

Universidade do Minho
Escola de Engenharia

Computação Gráfica

Fase 4 - Iluminação e Texturas

Relatório Trabalho Prático

Grupo 40

LEI - 3º Ano - 2º Semestre

A90969 Gonçalo Gonçalves
A98695 Lucas Oliveira
A89292 Mike Pinto
A96208 Rafael Gomes

Braga,
26 de abril de 2024

Conteúdo

1	Introdução	3
2	Generator	4
2.1	Torus	4
2.1.1	Normais	4
2.1.2	Coordenadas de Textura	4
2.2	Esfera	5
2.2.1	Normais	5
2.2.2	Coordenadas de Textura	5
2.3	Plano	5
2.3.1	Normais	5
2.3.2	Coordenadas de Textura	5
2.4	Cone	6
2.4.1	Normais	6
2.4.2	Coordenadas de Textura	6
2.5	Box	6
2.5.1	Normais	6
2.5.2	Coordenadas de Textura	7
2.6	Patches	7
2.6.1	Normais	7
2.6.2	Coordenadas de Textura	7
3	Engine	9
3.1	Texturas	9
3.2	Iluminação	10
3.3	Correções Fase 3	10
4	Demo Scene	11
4.1	Textura Mercúrio	13
4.2	Textura Vénus	13
4.3	Textura Terra e Lua	14
4.4	Textura Marte e respectivas Luas	14
4.5	Textura Júpiter	15
4.6	Textura Saturno	15
4.7	Textura Urano	16
4.8	Textura Neptuno	16
5	Conclusão	17

List a de Figuras

4.1	Novas curvas de <i>Catmull-Rom</i> utilizadas para a translação dos planetas.	11
4.2	Sistema Solar com iluminação.	12
4.3	Sistema Solar sem iluminação.	12
4.4	Textura do Planeta Mercúrio.	13
4.5	Textura do Planeta Vénus.	13
4.6	Textura do Planeta Terra e Lua.	14
4.7	Textura do Planeta Marte e respetivas Luas.	14
4.8	Textura do Planeta Júpiter.	15
4.9	Textura do Planeta Saturno e Anel.	15
4.10	Textura do Planeta Urano.	16
4.11	Textura do Planeta Neptuno.	16

Capítulo 1

Introdução

Este relatório é elaborado no âmbito da quarta e última fase do trabalho prático da Unidade Curricular de Computação Gráfica do 2.º semestre do 3.º ano do curso de Engenharia Informática da Universidade do Minho.

A presente fase tem como principal objetivo a aplicação de luzes, sombras e texturas ao modelo do sistema solar desenvolvido ao longo das fases anteriores.

Neste documento, serão apresentados os processos desenvolvidos e aplicados até esta quarta e ultima fase.

Capítulo 2

Generator

De forma a ser possível colocar iluminação e aplicar texturas nas primitivas é necessário modificar o gerador desenvolvido anteriormente. Para isto, deve-se calcular as normais e as coordenadas de textura de cada ponto para cada primitiva, e escrevê-las no ficheiro.

2.1 Torus

2.1.1 Normais

As normais para a superfície de um *torus* podem ser calculadas a partir das derivadas parciais das equações paramétricas. No entanto, uma maneira mais direta é considerar a direção do vetor a partir do ponto no tubo para o centro do tubo.

$$N_x = \cos(v)\cos(u) \quad (2.1)$$

$$N_y = \cos(v)\sin(u) \quad (2.2)$$

$$N_z = \sin(v) \quad (2.3)$$

Isto resulta num vetor unitário que é perpendicular à superfície no ponto (u,v) .

2.1.2 Coordenadas de Textura

As coordenadas de textura são então calculadas com base na divisão da superfície do *torus* em segmentos, determinados pelos parâmetros *rings* e *sides*. *Rings* representa o número de divisões ao longo do eixo principal do *torus* (em torno de u), enquanto *sides* representa o número de divisões ao longo do círculo gerador do toro (em torno de v). Para cada ponto calculado as coordenadas de textura são determinadas pela fração da posição atual em relação ao número total de divisões. Assim, para um ponto com índice i ao longo dos *rings* e j ao longo dos *sides*, as coordenadas de textura (u,v) são dadas por:

$$u = \frac{i}{rings} \quad (2.4)$$

$$u = \frac{j}{sides} \quad (2.5)$$

2.2 Esfera

2.2.1 Normais

Para uma esfera, a normal de um ponto é simplesmente o vetor do centro da esfera até esse ponto, normalizado. A normalização de um vetor $P(x,y,z)P(x,y,z)$ é feita dividindo cada componente pela magnitude do vetor. A magnitude é calculada como:

$$r = \sqrt{x^2 + y^2 + z^2} \quad (2.6)$$

Portanto, as normais são calculadas da forma que se encontra em 2.7.

$$N = \left(\frac{x}{r}, \frac{y}{r}, \frac{z}{r} \right) \quad (2.7)$$

2.2.2 Coordenadas de Textura

Para mapear uma textura 2D na superfície da esfera, utilizamos coordenadas de textura u e v , que variam de 0 a 1. Essas coordenadas são calculadas com base na divisão da esfera em *stacks* e *slices*:

$$u = \frac{a}{slices} \quad (2.8)$$

$$v = \frac{h}{stacks} \quad (2.9)$$

em que $a \in [0, slices]$ e $h \in [0, stacks]$.

2.3 Plano

2.3.1 Normais

No caso do plano gerado, que está situado no plano XY com Y=0, todas as normais são iguais e apontam na direção do eixo positivo Y. Portanto, para cada vértice do plano, a normal é definida como (0, 1, 0).

2.3.2 Coordenadas de Textura

Para um plano dividido em *div* subdivisões, cada quadrado da grade tem suas coordenadas de textura calculadas de forma a cobrir uniformemente a textura sobre toda a superfície.

O cálculo das coordenadas de textura para cada vértice é feito da seguinte forma: (i/div) e $((i+1)/div)$ definem as coordenadas horizontais (U) da textura para a subdivisão *i*. (j/div) e $((j+1)/div)$ definem as coordenadas verticais (V) da textura para a subdivisão *j*.

2.4 Cone

2.4.1 Normais

Para calcular as normais, primeiro determinamos os vetores perpendiculares a dois vetores dados (a e b). Isso é feito utilizando o produto vetorial, que resulta num vetor perpendicular ao plano formado por a e b . Após calcular o vetor perpendicular, é necessário normalizá-lo para obter uma normal unitária. A normalização envolve dividir cada componente do vetor pelo seu comprimento. Para cada triângulo do cone, a normal é calculada utilizando três pontos ($p1$, $p2$, $p3$). Os vetores u e v são obtidos subtraindo $p1$ de $p2$ e $p3$, respectivamente. A normal é então calculada usando o produto vetorial destes vetores e normalizado.

2.4.2 Coordenadas de Textura

O cone é dividido em *slices* (fatias verticais) e *stacks* (fatias horizontais ao longo da altura do cone). A altura de cada fatia horizontal (*stack_height*) é calculada como $\text{stack_height} = \frac{\text{height}}{\text{stacks}}$. Para cada fatia (slice), os pontos na base são calculados: p_x e p_z são as coordenadas dos pontos no perímetro da base, variando o ângulo α de 0 a 2π . As coordenadas de textura para a base são todas mapeadas para o ponto central da textura, simplificando para (0,0). Os pontos são calculados para cada fatia vertical, ajustando o raio conforme a altura, usando: $\text{radius} - \frac{\text{radius}}{\text{stacks}} * i$ para cada stack i .

Assim, as coordenadas de textura são calculadas como frações da posição atual em relação ao número total de *slices* e *stacks*:

- $t1 = \frac{j}{\text{slices}}$
- $t2 = \frac{j+1}{\text{slices}}$
- $t3 = \frac{i}{\text{stacks}}$
- $t4 = \frac{i+1}{\text{stacks}}$

2.5 Box

2.5.1 Normais

Para cada face do cubo, as normais são calculadas para apontar diretamente para fora da superfície, garantindo iluminação correta:

- **Face superior** (XZ no +Y): Normais (0,1,0)
- **Face inferior** (XZ no -Y): Normais (0,1,0)
- **Face frontal** (YZ no +X): Normais (0,0,1)
- **Face traseira** (YZ no -X): Normais (0,0,-1)
- **Face esquerda** (XY no +Z): Normais (-1,0,0)
- **Face direita** (XY no -Z): Normais (1,0,0)

2.5.2 Coordenadas de Textura

A caixa é dividida em div subdivisões ao longo de cada dimensão (largura e altura), criando uma grade de pequenos quadrados em cada face. Cada quadrado na grade tem um tamanho calculado como $square_size = size/div$, onde $size$ é o comprimento total da caixa. Para cada subdivisão identificada pelos índices i (horizontal) e j (vertical), as coordenadas de textura são calculadas como:

- $t1 = \frac{i}{div}$
- $t2 = \frac{i+1}{div}$
- $t3 = \frac{j}{div}$
- $t4 = \frac{j+1}{div}$

A caixa tem seis faces: superior, inferior, frente, trás, esquerda e direita. Cada face é mapeada individualmente, podendo haver variações apenas na orientação.

2.6 Patches

2.6.1 Normais

Para calcular a normal de um triângulo formado pelos pontos $p1$, $p2$ e $p3$, primeiro calculam-se os vetores u e v como:

- $u = p2 - p1$
- $v = p3 - p1$

O vetor normal n é obtido pelo produto vetorial dos vetores u e v : $n = u * v$, onde o produto vetorial é dado por:

- $n_x = u_y \cdot v_z - u_z \cdot v_y$
- $n_y = u_z \cdot v_x - u_x \cdot v_z$
- $n_z = u_x \cdot v_y - u_y \cdot v_x$

O vetor normal n é então normalizado para ter um comprimento unitário, calculando-se a sua magnitude e dividindo cada componente pela magnitude, $magnitude = \sqrt{n_x^2 + n_y^2 + n_z^2}$ e $normalizado = \left(\frac{n_x}{magnitude}, \frac{n_y}{magnitude}, \frac{n_z}{magnitude} \right)$.

2.6.2 Coordenadas de Textura

Cada $patch$ é dividido em $tesselation$ subdivisões tanto ao longo da dimensão u quanto da dimensão v , criando uma grade de pequenos quadrados. Para cada subdivisão identificada pelos índices i e j , as coordenadas de textura são calculadas como:

- $t_u1 = \frac{i}{tesselation}$
- $t_u2 = \frac{i+1}{tesselation}$

- $t_v1 = \frac{j}{tesselation}$
- $t_v2 = \frac{j+1}{tesselation}$

Para cada subdivisão do *patch*, os pontos são calculados e as coordenadas de textura são atribuídas aos vértices dos triângulos que formam cada subdivisão.

Capítulo 3

Engine

3.1 Texturas

A implementação das texturas no código fornecido começa com a inicialização do modelo, onde as coordenadas dos vértices, as normais e as coordenadas de textura são extraídas de um conjunto de dados e armazenadas em *buffers* apropriados para serem usados pela *OpenGL*. Durante a construção do modelo, as cores e propriedades do material são configuradas e, se um ficheiro de textura for especificado, os dados da textura são carregados utilizando a biblioteca *DevIL*. A textura é configurada para ser repetida tanto horizontal quanto verticalmente, e são definidos os filtros de magnificação e minimização para a textura, garantindo que a imagem da textura seja aplicada de forma suave e correta.

Ao carregar a textura, a imagem é carregada do ficheiro, convertida para o formato *RGBA* e transferida para a memória da *GPU* utilizando `glTexImage2D`. A geração de *mipmaps* é ativada para melhorar a qualidade visual da textura quando vista de diferentes distâncias, e são verificadas falhas no carregamento da textura para assegurar que a textura foi aplicada corretamente.

No método *draw*, as propriedades do material são configuradas e os *buffers* de vértices, normais e coordenadas de textura são ligados e habilitados. Se o modelo tiver uma textura associada, a textura é ligada e as coordenadas de textura são apontadas para a *GPU*. A função `glDrawArrays` é utilizada para desenhar os triângulos do modelo, garantindo que as texturas e as normais são aplicadas corretamente a cada vértice.

Após o desenho, os estados do cliente são desabilitados e a textura é desassociada para evitar efeitos indesejados em desenhos subsequentes. Este processo assegura que o modelo é desenhado com texturas aplicadas de forma correta e eficiente, utilizando a capacidade da *GPU* para renderizar gráficos complexos com alta performance.

3.2 Iluminação

Para implementar e aplicar a iluminação numa cena gerada pelo *Engine*, foi necessário caracterizar tanto a luz que ilumina todo o espaço quanto a iluminação individual dos objetos desenhados.

Quanto aos tipos de luzes, consideramos três: Luzes Posicionais, Direcionais e de Foco (*Spotlight*).

Para as luzes posicionais, bastou fornecer sua posição. Para as luzes direcionais, fornecemos seu vetor de direção correspondente.

Adicionalmente, foi introduzida uma nova classe denominada *Light*, responsável por armazenar as informações das luzes utilizadas no programa, incluindo aquelas mencionadas anteriormente.

No que diz respeito à iluminação dos objetos, foram considerados cinco tipos: difusa, especular, ambiente, emissiva e um quinto tipo, *shininess*. Para definir esses tipos, utilizamos a função *glMaterialfv*, na qual atribuímos a cor correspondente utilizando os parâmetros *GL_DIFFUSE*, *GL_AMBIENT*, *GL_SPECULAR* e *GL_EMISSIVE*. O valor de *shininess* foi definido através da função *glMaterialf*, atribuindo-o ao parâmetro *GL_SHININESS*.

3.3 Correções Fase 3

Visando aumentar a eficiência da aplicação, os VBO's implementados na fase anterior foram rescritos, tento em especial atenção aos VBO's implementados nas curvas utilizadas para as animações das primitivas.

Capítulo 4

Demo Scene

Na presente fase, foi solicitado uma *demo scene* do sistema solar animado com textura e iluminação.

Com o intuito de tornar a *demo scene* o mais "realista" possível, as curvas de Catmull-Rom utilizadas para a translação dos planetas em torno do sol foram recalculadas, tendo agora uma forma elíptica. Além disso, foi corrigido a sua inclinação axial e retificado os seus tamanhos e tempo total de translação em volta do sol.

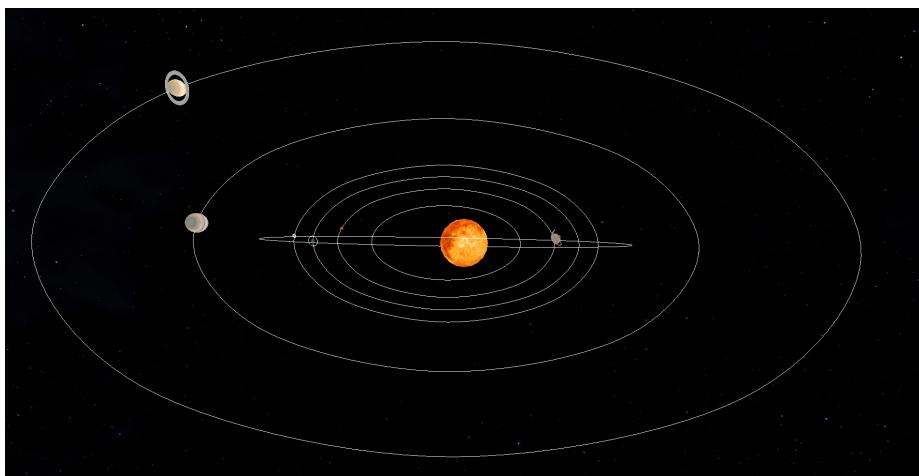


Figura 4.1: Novas curvas de *Catmull-Rom* utilizadas para a translação dos planetas.



Figura 4.2: Sistema Solar com iluminação.



Figura 4.3: Sistema Solar sem iluminação.

4.1 Textura Mercúrio



Figura 4.4: Textura do Planeta Mercúrio.

4.2 Textura Vénus



Figura 4.5: Textura do Planeta Vénus.

4.3 Textura Terra e Lua



Figura 4.6: Textura do Planeta Terra e Lua.

4.4 Textura Marte e respetivas Luas



Figura 4.7: Textura do Planeta Marte e respetivas Luas.

4.5 Textura Júpiter

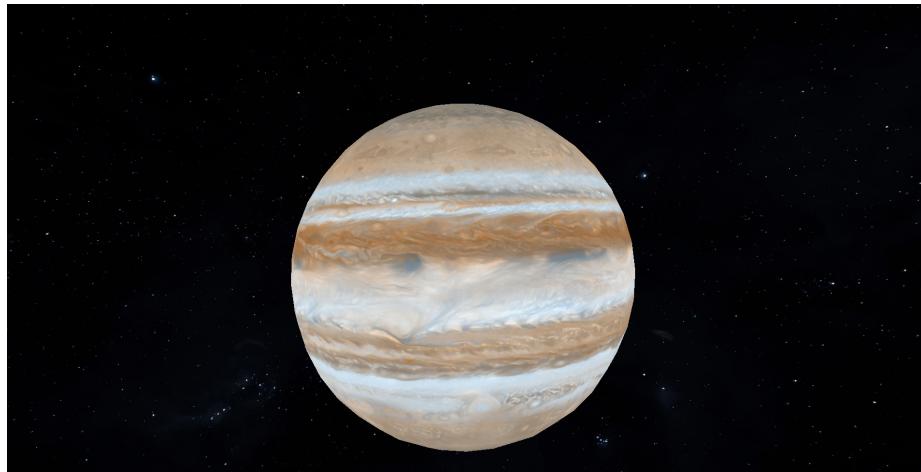


Figura 4.8: Textura do Planeta Júpiter.

4.6 Textura Saturno

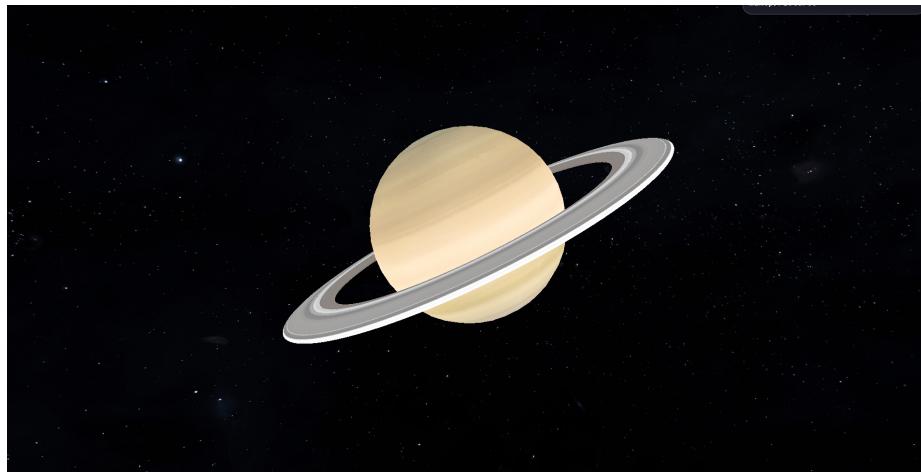


Figura 4.9: Textura do Planeta Saturno e Anel.

4.7 Textura Urano

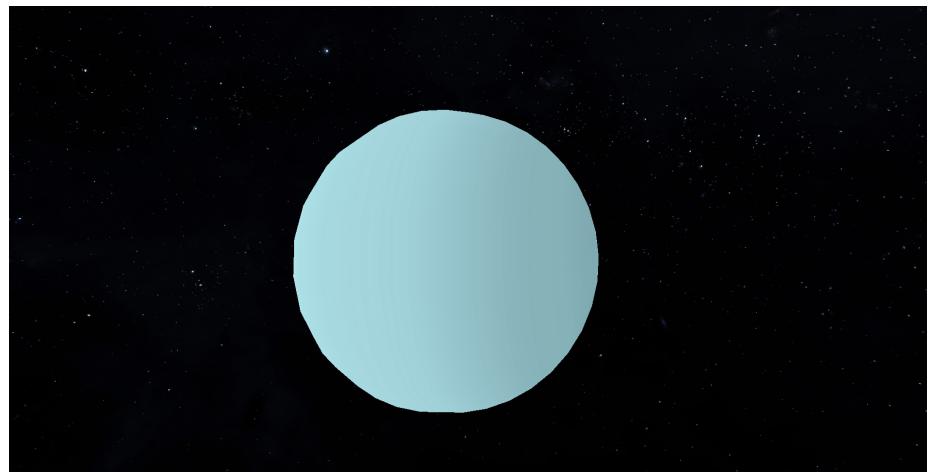


Figura 4.10: Textura do Planeta Urano.

4.8 Textura Neptuno



Figura 4.11: Textura do Planeta Neptuno.

Capítulo 5

Conclusão

A conclusão desta fase do projeto possibilitou a consolidação dos conhecimentos adquiridos durante as aulas sobre texturas e iluminação.

Desta forma, conseguimos definir e posicionar fontes de iluminação no sistema solar previamente modelado e aplicar texturas nos planetas e demais corpos celestes.

Acreditamos ter alcançado os objetivos propostos para este trabalho. Apesar das dificuldades encontradas ao longo do processo, consideramos que o resultado é satisfatório.