

Universidade do Minho

Normais e Coordenadas de Texturas
Unidade Curricular de Computação Gráfica
Licenciatura em Ciências da Computação
Universidade do Minho

Bruno Jardim
(A91680)

Inês Presa
(A90355)

Tiago Carriço
(A91695)

Tiago Leite
(A91693)

5 de junho de 2022

Índice

1	Contextualização	2
1.1	Enunciado	2
2	Apresentação das soluções	3
2.1	Normais e coordenadas de texturas dos modelos	3
2.1.1	Plano	3
2.1.2	Cubo	3
2.1.3	Esfera	3
2.1.4	Cone	4
2.1.5	Cilindro	5
2.1.6	Torus	5
2.1.7	Bezier	5
2.2	Iluminação	7
2.2.1	Demonstração	8
2.3	Texturas	10
2.3.1	Demonstração	10
2.4	Sistema solar com iluminação e texturas	11
2.4.1	Demonstração	11
2.5	Extras	12
2.5.1	Demonstração	12
3	Conclusão	13

Capítulo 1

Contextualização

No âmbito da unidade curricular de Computação Gráfica da Licenciatura em Ciências da Computação foi proposto o desenvolvimento em *OpenGL* de um motor gráfico genérico que terá como função a criação de um sistema solar. Desenvolvimento esse que deve ser composto por quatro etapas.

1.1 Enunciado

Nesta quarta etapa foi proposto:

- **Normais e coordenadas de texturas dos modelos**

Alteração do *generator* para produzir as normais dos modelos gerados.

Alteração do *generator* para produzir as coordenadas de textura dos modelos gerados.

- **Iluminação**

Alteração do *engine* para aplicar a iluminação definida no ficheiro de configuração *xml*.

- **Texturas**

Alteração do *engine* para aplicar as texturas definidas no ficheiro de configuração *xml*.

- **Sistema solar com iluminação e texturas**

Alteração do ficheiro de configuração *xml* de modo a gerar um modelo do sistema solar com iluminação e texturas.

Capítulo 2

Apresentação das soluções

2.1 Normais e coordenadas de texturas dos modelos

2.1.1 Plano

As normais do plano são $(0,1,0)$ para todos os seus pontos.

As coordenadas de textura do plano são calculadas de acordo com a variação de x e z nos dois ciclos de geração dos vértices. Ou seja, se o plano tiver d divisões, as coordenadas do vértice $(x, 0, z)$ serão associadas às coordenadas de textura $(x/d, y/d)$.

2.1.2 Cubo

As normais do cubo são as seguintes:

- Topo : $(0,1,0)$
- Base : $(0,-1,0)$
- Frente : $(0,0,1)$
- Trás : $(0,0,-1)$
- Direita : $(1,0,0)$
- Esquerda : $(-1,0,0)$

Para o cálculo das coordenadas de textura do cubo, foi utilizada a mesma estratégia do plano para cada uma das suas faces.

2.1.3 Esfera

As normais, n , da esfera são calculadas com as coordenadas polares dos vértices sem o uso do valor do raio, da seguinte forma:

- $x = \cos(\beta) * \sin(\alpha)$
- $y = \sin(\beta)$
- $z = \cos(\beta) * \cos(\alpha)$
- $n = (x, y, z)$

Sendo que, o β varia entre -90° e 90° e o α varia entre 0° e 360° .

As coordenadas de textura da esfera são calculadas utilizando as variáveis de iteração (i, j) nos dois ciclos de cálculo dos valores dos vértices de cada triângulo da esfera. Por exemplo, se i iterar pelas *slices* e j iterar pelas *stacks*, as coordenadas de textura do vértice P serão constituídas pelos seguintes pontos:

- $x = raio * \cos(90^\circ - \beta' * j) * \sin(\alpha' * i)$
- $y = raio * \sin(90^\circ - \beta' * j)$
- $z = raio * \cos(90^\circ - \beta' * j) * \cos(\alpha' * i)$

Com $\alpha' = 360^\circ / slices$ e $\beta' = 180^\circ / stacks$.

As coordenadas de texturas de P serão $(i/slices, j/stacks)$.

2.1.4 Cone

Para o cone, à semelhança do que acontece com o cálculo dos vértices de cada triângulo, o processo de cálculo das normais e coordenadas de textura foi dividido em base e corpo.

Para a base as normais são $(0, -1, 0)$ para todos os seus pontos. Para as coordenadas de textura, o ponto central tem as coordenadas $(0.5, 0.5)$ e os restantes são calculados com os valores das coordenadas de cada vértice sem o uso do valor do raio da seguinte forma:

- $x = 0.5 + \sin(\alpha)$
- $y = 0$
- $z = 0.5 + \cos(\alpha)$

Com α a variar entre 0° e 360° .

Para o corpo do cone, se se considerar i como a variável que itera pelo número de *stacks* e j como a variável que itera pelo número de *slices* e que cada ponto P tem as seguintes coordenadas:

- $x = r * \sin(\alpha * j)$
- $y = i * stack_size$
- $z = r * \cos(\alpha * j)$

Com:

- $stack_size = height / stacks$
- $\alpha = 360^\circ / slices$
- $r = (height - i * stack_size) / (height / radius)$

As normais de P têm os valores (x, y, z) , com:

- $x = \sin(\alpha * j)$
- $y = \sin(\text{atan}(radius / height))$
- $z = \cos(\alpha * j)$

As coordenadas de textura de P são $(j/slices, i/stacks)$

2.1.5 Cilindro

O cálculo das normais e coordenadas de textura do cilindro foi dividido em base, topo e corpo.

Para a base e topo utilizou-se a mesma estratégia da base do cone, com as devidas adaptações no topo, pois esta face encontra-se virada para direção oposta.

Para o corpo do cilindro, para cada ponto, os valores das normais em x e z

O cálculo das coordenadas de textura no corpo do cilindro é feito com a mesma estratégia do corpo do cone.

2.1.6 Torus

Seja P um ponto do torus em que as suas coordenadas são calculadas da seguinte forma:

- $x = (R + r * \cos(\alpha * i)) * \cos(\beta * j)$
- $y = r * \sin(\alpha * i)$
- $z = (R + r * \cos(\alpha * i)) * \sin(\beta * j)$

Com:

- i a variável que itera pelo número de *stacks*
- j a variável que itera pelo número de *slices*
- $\alpha = 360^\circ / \text{stacks}$
- $\beta = 360^\circ / \text{slices}$
- $R = (\text{raio} + \text{espessura})/2$
- $r = R - \text{espessura}$

As normais de P têm os valores (x, y, z) , com:

- $x = r * \cos(\alpha * i) * \cos(\beta * j)$
- $y = r * \sin(\alpha * i)$
- $z = r * \cos(\alpha * i) * \sin(\beta * j)$

As coordenadas de textura de P são $(i/\text{stacks}, j/\text{slices})$

2.1.7 Bezier

Relembrando que para o processo de cálculo dos pontos necessários à construção de cada superfície foi utilizada a seguinte fórmula:

$$\text{Seja a matriz de Bezier } M = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

T = nível de tesselação

4 curvas de Bezier C_0, C_1, C_2, C_3

$C_0 = P_{00}, P_{10}, P_{20}, P_{30}$

$C_1 = P_{01}, P_{11}, P_{21}, P_{31}$

$$C_2 = P_{02}, P_{12}, P_{22}, P_{32}$$

$$C_3 = P_{03}, P_{13}, P_{23}, P_{33}$$

$$B(u, v) = [u^3 \ u^2 \ u \ 1] M \begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{bmatrix} M^T \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

Os vários pontos da superfície são obtidos utilizando a função B variando u e v entre 0 e 1.

Por exemplo com:

$$a = B(0, 0) \ b = B(1/T, 0) \ c = B(1/T, 1/T) \ d = B(0, 1/T)$$

Podem-se construir os triângulos a, b, d e b, c, d .

Para calcular as normais de um ponto calcula-se:

$$u' = [3 * u^2 \ 2 * u \ 1 \ 0] M \begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{bmatrix} M^T \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

$$v' = [u^3 \ u^2 \ u \ 1] M \begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{bmatrix} M^T \begin{bmatrix} 3 * v^2 \\ 2 * v \\ 1 \\ 0 \end{bmatrix}$$

Cada normal de cada componente de P é obtida fazendo $u' \times v' / \|u' \times v'\|$.

As coordenadas de textura de P são simplesmente (v,u).

2.2 Iluminação

Para a iluminação da cena criou-se duas classes *abstracts*. A classe *Light* e a classe *Color*.

A classe *Light* é estendida pelas classes:

- *LightPoint*
- *LightDirectional*
- *LightSpotlight*

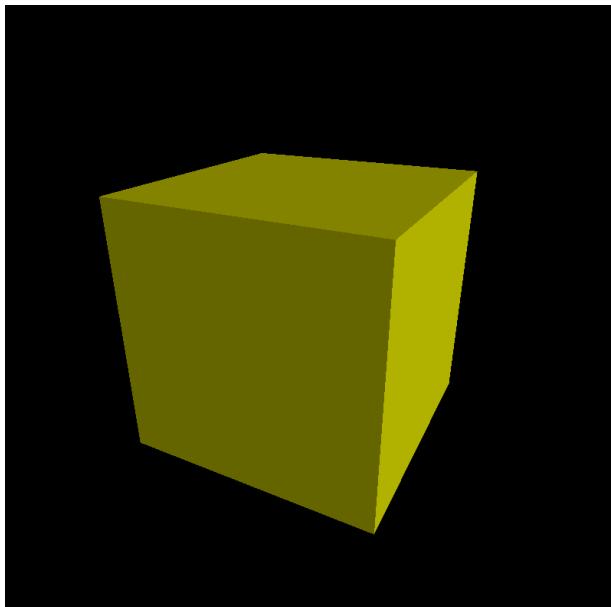
A classe *Color* é *estendida* pelas classes:

- *Diffuse*
- *Ambient*
- *Specular*
- *Emissive*
- *Shininess*

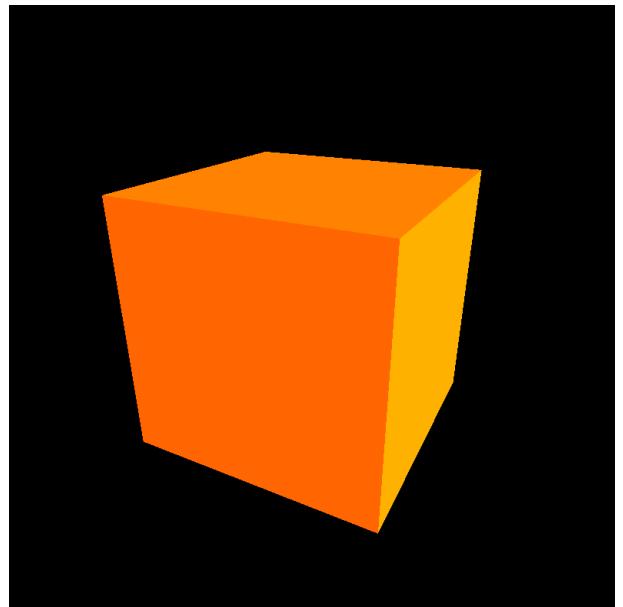
Na leitura do ficheiro *xml*, caso este possua informação de iluminação, controlo-se instâncias da classe *Light*, de acordo com o tipo de luz, e armazena-se estas num vetor. Na função *renderScene* este vetor vai ser percorrido e cada uma das "luzes" armazenadas será aplicada. Cada *model* possui agora também um vetor de *Color* onde se armazena as propriedades de iluminação dos seus materiais e a informação do *VBO* onde estão guardadas as normais de cada ponto. No momento em que os *models* são desenhados, o vetor de *Color* é percorrido e as propriedades dos materiais são aplicadas.

Nesta fase, tal como na anterior criou-se um *map* de primitivas para vetor de normais e um *map* de primitivas para *VBO* das normais, de forma a que para cada primitiva as suas normais apenas sejam lidas as normais uma vez e apenas exista um *VBO* onde estas ficam guardadas.

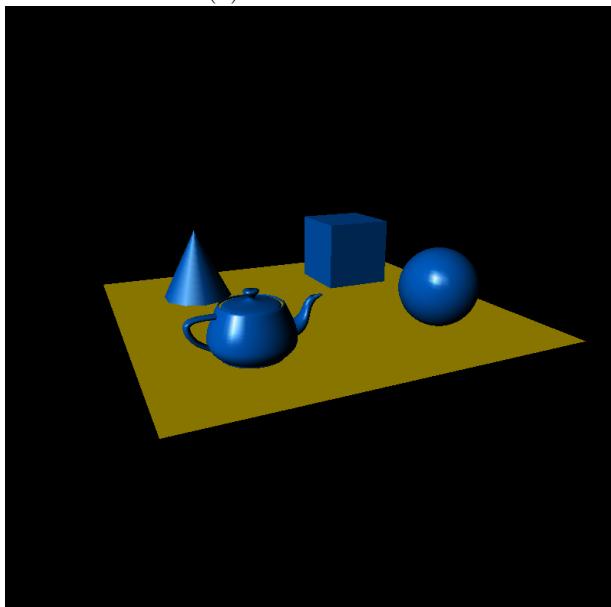
2.2.1 Demonstração



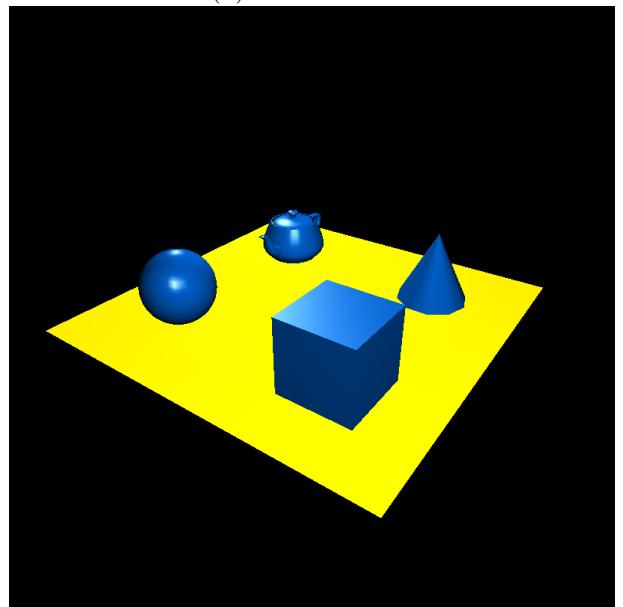
(a) test_4_1.xml



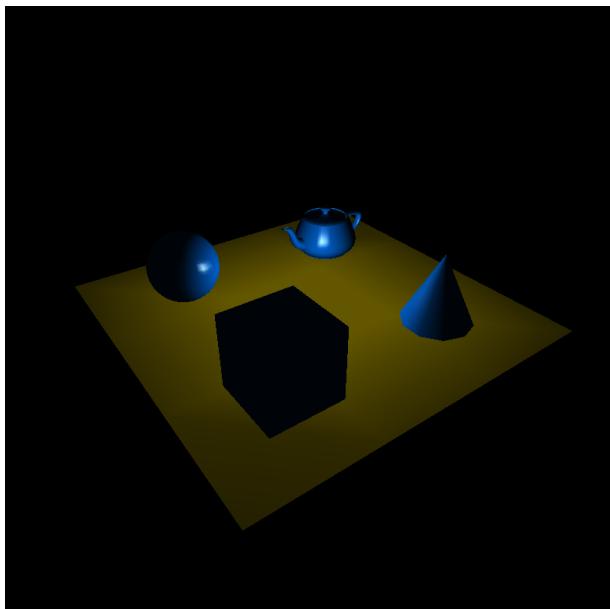
(b) test_4_2.xml



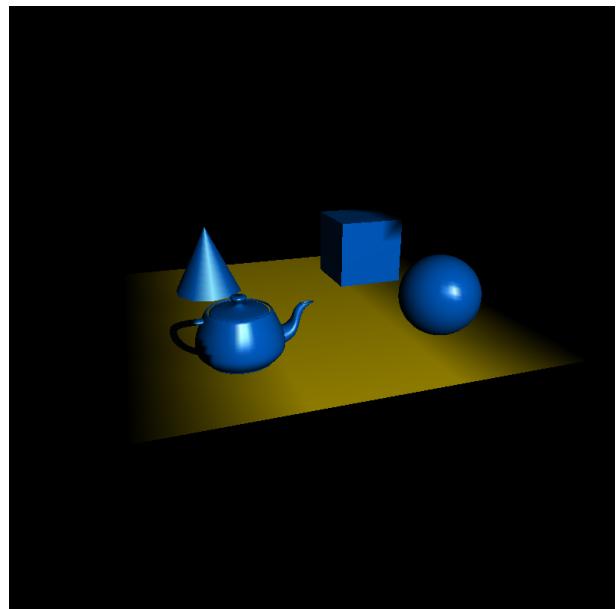
(c) test_4_3.xml



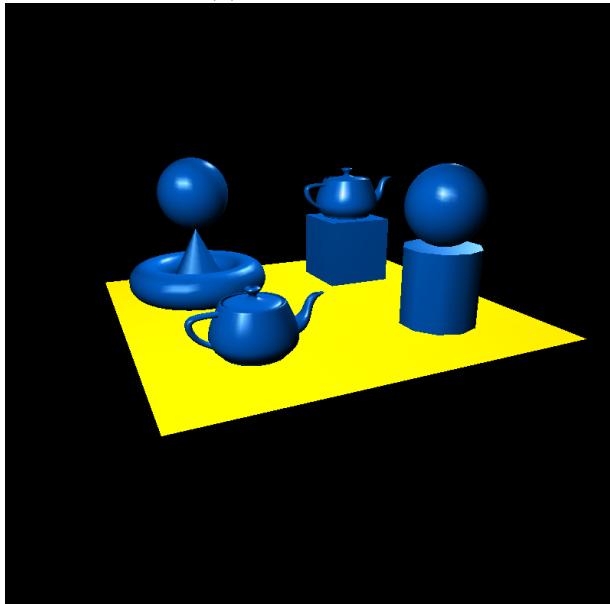
(d) test_4_4.xml



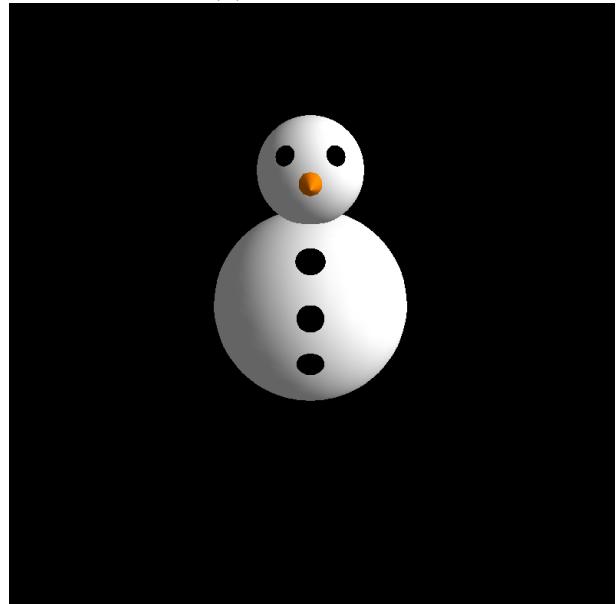
(a) `test_4_5.xml`



(b) `test_4_6.xml`



(c) `test_light.xml`



(d) `snowman.xml`

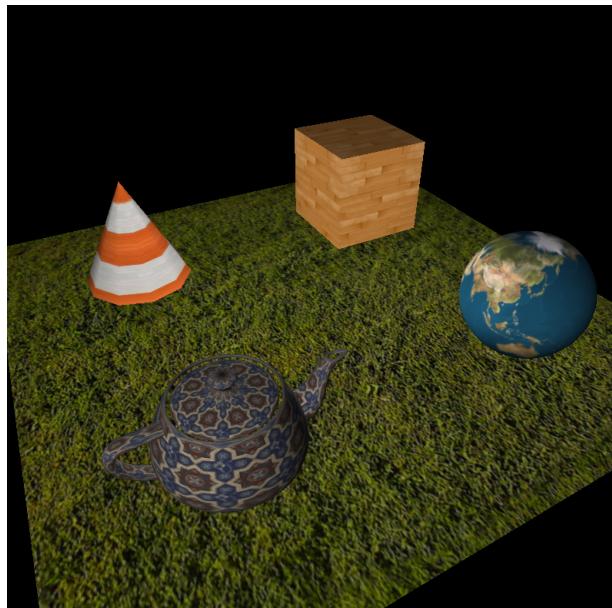
Figura 2.2: Iluminação

2.3 Texturas

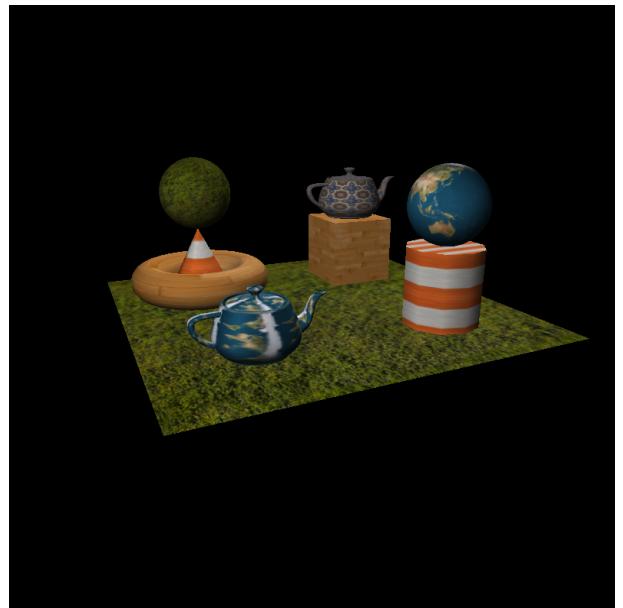
Para as texturas, foi adicionou-se para cada *model* uma variável a indicar qual o *VBO* com as coordenadas de textura e outra com a informação gerada pela função *glGenTextures*.

O *engine* analisa o ficheiro de configuração de *xml* e caso os modelos possuam texturas as suas coordenadas são lidas e armazenadas num vetor de coordenadas de textura, cada textura é carregada e a sua informação é armazenada junto do *model* respetivo. Tal como nas coordenadas de posição e na iluminação, aqui também se utiliza um *map* de primitivas para o vetor de coordenadas de textura e um mapa de apontadores de textura para a textura. Assim, cada textura é lida apenas uma vez e cada primitiva apenas vai possuir um *VBO* com as suas coordenadas de textura.

2.3.1 Demonstração



(a)



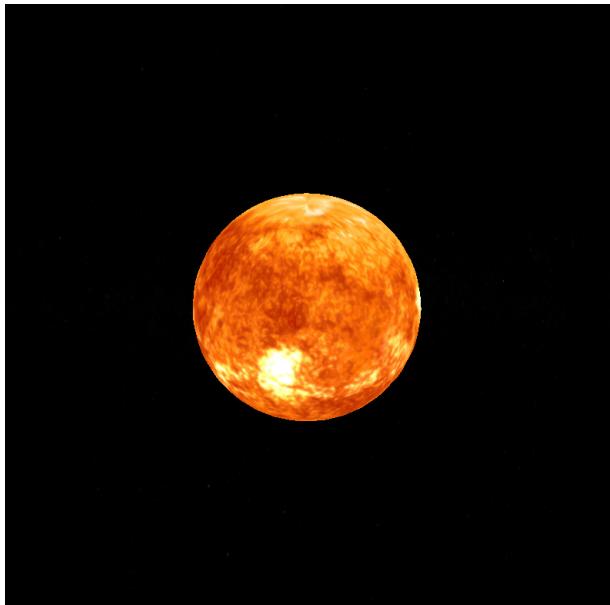
(b)

Figura 2.3: Texturas

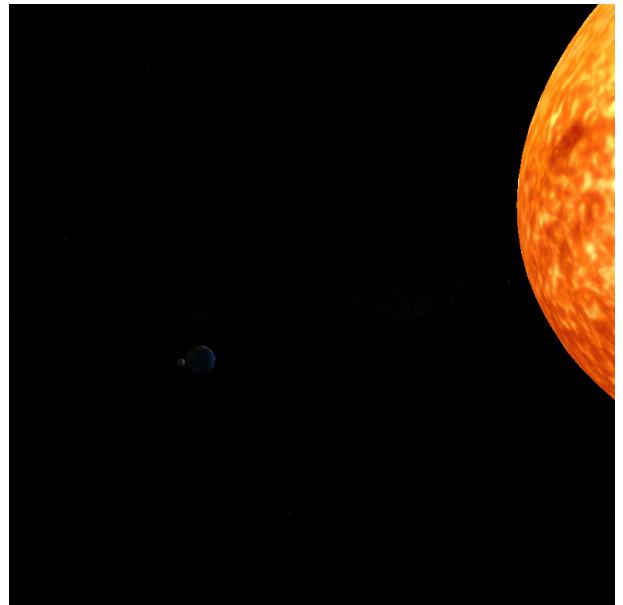
2.4 Sistema solar com iluminação e texturas

Para iluminar o sistema solar colocou-se um ponto de luz no ponto (0,0,0) e definiu-se o sol como um objeto emissivo. Colocou-se texturas a todos os objetos do sistema solar.

2.4.1 Demonstração



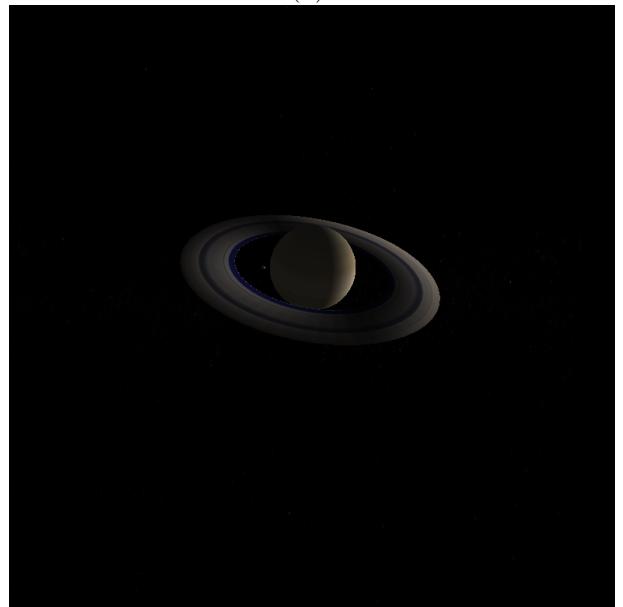
(a)



(b)



(c)



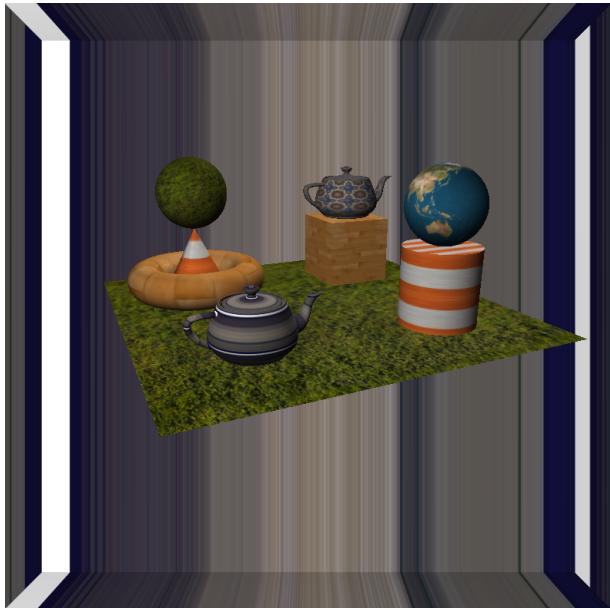
(d)

Figura 2.4: Texturas

2.5 Extras

Como extra, alterou-se o *engine* por forma a colocar uma textura de fundo. Para tal, é desenhado um cubo invertido antes de colocar a câmara. O *xml* passa também a ter o campo *background* onde se define a textura de fundo da cena.

2.5.1 Demonstração



(a)



(b)

Figura 2.5: Textura de fundo

Capítulo 3

Conclusão

Esta fase do projeto foi a que requereu mais alterações tanto no engine como no generator, no entanto, devido ás decisões tomadas nas fases anteriores que tornaram os dois programas escaláveis, essas alterações tornaram-se simples de fazer.

A principal dificuldade foi encontrar as estratégias certas para o cálculo das normais e coordenadas de textura das primitivas.

O grupo considera que todos os objetivos propostos foram atingidos.