Universidade do Minho
Escola de Ciências

Licenciatura em Ciências da Computação

Computação Gráfica

Trabalho Prático - Fase 4

Ana Beatriz Silva (a91678) Paulo Jorge Freitas (100053)

26 de maio de 2024

Conteúdo

Capítulo 1

Introdução

O seguinte relatório vai abordar a resolucão da última fase do projeto proposto na Unidade Curricular de Computação Gráfica. Nesta ultima fase foi proposta a alteração do *generator* de forma a implementar coordenadas de textura e normais para cada vértice; e também a alteração do *engine* para aplicar a iluminação e texturas definidas no ficheiro xml.

Capítulo 2

Resolução

2.1 Cálculo de Normais

2.1.1 Plano

No Plano, as normais são (0,1,0) para todos os seus pontos.

2.1.2 Cubo

No Cubo vamos ter as seguintes normais para os pontos pertencentes à face:

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

2.1.3 Esfera

Na Esfera, usamos as coordenadas polares para calcular as normais:

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

Tendo em consideração que β varia de $-90°$ a $90°$, e α entre $0°$ e $360°$.

2.1.4 Cone

No Cone, dividimos o calculo das normais em duas partes, base e corpo do cone para tornar o processo mais simples.

Para a base, as normais são $(0, -1, 0)$ para todos os seus pontos.

Para o corpo do cone, vamos considerar i a variável que itera as stacks, e j a variável que itera as slices, logo cada pontos de P vai ter as seguintes coordenadas:

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

Considerando $r = (height - i * (height/stacks)/(height/radius))$ e $\alpha = 360°/slices$.

Assim, as normais de P têm os valores (x, y, z), com $(sin(\alpha * j), sin(atan(radius/height)), cos(\alpha * j))$.

2.1.5 Torus

Seja P um ponto do torus, vamos calcular as suas coordenadas da seguinte forma:

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

Admitindo que i é a variável que itera o número de *stacks*, j a variável que itera o número de *slices*, $\alpha = 360°/stacks$, $\beta = 360°/slices$, $R = (raio + espessura)/2$, e $r = R - espessura$.

Assim, as normais de P têm os valores (x, y, z), com $(r * cos(\alpha * i) * cos(\beta * j), r * sin(\alpha * i), r * cos(\alpha * i) * sin(\beta * j))$

2.1.6 Bezier

Considerando 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

Temos 4 curvas de Bezier $C0$, $C1$, $C2$, $C3$:

$C0 = P_{00}, P_{10}, P_{20}, P_{30}$, $C1 = P_{01}, P_{11}, P_{21}, P_{31}$, $C2 = P_{02}, P_{12}, P_{22}, P_{32}$, e $C3 = P_{03}, P_{13}, P_{23}, P_{33}$

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

Obtivemos os pontos da superfície com a função B, ao variar u e v entre 0 e 1.

Para calcular as normais de um ponto temos:

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

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

Obtemos as normais de cada componente P ao fazer $u*v$ / $||u' * v'||$

2.2 Iluminação

Para a implementação da luz foram criadas duas classes abstratas, uma para a luz, e outra para a cor. A classe *Light* é extendida pelas classes: *LightPoint*, *LightDirectional*, e *LightSpotlight*.

A classe Color é acrescentada pelas classes: *Diffuse*, *Ambient*, *Specular*, *Emissive*, e *Shininess*.

Na leitura do ficheiro *xml*, são construidas classes de instância *Light* de acordo com o tipo de luz. Após isto, as luzes serão aplicadas a partir da função *renderScene*. No momento em que os *models* são desenhados, as informações armazenadas em *Color* são também aplicadas de forma a obter os resultados que podemos ver em baixo.

Apenas a luz emissiva e ambiente funciona corretamente.

2.3 Resultados

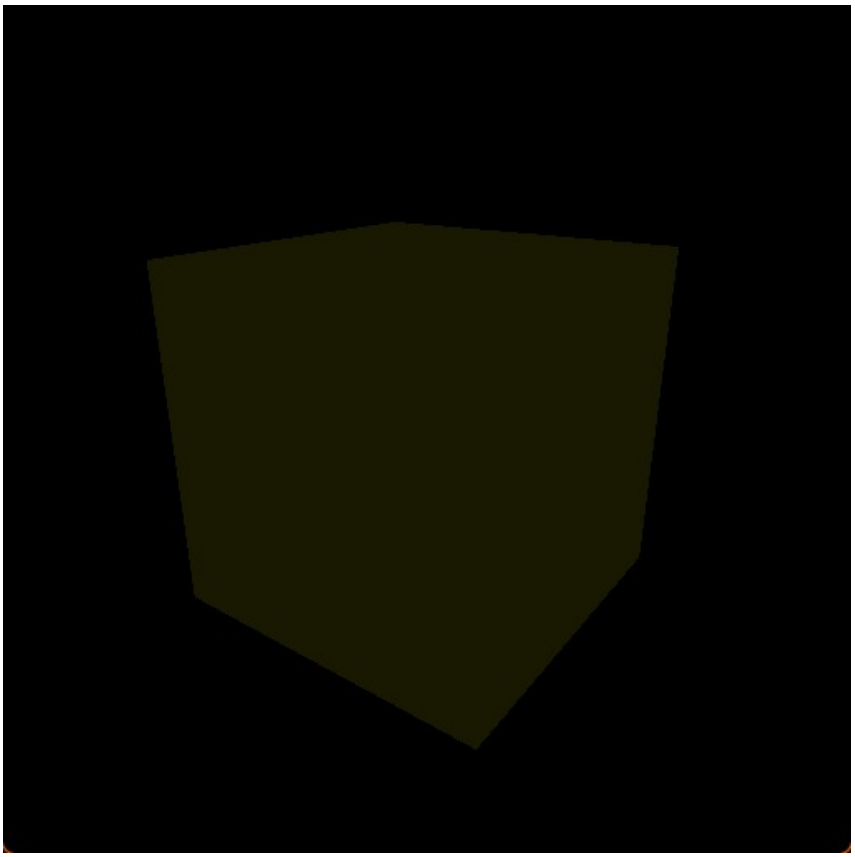

Figura 2.1: test_4_1.xml

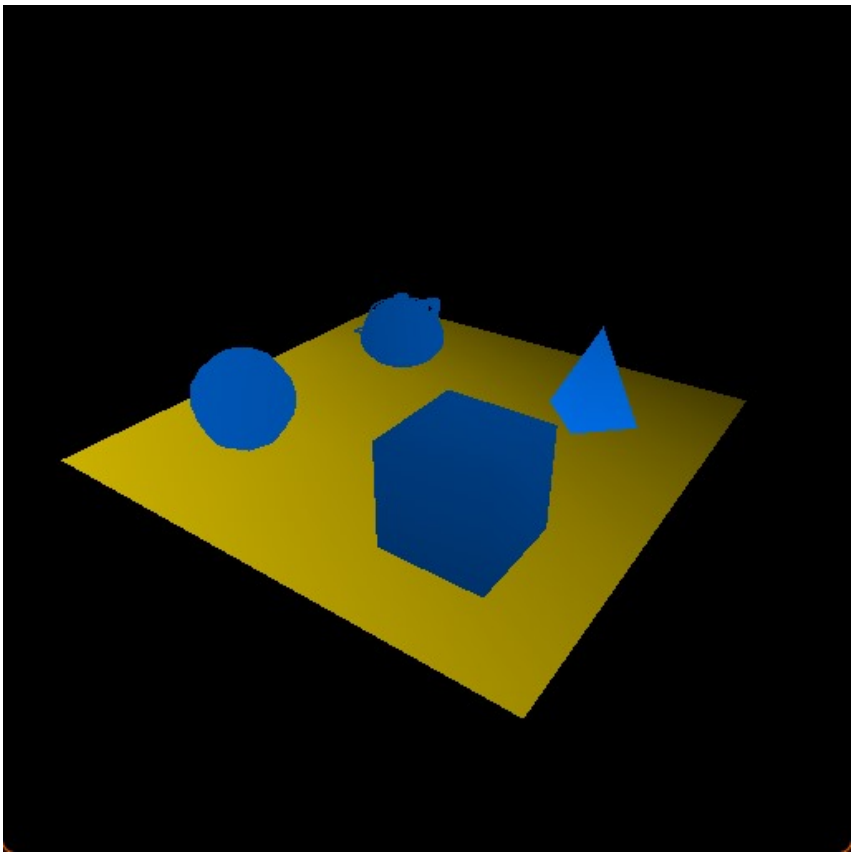

Figura 2.2: test_4_4.xml

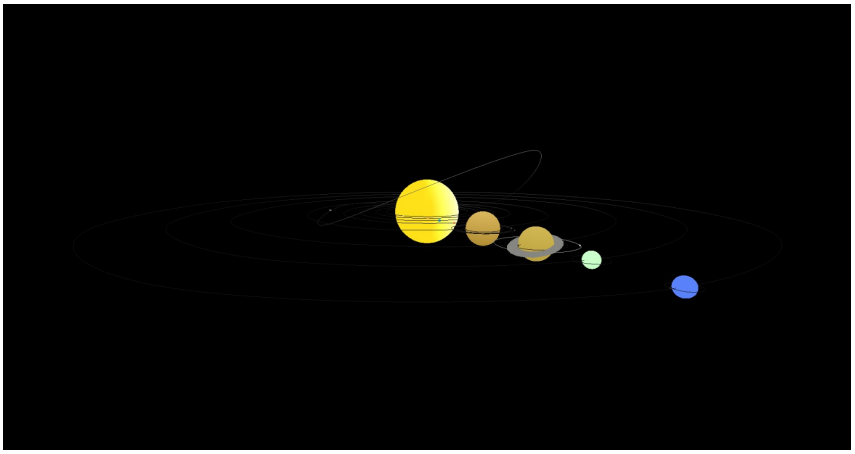

Figura 2.3: Sistema Solar

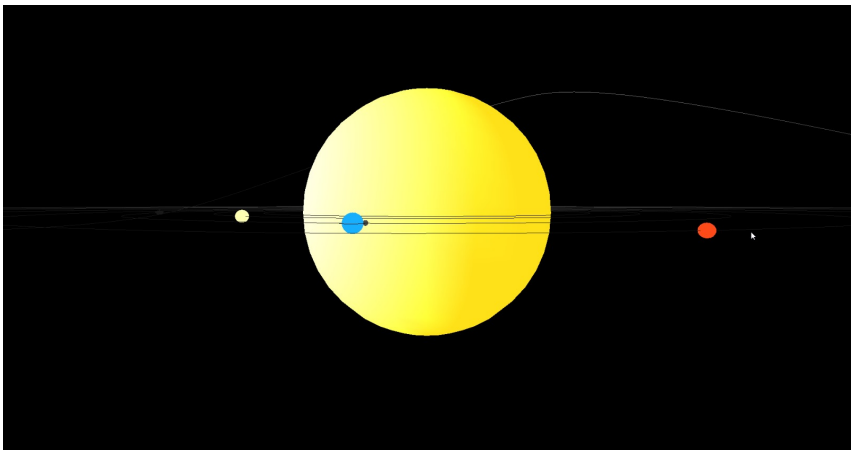

Figura 2.4: Sistema Solar

Capítulo 3

Conclusão

Como se pode perceber pelo presente relatório, não fomos capazes de implementar a parte do projeto relativo a texturas.

No entanto, apesar disto fizemos o nosso melhor para implementar a iluminação da melhor forma possível e pensamos que este objetivo foi alcançado, apesar das dificuldades que sentimos no cálculo das normais dos diferentes modelos.

Num futuro projeto, vamos ter em maior atenção a nossa gerência de tempo para garantir que conseguimos concretizar todas as partes do mesmo no tempo.