

Relatório 4^a Fase | Computação Gráfica Grupo 25 | 2024/2025

**Afonso Dionísio
(A104276)**

**João Lobo
(A104356)**

**Rafael Seara
(A104094)**

**Rita Camacho
(A104439)**

Índice

| | | |
|--------|--|----|
| 1. | Introdução | 4 |
| 2. | Normais | 5 |
| 2.1. | Normais do Plano | 5 |
| 2.2. | Normais da Esfera | 5 |
| 2.3. | Normais da Caixa | 5 |
| 2.4. | Normais do Cone | 6 |
| 2.5. | Normais do Cilindro | 6 |
| 2.6. | Normais do Torus | 7 |
| 2.7. | Normais da Icoesfera | 7 |
| 2.8. | Normais dos <i>Bezier Patches</i> | 8 |
| 2.9. | Visualização <i>debug</i> de normais | 8 |
| 3. | Luzes | 9 |
| 3.1. | Luz Direcional | 9 |
| 3.2. | Luz Foco (<i>Spotlight</i>) | 9 |
| 3.3. | Luz Pontual | 10 |
| 3.4. | Luzes Coloridas | 10 |
| 3.5. | Materiais | 11 |
| 4. | Texturas | 11 |
| 4.1. | Coordenadas de Textura | 11 |
| 4.1.1. | Coordenadas de Textura do Plano | 11 |
| 4.1.2. | Coordenadas de Textura da Esfera | 12 |
| 4.1.3. | Coordenadas de Textura da Caixa | 12 |
| 4.1.4. | Coordenadas de Textura do Cone | 13 |
| 4.1.5. | Coordenadas de Textura do Cilindro | 13 |
| 4.1.6. | Coordenadas de Textura do Torus | 14 |
| 4.1.7. | Coordenadas de Textura da Icoesfera | 14 |
| 4.1.8. | Coordenadas de Textura dos <i>Bezier Patches</i> | 15 |
| 5. | Formato final dos ficheiros .3d | 15 |
| 6. | Sistema Solar | 16 |
| 6.1. | Texturas dos planetas | 16 |
| 6.2. | Luzes | 16 |
| 6.3. | <i>Skybox</i> | 17 |
| 6.4. | Materiais | 17 |
| 7. | Janela de configurações | 18 |
| 8. | Conclusão | 19 |
| 8.1. | Trabalho Futuro | 19 |
| 8.1.1. | <i>Shaders</i> | 19 |
| 8.1.2. | <i>Frustum Culling</i> | 19 |

1. Introdução

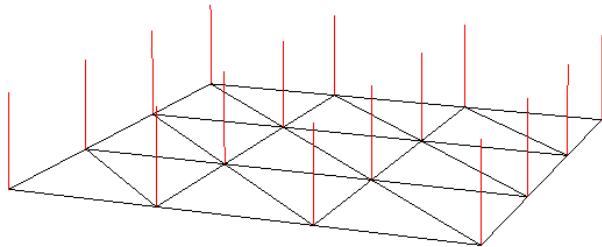
O presente relatório apresenta informações relativas à **4^a e Última Fase do Trabalho Prático** da Unidade Curricular **Computação Gráfica**, pertencente ao 2º Semestre do 3º Ano da Licenciatura em Engenharia Informática, realizada no ano letivo 2024/2025, na Universidade do Minho.

A 4^a Fase consistiu em implementar luzes, normais, materiais e texturas, assim como a versão final do Sistema Solar. Como funcionalidades adicionais, implementámos visualização de normais, luzes coloridas e uma janela de configuração do *engine*.

2. Normais

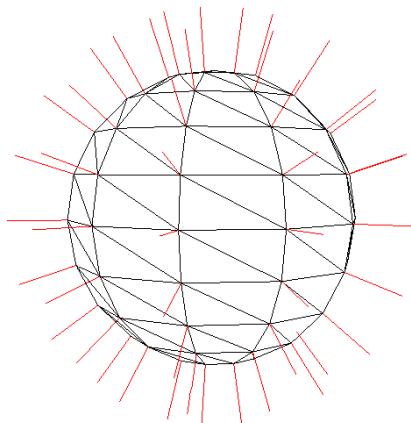
2.1. Normais do Plano

Para o plano, todos os pontos têm a mesma normal, $(0, 1, 0)$ (direção positiva no eixo y).



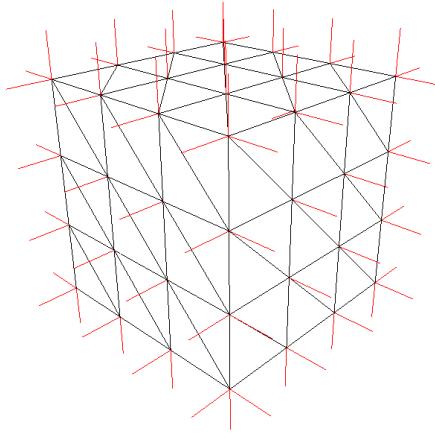
2.2. Normais da Esfera

Para calcular as normais da esfera, a partir de coordenadas esféricas, determina-se, para cada ponto da superfície, o vetor que liga o centro da esfera ao respetivo vértice.



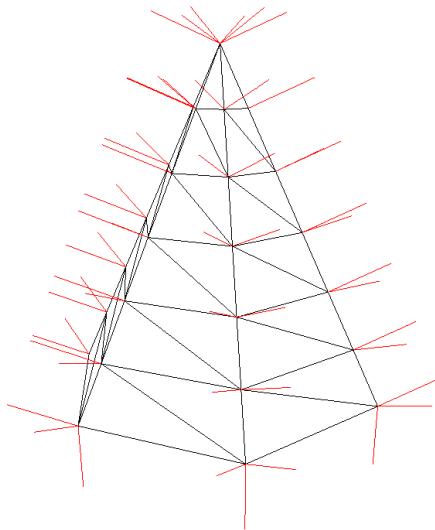
2.3. Normais da Caixa

As normais da caixa apenas dependem do lado em que estão inseridas, sendo o valor constante ao longo de cada lado. Topo: $(0, 1, 0)$, Base: $(0, -1, 0)$, Laterais: $(-1, 0, 0)$, $(1, 0, 0)$, $(0, 0, -1)$, $(0, 0, 1)$.



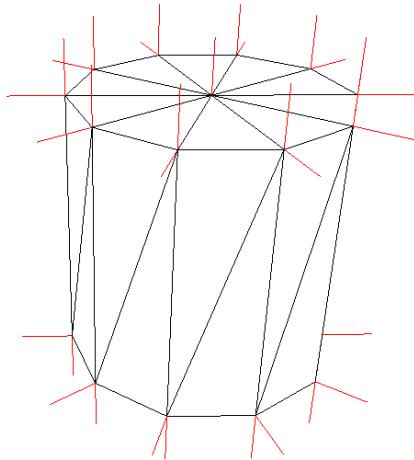
2.4. Normais do Cone

A normal das faces laterais do cone é calculada como o vetor perpendicular ao plano definido por três vértices adjacentes, usando o produto vetorial entre duas arestas do triângulo. Esse vetor é então normalizado e aplicado igualmente aos três vértices da face. Já na base, a normal é constante e aponta para baixo, no eixo Y ($0, -1, 0$).



2.5. Normais do Cilindro

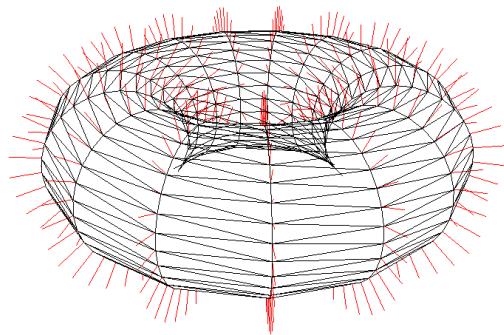
As normais das laterais do cilindro são calculadas como vetores unitários radiais no plano XZ, perpendiculares ao eixo do cilindro. Já nas tampas, as normais são constantes, apontando para cima ou para baixo no eixo Y, resultando em superfícies planas.



2.6. Normais do Torus

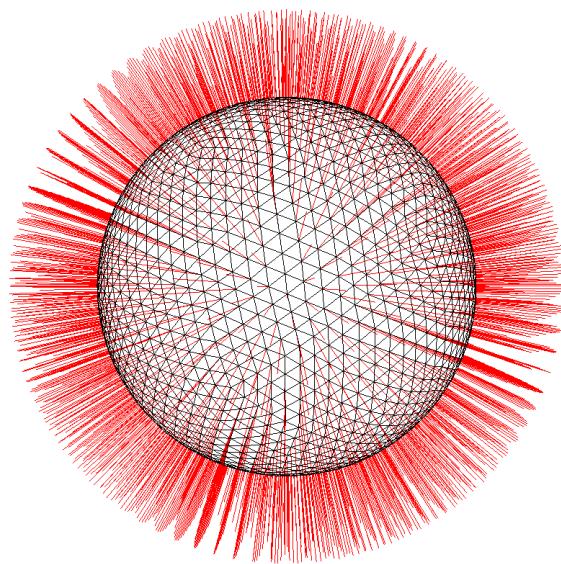
As normais do torus são calculadas subtraindo o centro do tubo (no círculo principal) da posição de cada vértice. Esse centro é obtido com raio $\star \frac{\cos(\theta)}{\sin(\theta)}$ e representa o ponto médio do anel naquele segmento.

A normal é então o vetor normalizado obtido através de vértice – centro_do_tubo, apontando radialmente para fora do tubo, perpendicular à superfície.



2.7. Normais da Icoesfera

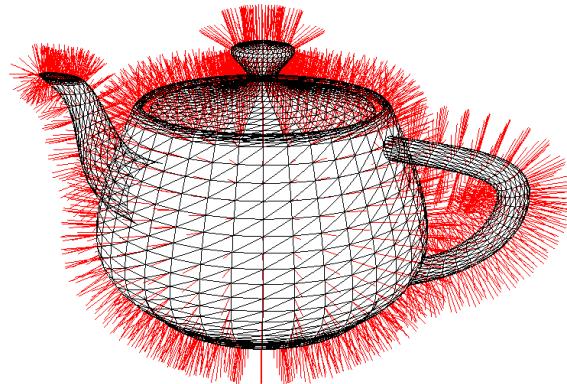
As normais da icoesfera são calculadas com base nos próprios vértices, pois todos eles já estão normalizados (apontando do centro para a superfície). Como cada ponto da icoesfera tem uma normal igual à direção radial, basta usar o vetor do vértice antes de multiplicar pelo raio. Isso garante que a normal de cada vértice seja unitária e perpendicular à superfície.



2.8. Normais dos *Bezier Patches*

As normais do modelo baseado em patches de Bezier são calculadas numericamente a partir das derivadas parciais da superfície em relação aos parâmetros u e v , que representam as coordenadas paramétricas no domínio da superfície.

A normal $N(u, v)$ é obtida pelo produto vetorial das derivadas parciais verticais de u e v .



2.9. Visualização *debug* de normais

Para nos ajudar no processo de implementação das normais dos vários modelos, desenvolvemos um modo de visualização extra que permite ver a representação das normais dos modelos em cena através de linhas. Estas linhas correspondem à ligação entre o vértice somado com o seu vetor normal.

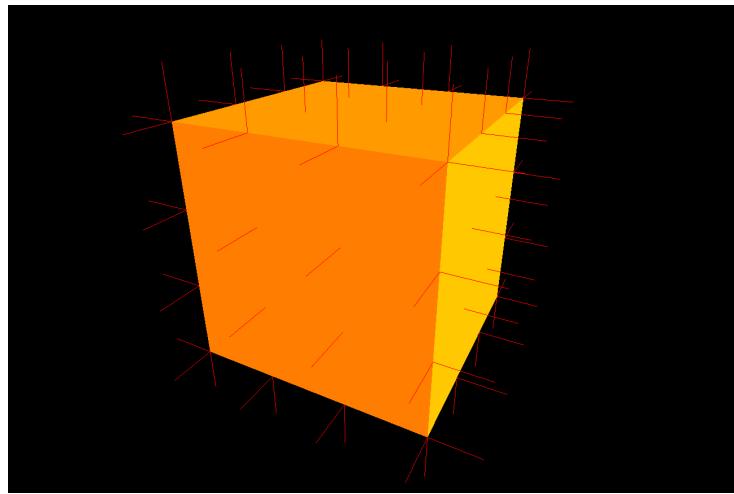


Figura 9: Visualização das normais

3. Luzes

Para as luzes, utilizámos a “*Fixed Function Pipeline*” da API do OpenGL, que permite configurar diferentes tipos de fontes de luz (direcional, pontual e foco).

3.1. Luz Direcional

A luz direcional simula uma fonte de luz que está infinitamente distante, como o Sol. A direção da luz é definida por um vetor e não possui uma posição concreta.

Esta configuração faz com que a luz afete todos os objetos na mesma direção, independentemente da sua posição no espaço.

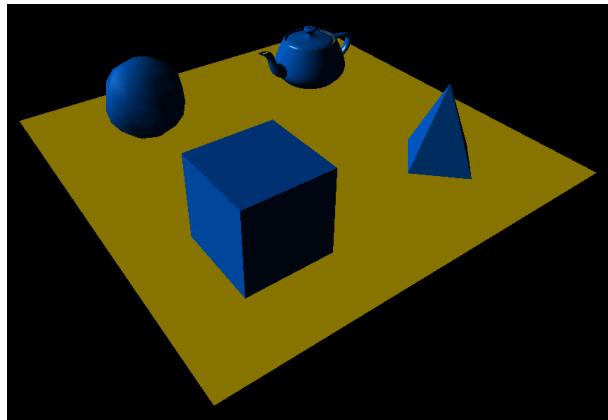


Figura 10: Luz direcional (direção direita, baixo)

3.2. Luz Foco (*Spotlight*)

A luz do tipo *spotlight* simula um foco direcional, como uma lanterna, onde a luz é emitida a partir de uma posição e numa direção específica, com um cone de abertura definido por um ângulo. No código, isto é feito ao combinar uma posição com uma direção e um valor de *cutoff*.

O parâmetro *cutoff* determina o ângulo de abertura do cone de luz. Quanto menor o valor, mais estreito o feixe de luz.

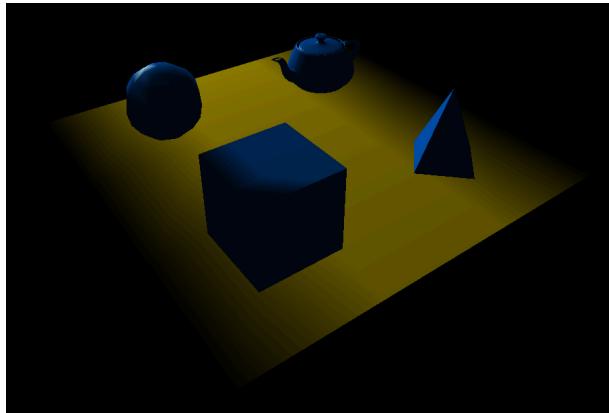


Figura 11: Luz foco

3.3. Luz Pontual

A luz pontual representa uma fonte de luz localizada num ponto específico do espaço, irradiando luz em todas as direções (como uma lâmpada).

O OpenGL, por omissão, aplica atenuação a luzes pontuais com base na distância, embora esse comportamento possa ser ajustado.

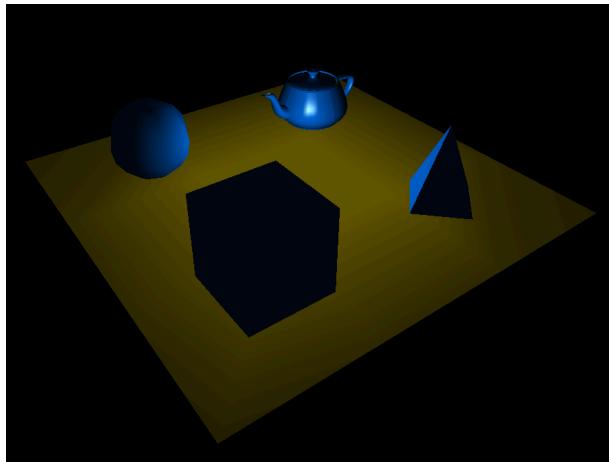


Figura 12: Luz pontual no centro da cena

3.4. Luzes Coloridas

Como extra, decidimos implementar luzes com cor. Para as utilizar, é necessário incluir uma nova tag **color** com os atributos R, G e B dentro da tag XML da **light** já existente:

```
<light type="point" posx="2" posy="2" posz="-2">
    <color R="255" G="0" B="0" />
</light>
<light type="point" posx="-2" posy="2" posz="2">
    <color R="0" G="255" B="0" />
</light>
<light type="point" posx="0" posy="2" posz="0">
    <color R="0" G="0" B="255" />
</light>
```

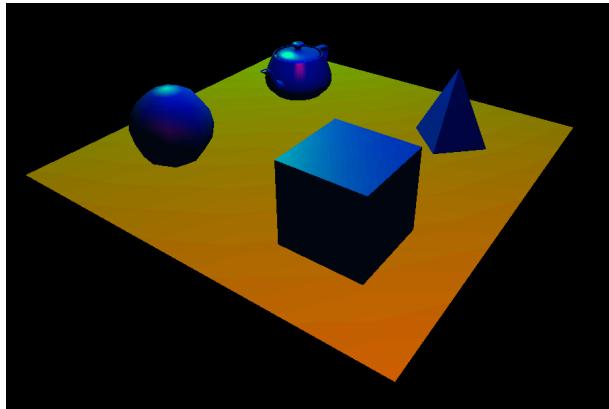


Figura 13: 3 Luzes de tipo pontual, uma vermelha, uma verde e uma azul (exemplo acima)

3.5. Materiais

Como foi implementado o sistema de iluminação, também se tornou necessário atribuir materiais aos modelos de cada grupo. Cada material é definido por quatro componentes de cor — ambiente, difusa, especular e emissiva — além de um coeficiente de brilho (*shininess*). A implementação do material foi realizada de acordo com os requisitos especificados no enunciado.

Para qualquer modelo que não contenha informação sobre o seu material, é lhe atribuído os seguintes valores *default*:

```
<diffuse R="200" G="200" B="200" />
<ambient R="50" G="50" B="50" />
<specular R="0" G="0" B="0" />
<emissive R="0" G="0" B="0" />
<shininess value="0" />
```

4. Texturas

Nesta fase, foi necessário adicionar texturas aos modelos. Sempre que, durante a leitura de um modelo, é detetada a presença de uma textura, esta é carregada para um *buffer* de textura do OpenGL, e o identificador gerado é armazenado no próprio modelo. Foi também necessário adicionar informação de como as texturas são mapeadas nos modelos gerados, as coordenadas de textura.

4.1. Coordenadas de Textura

4.1.1. Coordenadas de Textura do Plano

As coordenadas de textura do plano são geradas com base na posição dos vértices em relação ao centro do plano. Cada coordenada (x, z) é normalizada para o intervalo $[0, 1]$ ao ser deslocada por `half` e dividida por `length`, assegurando que a textura cobre todo o plano uniformemente. Para cada quadrado gerado são atribuídas quatro coordenadas de textura, correspondentes aos seus quatro vértices.

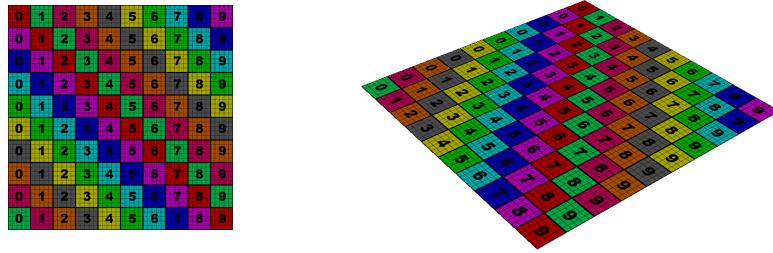


Figura 14: Demonstração de coordenadas de textura do plano

4.1.2. Coordenadas de Textura da Esfera

As coordenadas de textura da esfera são geradas com base na posição relativa de cada vértice nas divisões de `slices` e `stacks`. A coordenada U (horizontal) é dada por `slice / slices` e a coordenada V (vertical) por `stack / stacks`, o que mapeia a textura ao longo da esfera de forma contínua. Isso garante que a textura seja distribuída uniformemente da esquerda para a direita (longitude) e de baixo para cima (latitude).

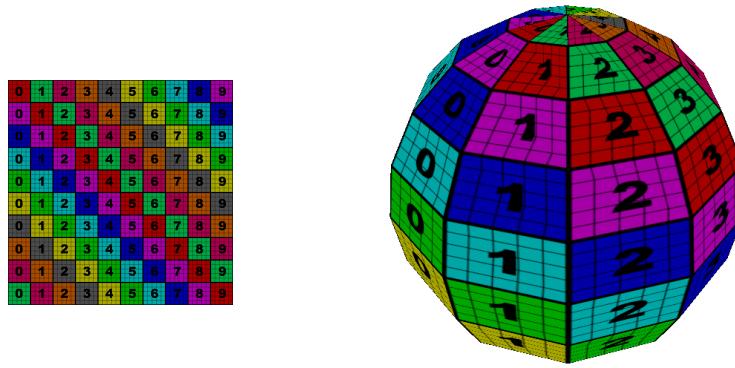


Figura 15: Demonstração de coordenadas de textura da esfera

4.1.3. Coordenadas de Textura da Caixa

As coordenadas de textura da caixa são geradas com base na posição relativa de cada subdivisão dentro de cada face, normalizadas entre 0 e 1. Para cada face, `tu` e `tv` representam a posição horizontal e vertical da célula na grelha de divisões. Cada célula da face recebe quatro coordenadas UV que variam conforme a orientação da face, garantindo o mapeamento correto da textura.

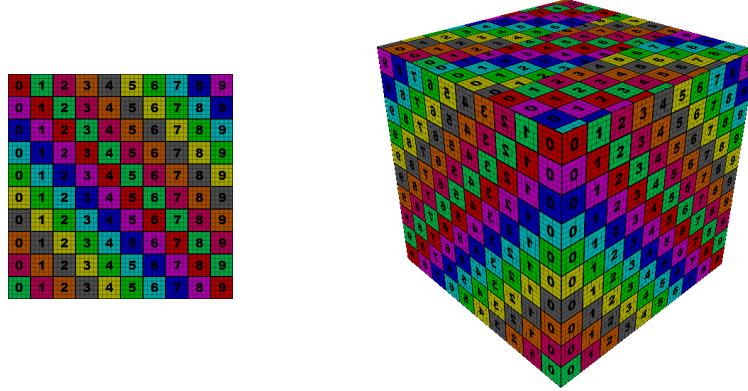


Figura 16: Demonstração de coordenadas de textura da caixa

4.1.4. Coordenadas de Textura do Cone

As coordenadas de textura do cone são geradas mapeando o parâmetro angular (`slice`) em u , variando de 0 a 1 ao redor da base, e a altura (`stack`) em v , de 0 (base) a 1 (topo). Cada fatia do cone recebe um retângulo UV correspondente ao seu segmento angular e altura, criando um mapeamento contínuo na lateral. Para a base, as coordenadas UV são mapeadas num círculo, com o centro fixo em (0.5, 0.5).

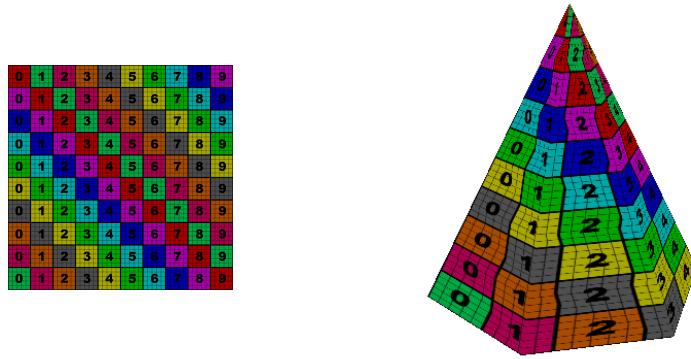


Figura 17: Demonstração de coordenadas de textura do cone

4.1.5. Coordenadas de Textura do Cilindro

As coordenadas de textura do cilindro são geradas mapeando o ângulo do `slice` em u , variando de 0 a 1 ao redor da circunferência, e a altura em v , de 0 (base) a 1 (topo), criando um retângulo UV contínuo para a lateral. Para as bases (topo e fundo), as coordenadas UV são mapeadas num círculo no plano, com o centro fixo em (0.5, 0.5) e as bordas calculadas usando funções trigonométricas para distribuir os pontos em torno do centro.

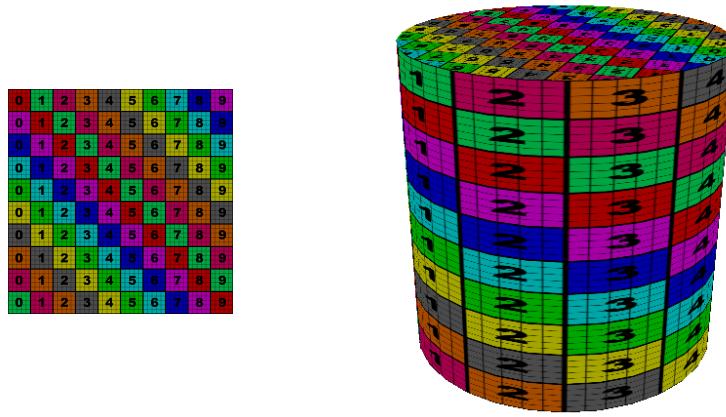


Figura 18: Demonstração de coordenadas de textura do cilindro

4.1.6. Coordenadas de Textura do Torus

As coordenadas de textura do torus são geradas mapeando linearmente os índices de fatias (`slices`) e anéis (`stacks`) no intervalo [0, 1], onde o eixo U corresponde ao ângulo ao redor do círculo principal (`stacks`) e o eixo V ao redor do tubo (`slices`). Cada vértice recebe um UV calculado como $(\frac{\text{stack}}{\text{totalStacks}}, \frac{\text{slice}}{\text{totalSlices}})$, criando um grelha regular na superfície.

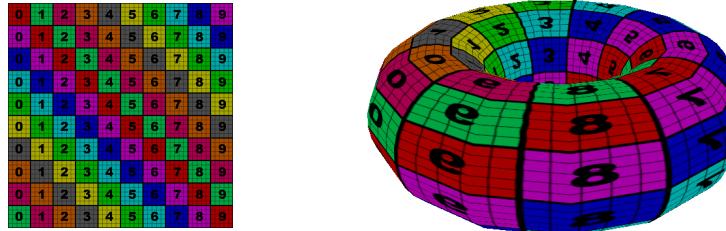


Figura 19: Demonstração de coordenadas de textura do torus

4.1.7. Coordenadas de Textura da Icosfera

As coordenadas de textura da icosfera são geradas projetando a posição normalizada de cada vértice em coordenadas esféricas, calculando u pelo ângulo azimutal (`atan2`) e v pelo ângulo polar (`acos`). O valor u é mapeado de 0 a 1 ao redor da esfera, enquanto v varia de 0 a 1 do topo ao fundo.

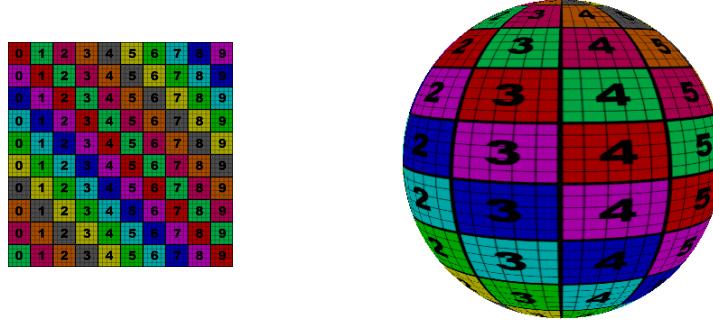


Figura 20: Demonstração de coordenadas de textura da icoesfera

4.1.8. Coordenadas de Textura dos *Bezier Patches*

As coordenadas de textura dos *Bezier patches* são geradas mapeando linearmente os parâmetros de tesselação u e v no intervalo $[0, 1]$, correspondendo às posições relativas no *patch*. Para cada ponto gerado na grelha, o UV é simplesmente (u, v) , criando uma parametrização uniforme e contínua sobre a superfície. O valor de u varia na direção dos “*stacks*” e v na direção dos “*slices*” da tesselação do *patch*.

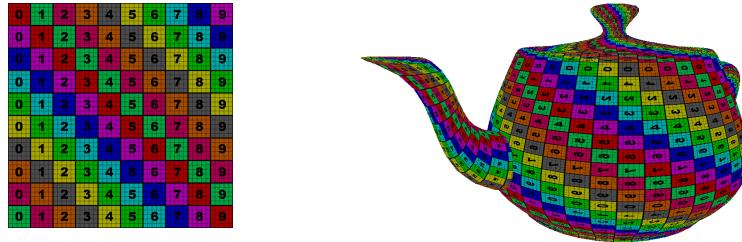


Figura 21: Demonstração de coordenadas de textura dos patches de beziér do *Teapot*

5. Formato final dos ficheiros .3d

O formato final dos ficheiros .3d, equivale exatamente a um .obj.

Os vértices, definem se através de:

`v <x> <y> <z>`

As normais, através de:

`vn <x> <y> <z>`

As coordenadas de textura, através de:

`vt <u> <v>`

E finalmente, cada triângulo ou “face” é definido através de:

`f <índice vértice 1>/<índice coordenada de textura 1>/<índice normal 1>
<índice vértice 2>/<índice coordenada de textura 2>/<índice normal 2> <índice
vértice 3>/<índice coordenada de textura 3>/<índice normal 3>`

```

v 0.5 0 0
v 0 0.5 0
v 0 0 0.5
vn 0 1 0
vt 0.5 1
vt 0 0
vt 1 0
f 1/3/1 2/2/1 3/1/1

```

Listagem 3: Exemplo do novo ficheiro .3d

6. Sistema Solar

De forma a aproveitar as funcionalidades adicionadas nesta fase, alterou-se a cena do sistema solar para esta ter texturas, materiais e luzes.

6.1. Texturas dos planetas

Foram adicionadas diversas texturas de alta resolução aos planetas que compõem o sistema solar.

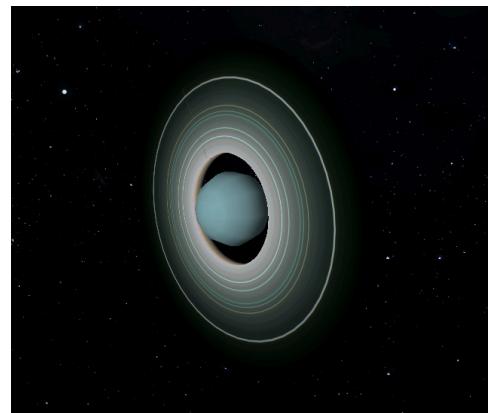
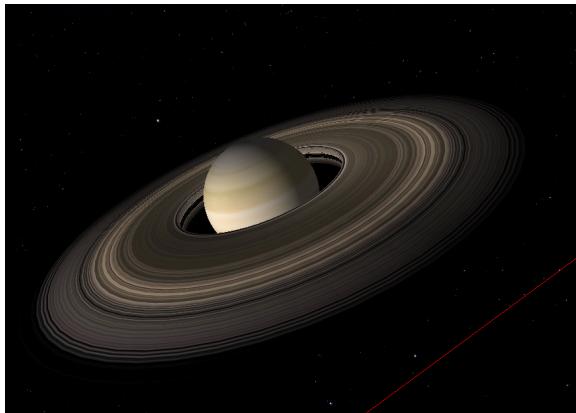


Figura 22: Terra e Lua, Sol, Saturno e Neptuno

6.2. Luzes

Apenas foi adicionada uma luz pontual no centro da cena para simular a luz do sol.

6.3. *Skybox*

Foi adicionada uma icoesfera de grande dimensão com textura de estrelas à cena para simular a vastidão do espaço. Para mostrar a textura no interior do modelo, a icoesfera foi escalada negativamente, estando invertida.



Figura 23: *Skybox*

6.4. Materiais

Para o Sol e para a *Skybox*, foram criados materiais com a emissiva no máximo em todos os campos (R, G, B) de forma a simular iluminação causada pelos modelos.

Para o cometa *Gumbo*, foi também adicionado um material com a emissiva no máximo no campo R, de forma a ficar completamente vermelho.



Figura 24: Cometa *Gumbo* (agora vermelho)

Todos os outros objetos na cena que não contêm textura (nomeadamente asteróides), têm um material de cor cinzenta.

7. Janela de configurações

Para facilitar a utilização da aplicação a utilizadores que não estejam acostumados com as *keybinds*, foi adicionada uma janela com *inputs* que alteram o estado desta.

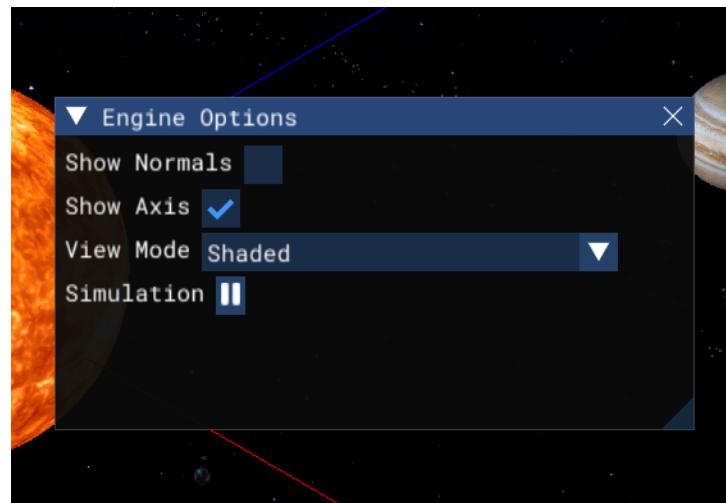


Figura 25: Nova janela

A utilização desta permite:

- Ligar o modo de visualização de normais
- Desligar e ligar os eixos
- Selecionar o modo de visualização de cena
- Parar e retomar a simulação

8. Conclusão

Concluindo, acreditamos ter alcançado um resultado muito positivo na finalização deste projeto. Além das funcionalidades obrigatórias, implementamos extras valiosos, ao longo de todas as 4 fases do trabalho prático.

8.1. Trabalho Futuro

8.1.1. *Shaders*

Gostaríamos de ter adotado o uso de *shaders* personalizados para lidar com a renderização de luzes e materiais. Isso dar-nos-ia muito mais flexibilidade e controlo sobre a pipeline gráfica, permitindo criar efeitos visuais mais complexos, como superfícies realistas (*physically-based rendering*), transparência, reflexos e outros efeitos avançados.

8.1.2. *Frustum Culling*

Implementar *frustum culling* como uma técnica de otimização na renderização da cena. Esta abordagem permite descartar automaticamente objetos que estão fora do campo de visão da câmera, reduzindo a carga de processamento gráfico e melhorando significativamente o desempenho, especialmente em cenas com grande quantidade de objetos (como a do sistema solar).

8.1.3. *Gizmos de transformações*

Consideramos interessante a implementação de *gizmos* de transformações interativos para facilitar a manipulação direta de objetos na cena. Algo como o que existe na maior parte dos programas de desenvolvimento 3D.