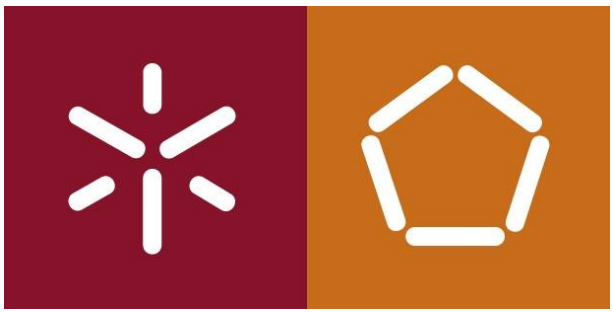

Universidade do Minho

Mestrado Integrado em Engenharia Informática

Computação Gráfica

Sistema Solar

Joan Rodriguez (a89980)
Mário Santos (a70697)
Pedro Costa (a85700)
Rui Azevedo (a80789)
1 de Junho de 2020

Conteúdo

1 Introdução

Nesta fase do trabalho prático, é pretendido que se acrescente ao sistema texturas e iluminação. Para isso, o gerador foi alterado de maneira a gerar as normais necessárias para o processo de iluminação bem como as coordenadas para as texturas. Quanto ao *engine*, apenas foi desenvolvida a funcionalidade de aplicar texturas às primitivas da cena.

Numa primeira fase do relatório, é apresentado o processo de cálculo das normais e coordenadas para as texturas e, de seguida, as alterações feitas no *engine* que permitem integrar as novas funcionalidades.

2 Generator

O *generator* foi alterado de modo a conseguir gerar normais para cada vértice, usadas no processo de iluminação, bem como coordenadas para as texturas que serão aplicadas às primitivas. Para isso, a classe *Primitive* passou a ter mais dois vectores, um para guardar as coordenadas das texturas e outro para guardar os vectores das normais dos vértices. No método da classe onde são gerados os vértices das primitivas, estes valores são calculados e adicionados aos respectivos vectores.

2.1 Cálculo de normais

Neste secção irá ser apresentada a lógica do cálculo das normais de cada vértice de cada primitiva.

2.1.1 Plano

As normais do plano são bastante simples, como se trata de um plano definido em XoZ, cada vértice contém o mesmo vector normal, designadamente, (0,1,0).

2.1.2 Caixa

As normais de uma caixa, mantêm vectores constantes para cada face. Uma vez que uma caixa é composta por 6 faces, iremos ter 6 vectores normais diferentes:

- Face frontal XY : (0,0,1)
- Face traseira XY : (0,0,-1)
- Face esquerda YZ : (-1,0,0)
- Face direita YZ : (1,0,0)
- Face de cima XZ : (0,1,0)
- Face de baixo XZ : (0,-1,0)

2.1.3 Cone

Os vetores normais para a base do cone são todos (0,-1,0). Quanto às normais das faces das laterais já requerem mais computação. Quanto à face lateral, e de maneira a obter um

vector unitário, a normal correspondente é $(sin(\alpha), cos(atan(h/radius), cos(\alpha))$ onde h é a altura do cone, $radius$ o seu raio e α o ângulo de uma determinada *slice*.

2.1.4 Esfera

A criação das normais da esfera são praticamente iguais à criação dos vértices mas apenas com uma ligeira diferença. Enquanto no cálculo dos vértices da esfera, quando se usam as coordenadas esféricas para a criação dos vértices é necessário aplicar o raio da primitiva na fórmula de cálculo. Neste caso, uma vez que nos basta o vector unitário, apenas foi necessário remover o raio da fórmula, ficando com o seguinte formato $(sin(\alpha) * cos(\beta), sin(\beta), cos(\alpha) * sin(\beta))$

2.1.5 Teapot

Quanto às normais dos *bezier patches*, a alteração necessária a fazer foi apenas, no processo de geração de pontos, fazer o produto externo entre as derivadas u e v.

2.2 Texturas

As coordenadas das texturas são representadas num plano bidimensional com os valores a variar entre 0 e 1. O objectivo é, para cada primitiva mapear os vértices com as respectivas coordenadas das texturas.

2.2.1 Plano

O plano tem um mapeamento praticamente directo com as coordenadas das texturas:

- canto superior direito : (1,1)
- canto superior esquerdo : (0,1)
- canto inferior direito : (0,1)
- canto inferior esquerdo : (0,0)

2.2.2 Caixa

Para o cálculo das texturas de uma caixa, imaginamos uma grelha com o mesmo número divisões de uma caixa, assumindo que a grelha tem a dimensão de 1×1. No cálculo dos vértices, a coordenada a adicionar é sempre $(i/division, j/division)$ ou $((division -$

$i)/division, (division - j)/division$), dependendo da face que está a ser mapeada, onde i e j são as variáveis de controlo que, em cada instância, tem o valor de uma porção das dimensões da caixa.

2.2.3 Cone

Numa primeira fase foi feito o mapeamento para a base do cone. Este processo passou por representar as coordenadas de um círculo centrado em $(0, 0)$ com raio unitário e, de seguida, aplicar uma transformação em ambos os eixos para o centrar no ponto $(0.5, 0.5)$. O mapeamento tem então a seguinte fórmula: $(0.5 * sin(\alpha) + 0.5, 0.5 * cos(\alpha) + 0.5)$.

De seguida, procedeu-se ao mapeamento das faces do cone que têm a seguinte fórmula: $(j/slices, i/slices)$ onde i e j são as variáveis de controlo do cone.

2.2.4 Esfera

As coordenadas de textura da esfera têm o seguinte formato: $(j/slices, (stacks-i)/stacks)$. Neste processo, pode-se imaginar uma grelha com as dimensões $slices \times stacks$ onde, dado uma *stack* e uma *slice* da esfera, estas são mapeadas para o respetivo ponto da grelha.

2.2.5 Teapot

Para o *teapot*, as coordenadas podem ser mapeadas da seguinte maneira $(1 - u, 1 - v)$, onde u e v são os valores de tesselagem.

3 Engine

A nível do Engine apenas fomos capazes de incluir, na sua totalidade, a funcionalidade das texturas. O parser é capaz de reconhecer luzes e desenvolvemos as classes base onde seriam guardadas e representadas as diversas luzes mas não tivemos tempo para as implementar.

3.1 Texturas

As texturas funcionam de forma bastante simples. A primeira alteração que tivemos de fazer foi no que diz respeito ao Parser para permitir ler texturas do *xml*. Fizemos ainda uma outra alteração que nos permitiu agora ler os pontos para as normais e para as texturas presentes, desde esta fase, no ficheiro de cada primitiva.

A partir daí apenas tivemos de dar enable ao estado **GL_TEXTURE_ARRAY** para permitir o uso de texturas. De seguida repetimos o processo feito previamente para os VBO's em que alocamos um buffer para guardar as texturas e preenchemo-lo antes até de começar a dar render de qualquer coisa. Após este momento aproveitamos para carregar as texturas e guardamos o *texID* num vetor para ser usado mais tarde. A função utilizada para carregar texturas foi extraída dos guiões e usa mipmapping.

Numa fase de render, para cada primitiva temos apenas de fazer o **bind** do array buffer correspondente e evocar o pointer de texturas. De seguida, damos ainda **bind** das texturas 2D e, após desenhar a primitiva, desfazemos o **bind** das texturas.

4 Conclusão

Nesta fase conseguimos finalmente obter resultados visualmente apelativos devido ao uso das texturas, compondo assim o que já se perceciona como sendo um sistema solar! A iluminação iria complementar o visual apelativo do programa final pois é bastante pouco natural olhar para objetos que não possuem qualquer iluminação.

É uma pena não termos conseguido acabar o projeto uma vez que é a primeira e, para vários membros do grupo, possivelmente a última vez que temos a oportunidade de trabalhar numa área tão gráfica.

Foi interessante perceber melhor como funciona a Computação Gráfica, algo com que, ao fim e a cabo, lidamos praticamente todos os dias!