Computação Gráfica
Fase 4

Universidade do Minho
Escola de Engenharia

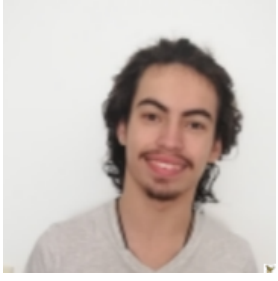
Trabalho realizado por:

Afonso Laureano Barros Amorim
Gonçalo Martins dos Santos
João Carlos Fernandes Novais
Telmo José Pereira Maciel

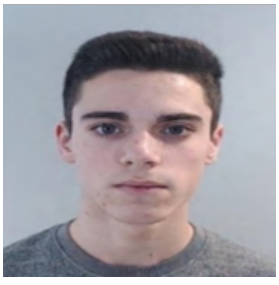
Grupo 26

A97569
Afonso Amorim

A95354
Gonçalo Santos



A96626
João Novais



A96569
Telmo Maciel

Índice

1 Introdução

Na quarta e última fase do trabalho prático de Computação Gráfica e por forma a implementar completamente o cenário do Sistema Solar, foi proposto desenvolver iluminação e texturas nos modelos.

Para isso, tivemos que definir as coordenadas de textura e as normais de cada um dos modelos. Para as normais, recorremos ao processo de normalização. Isto é bastante importante para os processos de aplicação de texturas e iluminação.

O produto final é o Sistema Solar completamente desenvolvido.

2 Alterações na arquitetura do projeto

2.1 Common

2.1.1 Classe Point2D

O grupo adicionou a classe *Point2D*, que representa um ponto de duas dimensões, para guardar as coordenadas de textutura nas Shapes.

2.1.2 Classe Shape

Uma vez que esta fase visa a adição de texturas e iluminação nas cenas, é necessário que as formas possuam os vetores normais em cada vértice e ainda as coordenadas de textura, que foi o que adicionamos.

Adicionamos dois atributos do tipo *std::vector* a esta classe, um de *Point* para as normais, e outro de *Point2D* para as coordenadas de textura.

2.2 Generator

Na realização desta quarta fase é pretendido que o gerador seja capaz de gerar as normais e coordenadas de textura.

2.3 Engine

Neste ficheiro, foi adicionado um vector para guardar todas as luzes presentes no cenário.

2.3.1 Classe Light

Foi adicionada a classe *Light* que representa a luz. Esta classe tem as componentes pos, dir e cutoff que podem ser dos tipos point, directional e spotlight, tal como pedido no enunciado.

2.3.2 Classe Model

Foi adicionada a classe *Model* que representa um Modelo com a *Shape*, ficheiro de textura e material (instância da classe *Color*, que correspondente ao *color* do ficheiro XML).

2.3.3 Classe Color

Foi adicionada a classe *Color* que representa um material aplicado a um Modelo e que tem as componentes todas da luz (difusa, ambiente, especular e emissiva), e ainda o valor de *shininess*, entre 0 e 128.

2.3.4 Classe *TextureLoader*

Foi adicionada a classe *TextureLoader* que é a classe que lida com o tratamento de texturas (ler os valores da cor dos píxeis do ficheiri e carregá-los para a memória gráfica para serem usados na renderização dos modelos).

Cada textura está assiciado ao ID, através de um *map<string,GLuint>*.

2.3.5 Classe *Group*

Esta classe tem os métodos *draw* e *drawPicking* usados para desenhar normalmente e para quando queremos usar o *picking* (no primeiro aplicamos a luz, texturas e mudanças de cor, enquanto no segundo apenas mudamos a cor para a cor que o modelo deve ter para ser reconhecido), respetivamente.

Também alteramos o tipo do atributo *models* para *std::vector<Model*>*.

2.3.6 Classe *VBOManager*

Esta classe lida com a geração dos vbo's e armazena os seus ID's, associando os vbo's de vértices, normais e de coordenadas de textura a um único ficheiro (faz isto usando 3 *maps*: *map<string,GLuint>*).

3 Alterações ao ficheiro XML

Com a adição de luzes, texturas e materiais, a estrutura do ficheiro XML alterou.

3.1 Luzes

É possível que agora apareça uma secção com a tag *lights*, que no seu interior pode conter até 8 elementos com a tag *light*, que tem que ter entre 4 e 8 atributos obrigatórios dependendo do seu tipo. O atributo *type* é obrigatório e deve ser igual a "point", "directional" ou "spotlight" (podendo ser reduzido para "spot" apenas). Para cada um dos tipos temos os seguintes atributos obrigatórios:

- *point* - *posX* (ou *posx*), *posY* (ou *posy*) e *posZ* (ou *posz*)
- *directional* - *dirX* (ou *dirx*), *dirY* (ou *diry*) e *dirZ* (ou *dirz*)
- *spotlight* - *posX* (ou *posx*), *posY* (ou *posy*), *posZ* (ou *posz*), *dirX* (ou *dirx*), *dirY* (ou *diry*), *dirZ* (ou *dirz*) e *cutoff*

3.2 Texturas e Materiais

De forma a suportar texturas e materiais, é possível agora dentro dos elementos do tipo *model* adicionar dois elementos adicionais, do tipo *texture* e *color*.

O elemento *texture* apenas recebe um atributo *file* que deve conter o ficheiro que contém a textura a ser usada.

O elemento *color* pode conter no seu interior a definição do valor das 4 componentes da luz (*diffuse*, *ambient*, *specular* e *emissive*), todas com os atributos R, G e B definidos, e ainda um elemento *shininess*, que deve ser entre 0 e 128.

4 Texturas e Materiais

4.1 Cubo

Para o cubo a textura pretendida é replicada em cada uma das faces, ficando com um aspeto deste tipo:

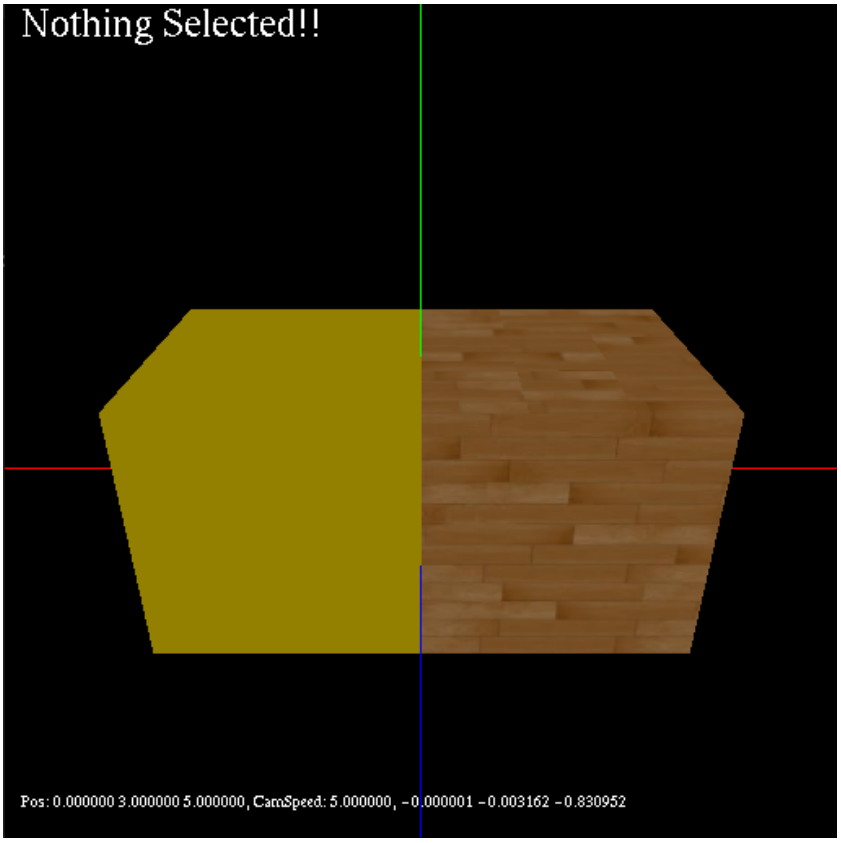


Figure 1: Exemplo de Material e Textura num cubo

4.2 Plano

No plano, as texturas são repetidas para cada divisão.

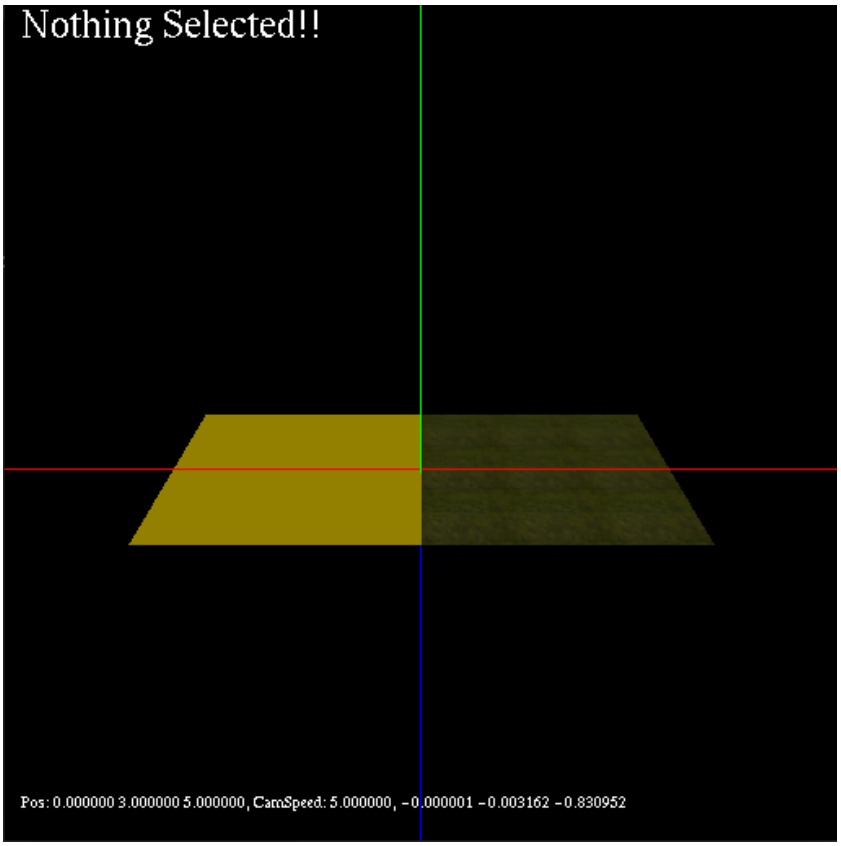


Figure 2: Exemplo de Material e Textura num plano

4.3 Cilindro

No cilindro fizemos de maneira similiar ao explicado nas aulas práticas para a ficha 13 (Textured Cylinder), ou seja, utilizamos a técnica de "UV Mapping", atribuindo a um vértice do cilindro um par de coordenadas (s,t) da textura, relativas ao pedaço da imagem correta.

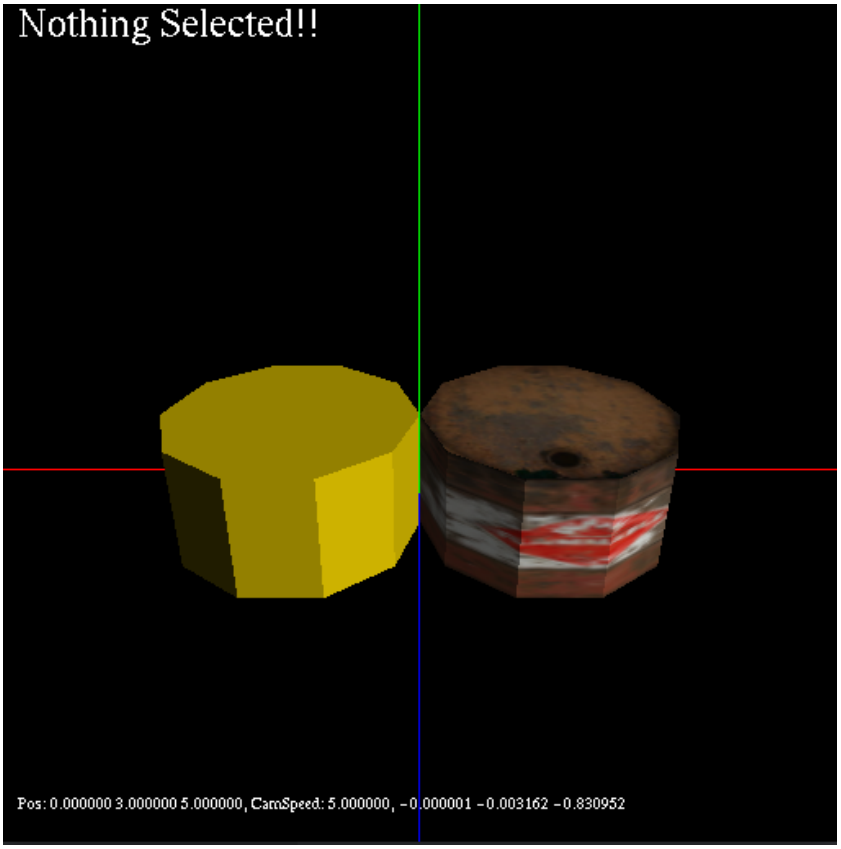


Figure 3: Exemplo de Material e Textura num cilindro

4.4 Cone

As coordenadas de textura do cone são divididas em 2 componentes: face de baixo e lateral.

Na face de baixo as coordenadas de textura são equivalentes a colocar a base por cima da textura com centro nas coordenadas s=0.5 e t=0.5.

Na face lateral, é como se cada geratriz do cone fosse colocada na vertical na textura.

Figure 4: Exemplo de Material e Textura num cone

4.5 Esfera

As coordenadas de textura na esfera são feitas com a seguinte fórmula:
$s = (1.0/(2.0 * \pi)) * (-alpha) + 1.0$ e $t = (1.0/(\pi)) * (beta + (\pi/2.0))$

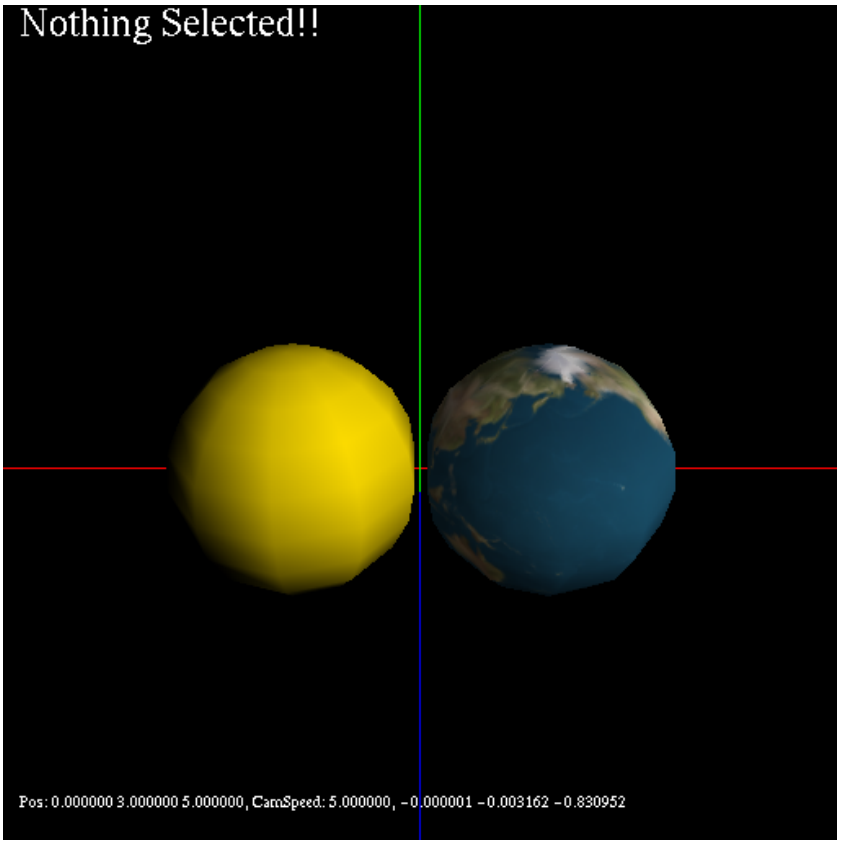


Figure 5: Exemplo de Material e Textura numa esfera

4.6 Washer

Na Washer, tal como no plano, repetimos a mesma textura para cada divisão.

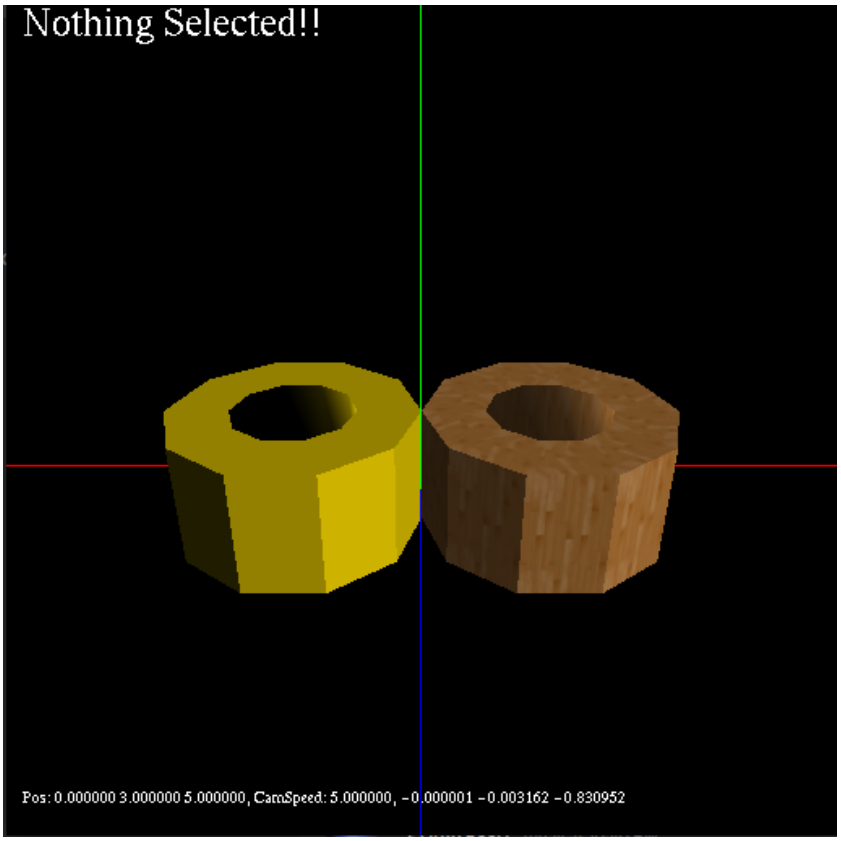


Figure 6: Exemplo de Material e Textura num washer

4.7 Nova forma adicionada

Foi adicionada uma nova forma, que consiste numa esfera virada para dentro usada para fazer o universo.

As coordenadas de textura na esfera inversa são feitas com a seguinte fórmula:
$s = (1.0/(2.0 * \pi)) * (-alpha) + 1.0$ e $t = (1.0/(\pi)) * (beta + (\pi/2.0))$

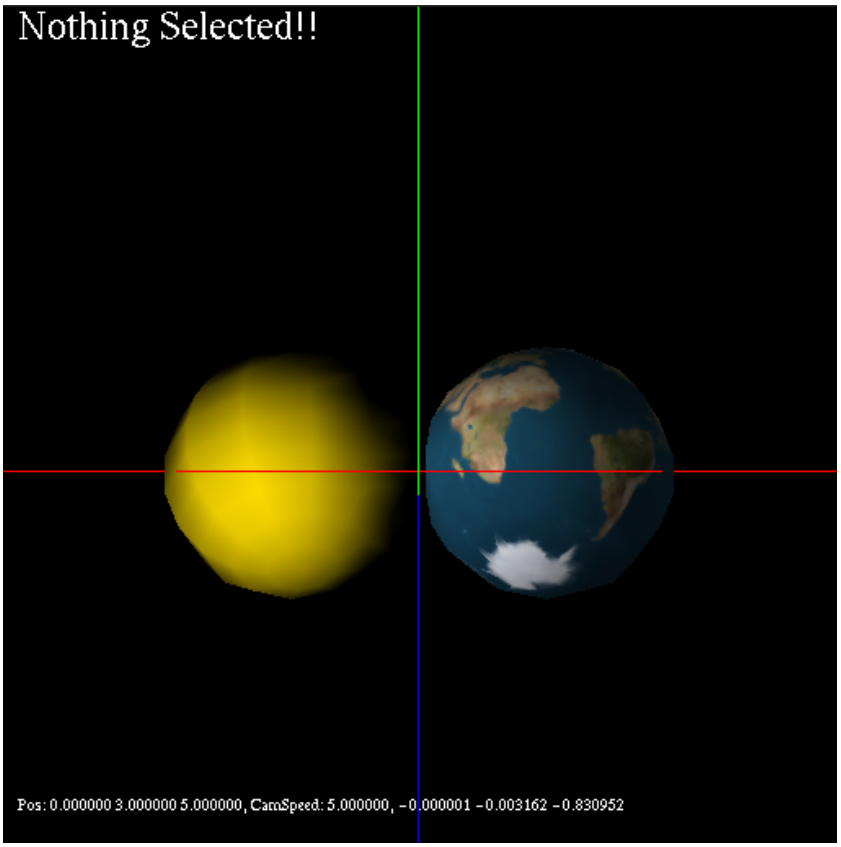


Figure 7: Exemplo de Material e Textura numa esfera invertida

5 Alterações no cenário final

No que toca ao cenário final apenas adicionamos as texturas e materiais para os planetas, ficando com o seguinte aspeto:

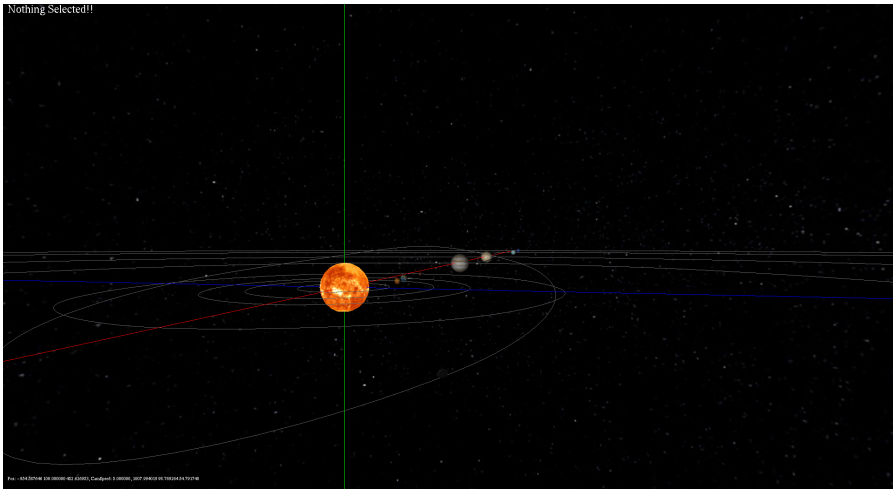


Figure 8: Modelo Final com uma visão lateral

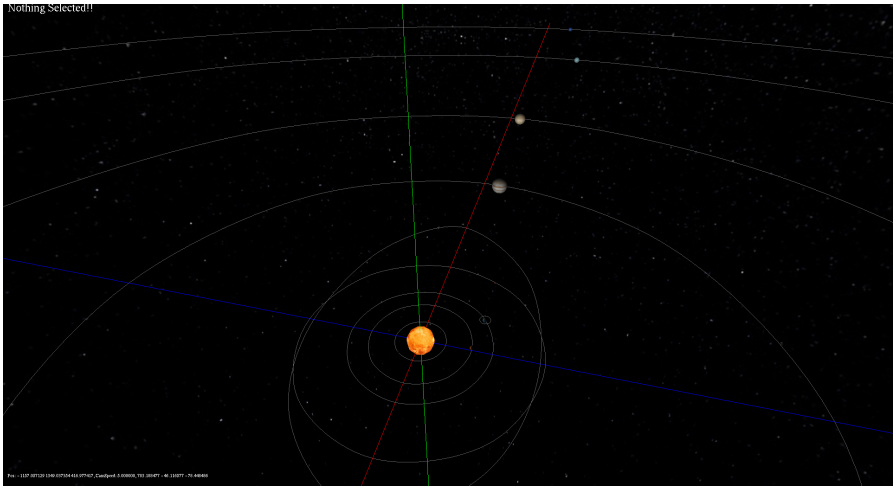


Figure 9: Modelo Final com uma visão superior

6 Conclusão

Com a conclusão da última fase do trabalho prático de Computação Gráfica, consideramos que cumprimos com todos os objetivos impostos em todas as fases.

Nesta fase tivemos que adicionar aos modelos as normais e as coordenadas de textura associadas para conseguirmos aplicar iluminação e texturas ao cenário final do Sistema Solar.

Olhando para todo o nosso percurso na conceção deste trabalho prático, concluímos que apesar de o trabalho nos ter apresentado bastantes dificuldades também apresentou-nos bastantes oportunidades para aplicar na prática todos os conceitos lecionados ao longo da Unidade Curricular.

Em suma, o trabalho foi desafiador mas, ao mesmo tempo, recompensador por termos um produto bastante realista do Sistema Solar.

Numa nota mais introspetiva, gostávamos de ter feito mais cenas de teste, mas estamos satisfeitos com o resultado final.