

Computação Gráfica (MIEIC)

Aula Prática 5


Aplicação de shaders


Objetivos

- Aprender conceitos básicos de shaders
- Utilizar vertex shaders para manipular geometria de objetos
- Utilizar fragment shaders para manipular cores e texturas em cena

Trabalho prático

Ao longo dos pontos seguintes são descritas várias tarefas a realizar. Algumas delas estão

anotadas com o ícone  (captura de imagem). Nestes pontos deverão, capturar uma imagem da aplicação para disco (p.ex. usando Alt-PrtScr em Windows ou Cmd-Shift-3 em Mac OS X para capturar para a clipboard e depois gravar para ficheiro num utilitário de gestão de imagens à escolha). No final de cada aula, devem renomear as imagens para o formato **"ex5-t<turma>g<grupo>-n.png"**, em que **turma** e **grupo** corresponde ao número de turma e grupo definido no ficheiro de grupos TP, e **n** corresponde ao número fornecido no exercício (p.ex. **"ex5-t1g01-1.png"**).

Nas tarefas assinaladas com o ícone  (código), devem criar um ficheiro **.zip da pasta que contém o vosso código (tipicamente na pasta 'ex5', se tiverem código noutras pastas incluam-no também)**, e nomeá-lo como **"ex5-t<turma>g<grupo>-n.zip"**, (com turma, grupo e n identificados tal como descrito acima **"ex5-t1g01-1.zip"**).

No final (ou ao longo do trabalho), um dos elementos deverá submeter os ficheiros via Moodle, através do link disponibilizado para o efeito. Bastará apenas um elemento do grupo submeter o trabalho.

Shaders

Estude com atenção os slides adicionais fornecidos com esta aula prática, e tenha presente os recursos adicionais disponibilizados no Moodle.

Experiências

Na cena do TP5 podemos observar um bule de chá, mesh tradicionalmente encontrada em projetos baseados em OpenGL. Observe atentamente os vários tipos de shaders disponibilizados e estude o código correspondente do vertex shader e do fragment shader.

1. Selecionando o shader “*Passing a scale as Uniform*”, altere na interface o valor do *scaleFactor* e verifique o que acontece aos vértices do bule, alternando o modo “Wireframe”. Estude o código verificando como é passado o valor da variável.
2. Faça o mesmo com o shader “*Multiple textures in VS and FS*”.
3. Estude o código dos primeiros 6 conjuntos de shaders.
4. Na **ShaderScene** observe a função *update(t)*, função standard da **CGFScene** que recebe o tempo atual em milissegundos. Verifique como é utilizado no *vertexShader* da animação. Note que para essa função ser invocada, teve de ser definido o período de atualização da cena na sua função *init*, usando o método *setUpdatePeriod*.
5. Observe o efeito do shader “Sepia”, e observe como a cor é alterada no *fragment shader*.
6. Observe a mudança na cor da textura aplicada no objeto com o shader “Convolution”, que implementa no *fragment shader* um *edge detector*.
([https://en.wikipedia.org/wiki/Kernel_\(image_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing))).

Exercícios

1. Shaders no Teapot

1. Crie novos *vertex* e *fragment shaders* por forma a colorir o bule em função da posição ocupada na janela pelos fragmentos - amarelo na metade superior, azul na metade inferior. Para isso, deve usar a posição dos vértices após a transformação (tal como armazenada em *gl_Position*). Para tal crie uma variável *varying* no *vertex shader* que guarde a posição do vértice para ser passada para o *fragment shader*. Aí, recupere esse valor e apresente a cor azul se $y > 0.5$ e amarela se menor.

(1 )

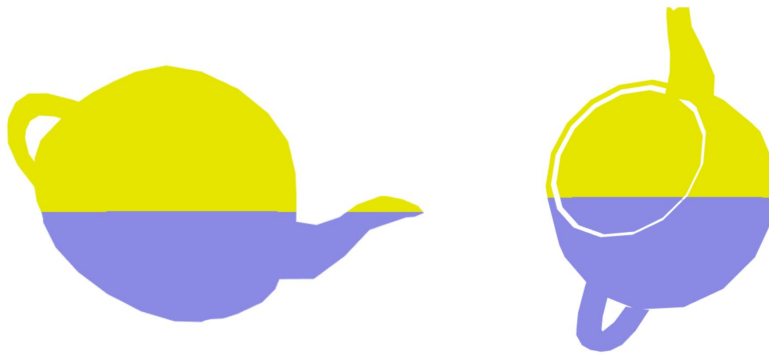


Figura 1: Exemplificação do bule com mudança de cor de acordo com as coordenadas em cena.

2. Altere o shader de animação para criar um efeito de translação para trás e para a frente no eixo XX, seguindo uma onda sinusoidal. O efeito de translação deverá depender do *scaleFactor* da interface. Dica: acrescente um novo offset ao *aVertexPosition* no vertex shader.
3. Crie um novo *Fragment Shader* baseado no de Sêpia que converta a cor para tons de cinza (Grayscale¹). Para tal converta todos os componentes RGB da cor para $L = 0.299R + 0.587G + 0.114B$.

(2 )

2. Shaders no Plane: Efeito de água

1. Crie dois novos shaders ***water.vert*** e ***water.frag*** baseado-se nos ***texture2.vert*** e ***texture2.frag***, adicione-os ao projeto e disponibilize-os na interface. Selecione o plano ***Plane***, fornecido no código exemplo, na cena (usando a *checkbox* na interface).
2. Substitua as texturas em cena com as imagens ***waterTex.jpg*** e ***waterMap.jpg***. Com os shaders criados na alínea anterior, verifique que vê uma textura de água com manchas de tom vermelho escuro.
3. Altere o *vertex shader* para utilizar o ***waterMap.jpg*** como mapa de alturas da textura de água (ou seja, cada vértice deve ser deslocado de acordo com o valor de uma das componentes de cor da textura).
4. Anime o plano através dos dois shaders criados, onde se deve variar a associação das coordenadas de textura aos vértices e fragmentos ao longo do tempo, para obter um efeito semelhante ao que pode ser visto no exemplo na página seguinte (link para vídeo).

(3 )(1 )

¹ https://en.wikipedia.org/wiki/Grayscale#Converting_color_to_grayscale

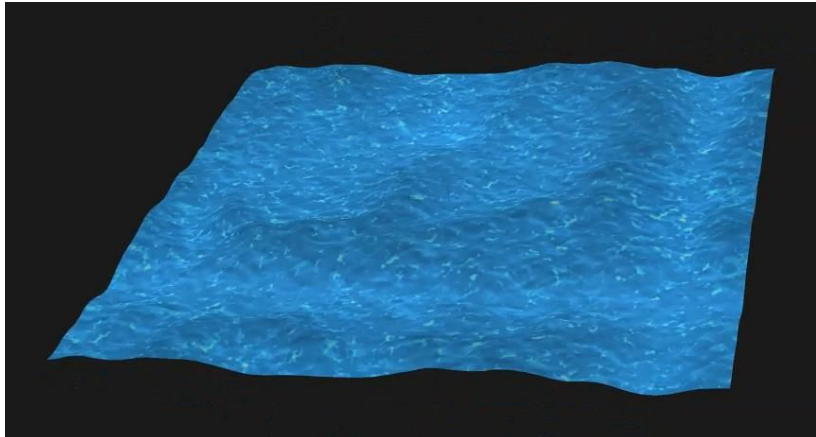




Figura 1: Water shader

Vídeo: <https://drive.google.com/open?id=1gSgOrhpVg10GxwIXMBcRewxV5dU8o1FH>

Checklist

Até ao final do trabalho deverá submeter as seguintes imagens e versões do código via Moodle, **respeitando estritamente a regra dos nomes**:

-  Imagens (3): 1, 2, 3 (nomes do tipo "ex5-t<turma>g<grupo>-n.png")
-  Código em arquivo zip (1): 1 (nomes do tipo "ex5-t<turma>g<grupo>-n.zip")