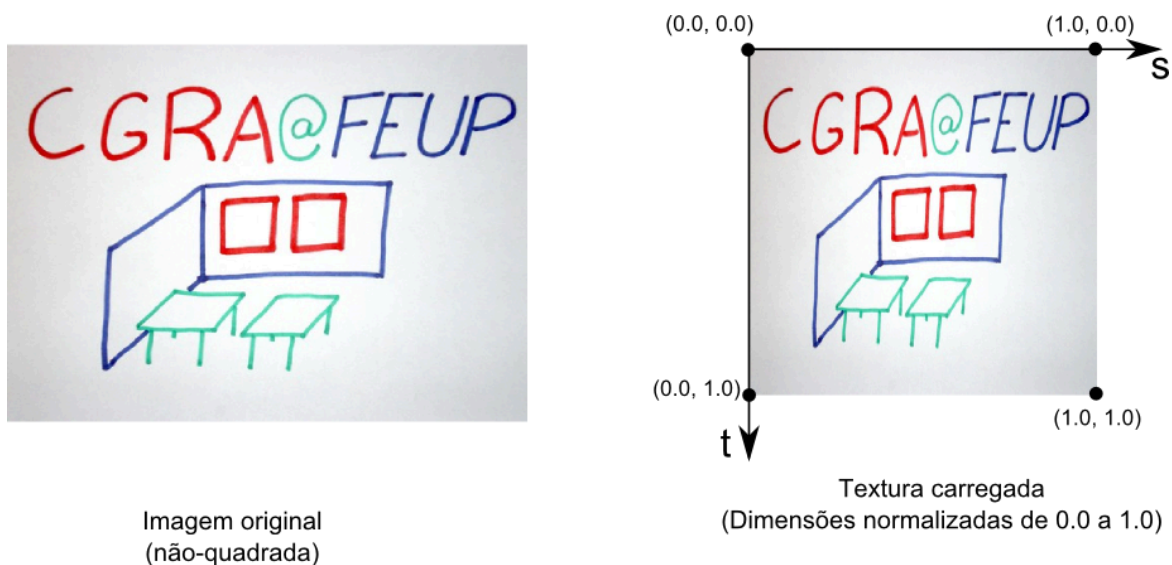


Figura 1: Imagem e correspondente textura carregada

Uma textura previamente carregada pode ser aplicada a uma dada geometria - no caso mais básico, um triângulo - fazendo o mapeamento entre os vértices da geometria e os pontos da imagem que lhes estarão associados, definindo para cada vértice uma **coordenada de textura** (ver fig. 2, a) e b)).

Conceptualmente, podemos considerar que estamos a definir o "recorte da imagem" que será aplicado ao triângulo em questão, sendo que caso o "recorte" não tenha as mesmas proporções do triângulo original, a imagem será distorcida de acordo (ver fig. 2, c) e d)).

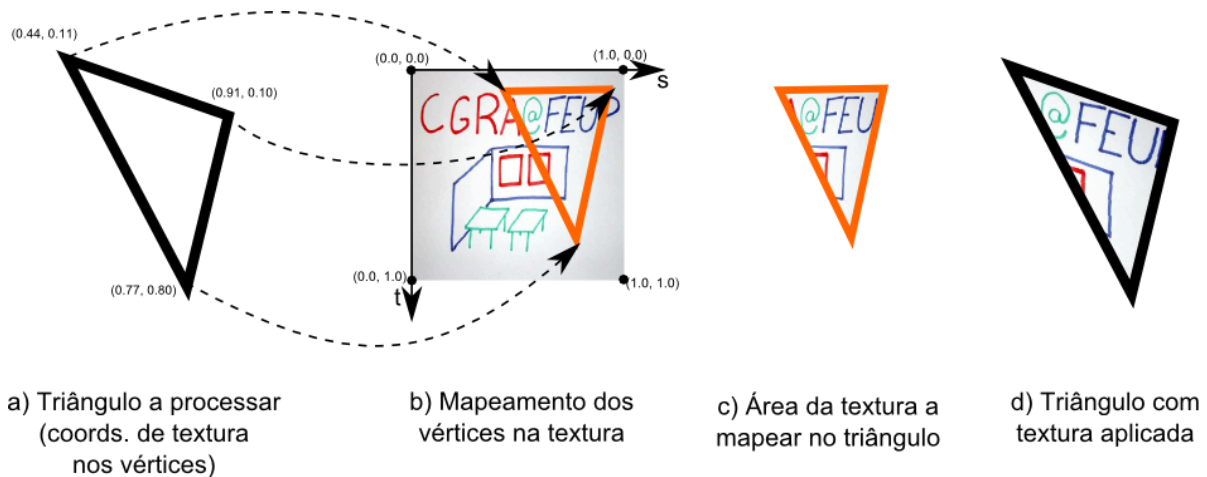
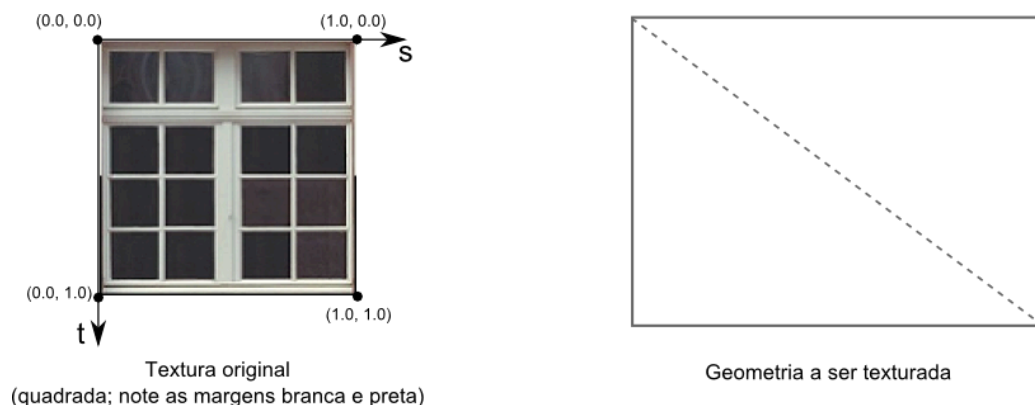


Figura 2: Mapeamento entre triângulo e textura definindo coordenadas de textura por vértice.

2. Modos de Wrapping de Texturas

No exemplo explorado até agora, as coordenadas de textura associadas a cada vértice encontram-se na gama normalizada de 0.0 a 1.0. No entanto, é possível indicar valores fora dessa gama, quando pretendemos, por exemplo, ter várias repetições da mesma imagem num polígono, ou mapear a totalidade da imagem apenas numa parte do polígono.

A forma como os valores fora da gama $[0..1]$ são utilizados na aplicação de uma textura é controlada definindo o modo de **wrapping**. Os modos de **wrapping** suportados variam um pouco entre versões de OpenGL, no caso do WebGL os modos possíveis são '**REPEAT**', '**CLAMP_TO_EDGE**' e '**MIRRORED_REPEAT**'. Na figura 3 estão ilustrados alguns exemplos de como manipular as coordenadas de textura em cada modo para obter diferentes efeitos. Note que o modo de **wrapping** pode ser diferente nas duas dimensões s e t .



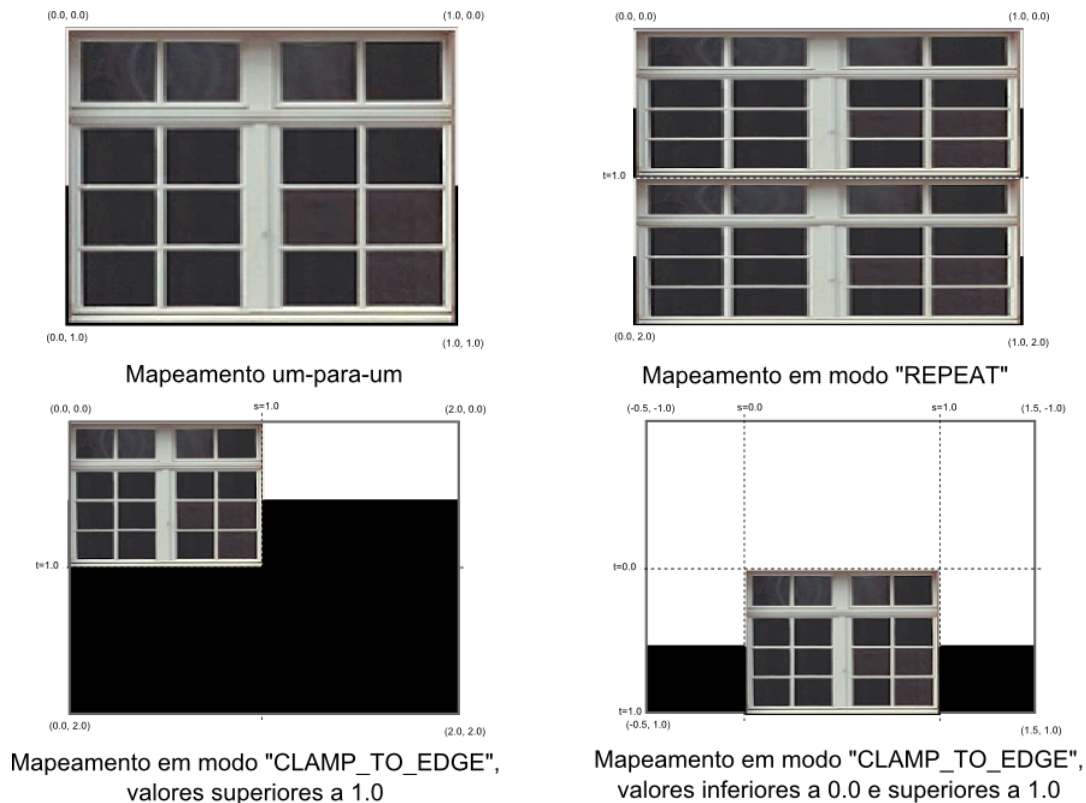


Figura 3: Aplicação de uma textura utilizando repetição ou *clamping*.

Note como no modo de 'CLAMP_TO_EDGE' as margens da imagem são estendidas ao longo das zonas de coordenadas fora da gama [0..1]

Experiências

Pretende-se nesta aula explorar a definição de coordenadas de textura em objetos e os diferentes modos de wrapping para a aplicação de diferentes texturas.

A cena fornecida contém um objeto do tipo **MyQuad** ao qual foi aplicado um material do tipo **CGFappearance** chamado **quadMaterial**. A interface contém um menu *dropdown* para a escolha de texturas do tipo **CGFtexture** (começando sem seleção), dois *dropdowns* para a seleção de modos de wrapping, e *sliders* para controlar as coordenadas de textura associadas aos quatro cantos do retângulo.

Analise como as texturas (objetos da classe **CGFtexture**) são inicializadas e aplicadas no código da classe **MyScene**.

1. Selecione a textura 'Board' na interface. Com o modo de wrap das coordenadas S e T em 'Repeat', altere os valores das coordenadas de textura de forma a que obtenha três colunas e duas linhas da imagem no objeto.
2. Reinicie a cena, e selecione a textura 'Floor' na interface. Mantendo o modo de wrap em 'Repeat', altere os valores das coordenadas de textura de forma a que a imagem seja invertida na vertical.
3. Altere o modo de wrap das coordenadas S e T para 'Clamp to Edge' e veja as diferenças no mapeamento da textura.
4. Reinicie a cena, e selecione a textura 'Window' na interface. Com o modo de wrap das coordenadas S e T em 'Clamp to Edge', altere o valores das coordenadas de

textura de forma a que a janela apareça centrada na geometria, ocupando metade da altura e largura totais, como mostrado na figura 4.

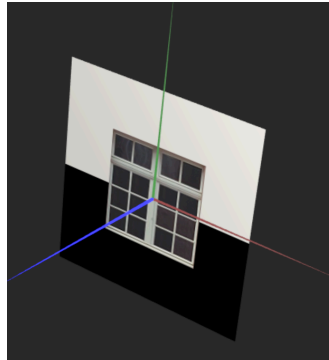


Figura 4: Janela centrada no objeto **MyQuad**.

5. Experimente alternar os modos de wrapping para S e T e observe as diferenças.

Exercícios



Inclua na pasta do código fornecido os ficheiros das classes **MyTangram** e das classes de todas as peças. Crie uma checkbox que permita esconder o objeto **MyQuad** e respectivo material **quadMaterial** de forma a que a cena fique vazia.

Aplicação de texturas ao Tangram

1. Crie um novo material na inicialização a ser aplicado na função *display* da classe **MyTangram**, no objeto de **MyDiamond**. Defina como textura desse material a imagem 'tangram.png' (ver exemplos no código).
2. Defina as coordenadas de textura do **MyDiamond** de forma a que as arestas da peça do losango na imagem coincidam com as arestas do objeto. Para ajudar no processo de determinação das coordenadas de textura a atribuir a cada vértice, sugere-se que abra uma cópia de 'tangram.png' num editor de imagem para anotar os eixos S e T tal como na figura 1 deste enunciado, identifique os vértices do losango nessa figura, e determine quais as suas coordenadas nesse espaço S, T (valores entre 0.0 e 1.0).
3. As coordenadas de textura são definidas criando na função *initBuffers* do objeto um *array* adicional **this.texCoords** com um par de coordenadas para cada vértice previamente declarado no array **this.vertices**:



```
this.texCoords=[
    s0, t0,
    s2, t2,
    ...
    sn, tn
];
```

4. Repita os dois passos anteriores para cada uma das outras peças do Tangram, de forma a que cada peça tenha mapeada a sua representação da imagem.

(1 ) (1 )

Aplicação de texturas a um cubo composto por planos

1. Copie a classe **MyUnitCubeQuad**, desenvolvida no **tp2** que define um novo cubo unitário utilizando um objeto do tipo **MyQuad**, desenhado várias vezes para definir as faces.
2. Altere o seu construtor para receber como parâmetros opcionais seis texturas (**CGFtexture**), a serem aplicadas às suas seis faces pela ordem *topo* (+Y), *frente* (+Z), *direita* (+X), *trás* (-Z), *esquerda* (-X), *fundo* (-Y). Altere a função *display()* para aplicar as texturas adequadamente.
3. Crie uma instância de **MyUnitCubeQuad** passando como parâmetros a textura 'mineSide.png' para as faces laterais, e as texturas 'mineTop.png' e 'mineBottom.png' para as faces de topo e de fundo, respetivamente.
4. Repare como as texturas ficam pouco definidas. Isso deve-se ao facto de terem originalmente dimensões de 16x16 pixels, mas na verdade estarem a cobrir uma área de desenho muito superior. Por omissão, nestes casos é feita uma **interpolação linear** das cores (**LINEAR FILTERING**, ver filtragem nos slides da teórica).
5. Encontre no código de exemplo o comando que permite alterar o tipo de filtragem usado (comentado originalmente na função *display()* de **MyScene**). Use-o para as texturas do cubo para atingir o efeito pretendido, ativando esse modo depois de ativar a textura e antes de desenhar as faces a afetar.

(2 ) (2 )