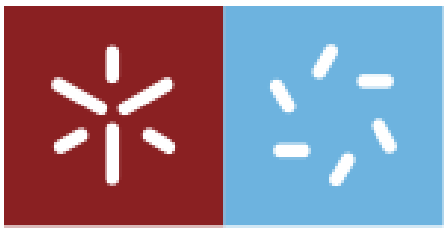

Universidade do Minho
Escola de Ciências

Computação Gráfica

Fase IV

Relatório de Desenvolvimento

André Oliveira Barbosa
A91684

Francisco António Borges Paulino
A91666

4 de junho de 2023

Resumo

Este relatório foi elaborado no âmbito da quarta fase do trabalho prático da disciplina de Computação Gráfica. Sendo esta a última fase do projeto, o grupo realizou várias alterações pertinentes, relatadas ao longo deste documento.

Conteúdo

Capítulo 1

Introdução

1.1 Enquandramento e contexto

Na quarta e última fase do trabalho prático da Unidade Curricular de Computação Gráfica foi-nos pedido fazer as seguintes alterações:

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

  Alteração do *gerador* para produzir as normais dos modelos gerados, e para produzir as cordenadas 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.

Devido a vários erros, esta fase não constrói os resultados pretendidos, nomeadamente na iluminação e nas texturas.

Capítulo 2

Concepção da Resolução

2.1 Normais e coordenadas de texturas dos modelos

2.1.1 Plano

As normais do plano são (0,1,0) em todos os 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 -90ř e 90ř e o α varia entre 0ř e 360ř.
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:

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

Com $\alpha' = 360ř/slices$ e $\beta' = 180ř/stacks$.
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 calculo 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 0ř e 360ř.
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:

- $stack\_size = height/stacks$
- $\alpha = 360ř/slices$
- $r = (height - i * stack\_size)/(height/radius)$

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

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

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

2.1.5 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*
- $\alpha = 360ř/stacks$
- $\beta = 360ř/slices$
- $R = (raio + espessura)/2$
- $r = R - espessura$

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 $(i/stacks, j/slices)$.

2.2 Iluminação

Para a iluminação da cena criamos duas classes abstratas. 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, construímos instâncias da classe *Light* de acordo com o tipo de luz, e armazenamo-as num vetor. Na função *renderScence* este vetor vai ser percorrido e cada uma das "luzes"armazenadas será aplicada. Cada *model* possui agora também um vetor de *Color* onde se aramazena as propriedades de iluminação dos seus materiais. No momento em que os *models* são desenhados, o vetor de *Color* é percorrido e as propriedades dos materiais são, assim, aplicadas.

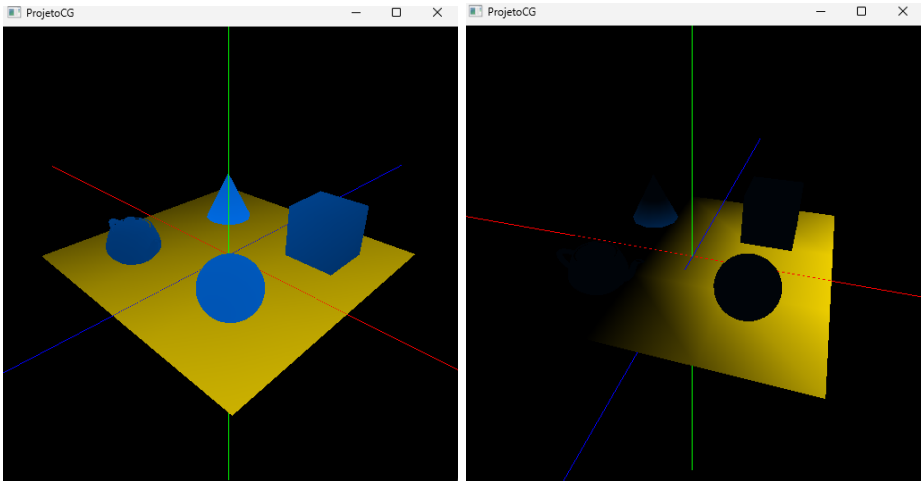

Figura 2.1: Teste 3 e Teste 4

2.3 Texturas

Para as texturas, foi adicionou-se para cada *model* uma variável a indicar as coordenadas de textura.
O *engine* analisa o ficheiro de configuração de *xml* e caso os modelos possuam texturas, as suas coordenadas são lidas e, na nossa concepção deveriam ser armazenadas num vetor de coordenadas de textura (o que não foi implementado). Cada textura deveria então ser carregada e a sua informação armazenada junto do *model* respetivo.

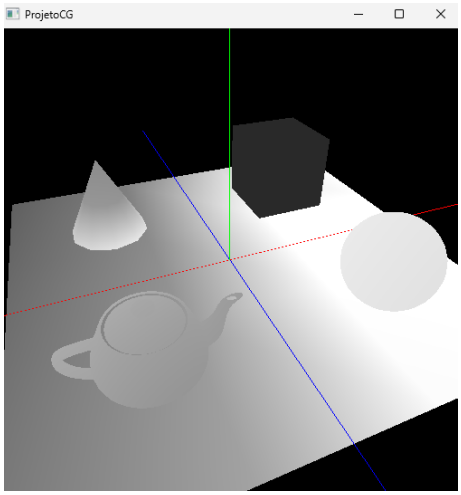

Figura 2.2: Teste 6, onde é notório que a aplicação das texturas não é bem implementada

2.4 Sistema solar com iluminação e texturas

Para iluminar o Sistema Solar e tentar aproximar o modelo da realidade, definimos o Sol como um objeto emissivo, colocando um ponto de luz no ponto (0,0,0). Foram adicionadas igualmente texturas em todos os objetos do Sistema Solar.

Capítulo 3

Conclusão

Nesta fase do projeto sentimos bastantes dificuldades que nos levaram a não concluí-lo com sucesso. Consideramos que temos um código que se aproxima do proposto, mas que acaba por não funcionar com sucesso. A luz e as texturas não funcionam como o pretendido e o modelo do Sistema Solar acaba assim por não ser gerado.