Relatório Quarta Fase Trabalho Prático - Computação Gráfica

Grupo 34 :
António Luís de Macedo Fernandes (a93312)
José Diogo Martins Vieira (a93251)
João Silva Torres (a93231)
Ricardo Lopes Santos Silva (a93195)

1 Maio 2022

Contents

1 Introdução **3**

2 Generator **4**

2.1 Esfera 4

2.2 Cone 4

2.3 Caixa 5

2.4 Plano 5

2.5 Cilindro 5

2.6 Teapot/Patches de Bezier 6

2.7 Torus 6

2.8 Estrutura dos ficheiros 3d 6

3 Engine **6**

3.1 Leitura dos ficheiros 3d 6

3.2 Luzes 6

3.3 Texturas 6

4 Extras **7**

5 Demo scenes **8**

6 Sistema Solar **10**

7 Conclusão **12**

1 Introdução

Para esta quarta e última fase do trabalho de Computação Gráfica o objetivo passa por adicionar texturas e iluminação, permitindo obter uma representação mais realista.

Ao longo deste relatório vamos explicar de forma detalhada as mudanças realizadas, bem como o tudo o que foi acrescentado.

Numa fase final, iremos apresentar algumas imagens, de forma a ilustrar o raciocínio utilizado, o produto final da representação do sistema solar e algumas conclusões do grupo sobre este trabalho.

2 Generator

De modo a poder aplicar as texturas e a iluminação às primitivas foi necessário alterar um pouco a maneira como as primitivas estavam a ser desenhadas. Então, em cada uma das funções das primitivas foi, também, efetuado o cálculo das coordenadas da textura e das normais de cada vértice.

2.1 Esfera

Para o cálculo das normais da esfera foi necessário calcular o vetor que vai da origem para cada vértice, para tal fazemos a diferença de cada ponto(P) com a origem(O): `N = P - O`.

Para o cálculo das texturas, começamos por calcular a divisão que vai ser feita, como queremos que a textura ocupe a figura todas, vamos fazer o uso de duas variáveis, para saber em que posição da textura nos encontramos: `textV = 1\stacks textH = 1\slices`. Desta forma, para cada ponto da esfera, conseguimos calcular o ponto correspondente na textura, formando assim a textura que ocupa a figura toda.

2.2 Cone

Para o cálculo das normais no cone, para os pontos que fazem parte de triângulos da superfície esférica, foi feita a normalização de cada ponto. Para os pontos fazem parte de triângulos da base, foi usado o vetor (0,-1,0) que aponta para baixo.

Para o cálculo dos pontos de textura foi seperado em duas partes, a base a superfície esférica. Para a superfície, utilizou-se um método semelhantes ao da esfera, a superfície esférica vai ser ocupada pela textura em si, e como tal é preciso fazer a divisão tanto da stacks como das slices. Utilizou-se assim: `textV = 1\stacks` `textH = 1\slices`. Desta forma, para cada ponto da superfície esférica, conseguimos calcular o ponto correspondente na textura, formando assim a textura que ocupa a superfície toda toda.

Para a base, como se torna impossível captar a textura toda sem deformação, optamos por outra estratégia.

Para tal, iremos usar apenas a variável `textH` para o cálculos dos pontos. Para cada divisão das slices, iremos usar os seguintes pontos:

```
baseNext = ((i+1)*textH,0)
baseNow = (i*textH,0)
center = ((i+0.5)*textH,1)
```

sendo os pontos baseNow e baseNext, os pontos que fazem parte da cirunferência, e center o ponto central da base.

Vejamos na imagem em baixo a explicação gráfica, sendo o ponto vermelho o centro, as linhas verdes o triângulo que irá ser formado, e as linhas azuis as divisões por slices.

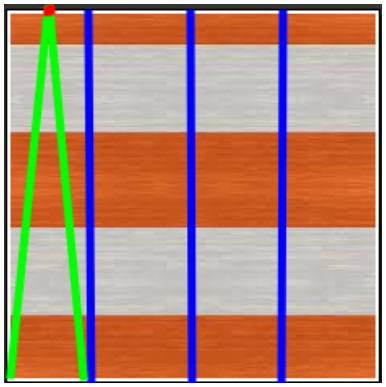

Figure 1: Explicação dos pontos de textura para a base do cone

2.3 Caixa

Para o cálculo das normais e das texturas para a caixa foi necessário alterar a nossa função *getPointsBox*. Então, para o cálculo das normais são efetuadas a partir da face em questão:

- **Face Frontal** - (0,0,1)
- **Face Traseira** - (0,0,-1)
- **Face Direita** - (1,0,0)
- **Face Esquerda** - (-1,0,0)
- **Topo** - (0,1,0)
- **Base** - (0,-1,0)

Para as coordenadas da textura estas vão ser aplicadas a cada face do cubo. Na caixa com divisões, a textura será aplicada a cada divisão.

2.4 Plano

Quanto ao plano, o cálculo das normais é simples, já que o vetor normal de cada ponto trata-se do vetor vertical (0,1,0). É de notar que este vetor apresenta direção de baixo para cima.

Para a textura do plano dividimos o plano e colocamos a textura nas várias divisões.

2.5 Cilindro

Para o cálculo das normais do cilindro, foi algo simples, para as bases foi calculado o vetor (0,1,0) para a de cima e (0,-1,0) para de baixo. Para os pontos laterais, foi usado apenas usados os valores de x (sin(angle)) e do z(cos(angle)), com o y a 0, assim irá ser criado um vetor perpendicular ao eixo do y.

Para o cálculo das texturas, para as bases foi usada a mesma estratégia aplicada para a base do cone. Para o calculo da face lateral, é utilizado o método já descrito anteriormente da divisão por slices e stacks.

2.6 Teapot/Patches de Bezier

Nas patches de Bezier, para o cálculo das normais foi usado o cálculo da normalização para cada ponto.

Para o cálculo das texturas, iremos aplicar a textura a cada patch. Para tal, iremos divir a patch em NxN partes, sendo N o tesselation dado. Para cada parte iremos fazer corresponder a parte da textura. No final, cada patch terá uma textura completa.

2.7 Torus

Para facilitar o cálculo das normais, iremos tratar o torus como se ele fosse sempre achatado para o uso como anel de Saturno. Iremos optar por tratá-lo como algo que tem sempre a mesma cor independente do lado, para tal iremos atribuir um único vetor para toda a figura. O vetor usado foi (0,1,0).

Para o cálculo das texturas, optamos por fazer com que as texturas se repetissem para cada divisão, assim para cada divisão, iremos ter como pontos de texturas valores em que X e Y são os cantos da textura (par de 0's ou 1's).

2.8 Estrutura dos ficheiros 3d

Devido à adição dos pontos da normal e da textura, é necessário colocar esses pontos no ficheiro 3d, para tal e de modo a reutilizar a função de escrita no ficheiro e permitir a utilização de ficheiros que não contenham estes pontos, irão ser escritos os pontos para as coordenadas da figura, seguidos dos pontos das normais e por fim, os pontos de textura, todos antes antes dos pontos necessitam de uma linha que indique o número de pontos que irão ser lidos.

3 Engine

Para suportar os requisitos desta última fase, também foi necessário realizar algumas alterações ao *Engine*. Nomeadamente, a mudança na leitura de ficheiros *XML* e *3d* mais complexos, a adição das luzes, assim como das texturas.

3.1 Leitura dos ficheiros 3d

Como já foi mencionado anteriormente, o engine permite a leitura de ficheiros que contenham apenas os pontos de coordenadas, assim como ficheiros que contenham todos os pontos. Para tal, será feito primeiramente o parsing da parte das coordenadas, esta é obrigatória, de seguida irá verificar se ainda existe mais linhas para ler, caso haja irá fazer o parsing para as normais, e de seguida, verifica outra vez e faz o parsing para as texturas, para este terá na mesma 3 coordenadas no ficheiro, mas será descartada a última, utilizando assim apenas as duas primeiras.

3.2 Luzes

Para as luzes, iremos criar uma nova classe que irá guardar os atributos para as luzes. Estas luzes irão ser guardadas na classe da câmara, e irão ser colocadas após o gluLookAt, consoante o número de luzes existentes irá ser feito o enable das mesmas.

3.3 Texturas

Para as texturas, iremos ter um nova variável global no engine, que será um array para guardar as texturas. Este array será preenchido no parsing do xml, iremos ter um mapa de texturas que irá ser preenchido com um par para cada ficheiro de texturas que existe. Quando tivermos a fazer a geração dos buffers para as coordenadas,

iremos fazer também a geração dos buffers para as normais e texturas. E, caso esse modelo tenha uma ficheiro de textura associada irá ser feita a leitura desse ficheiro, iremos guardar a variável no array de texturas, e guardamos o indíce da posição ao modelo associado. No final do parsing o array de texturas estará devidamente preenchido. Ao fazer o render para cada modelo iremos verificar se existe um index de texturas válido associado a este, caso exista, iremos buscar ao array de texturas.

Assim, fazemos a leitura de cada ficheiro apenas uma vez e fazemos a geração das texturas apenas uma vez, podendo ser utilizadas por diferentes modelos.

4 Extras

Como extras, de modo a tornar o engine mais iterativo, temos a funcionalidade de parar o movimento, desligar as luzes, mudar o *drawMode*. Estas funcionalidade e as respetivas teclas associadas são encontradas no terminal, juntamente com informação relativa ao tempo que demorou na preparação para o desenho e ao número de texturas e modelos.

```
./engine ../sistemaSolar.xml
Parsing completed
Reading files and generating buffers...
Buffers generated:
Models : 5
Textures : 13
Ready to draw | time of preparation :0.299000s
HELP: Key Bindings
Key                 Action
Left Mouse Button   Camera look around
Right Mouse Button  Camera zoom
Key Arrow Down      Reduce camera sensibility
Key Arrow Up        Increase camera sensibility
Key 'p'             Toggle Draw Mode
Key 'e'             Toogle Show axis
Key 's'             Toogle Movement
Key 'l'             Toogle Lighting
Key 'h'             Show this help
```

Figure 2: Informação no terminal após a execução do programa

5 Demo scenes

Nesta secção vamos apresentar a nossa solução para os ficheiros de teste fornecidos para esta fase.

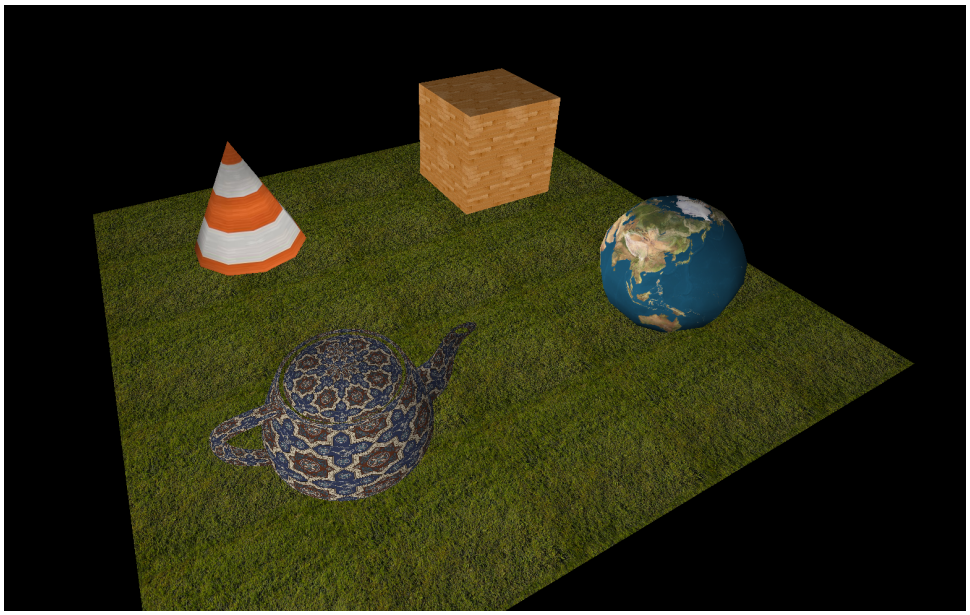

Figure 3: Ficheiro Teste

Figure 4: Demo Scene "lanche"

Agora vamos ilustrar as nossas primitivas com as respetivas texturas aplicadas.

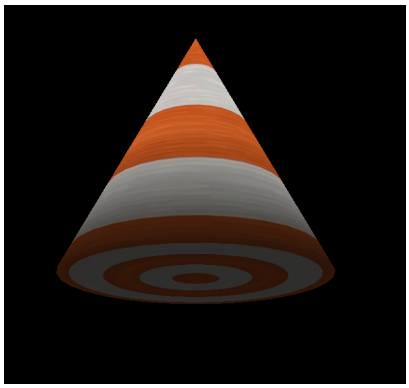

Figure 5: Cone com textura

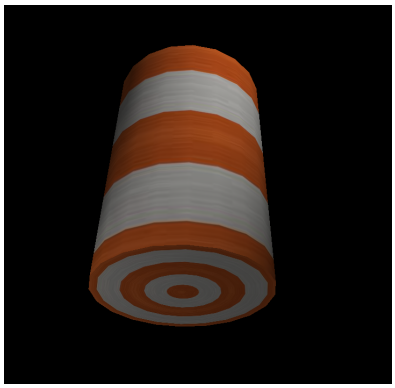

Figure 6: Cilindro com textura

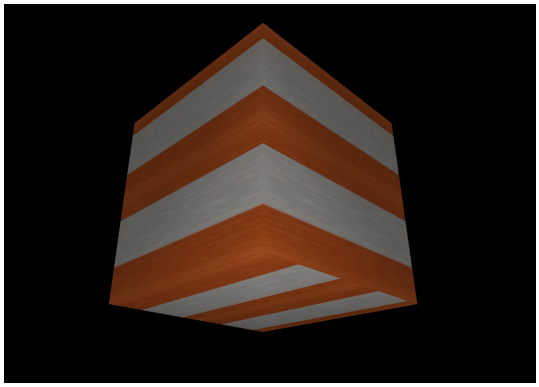

Figure 7: Box com poucas divisões

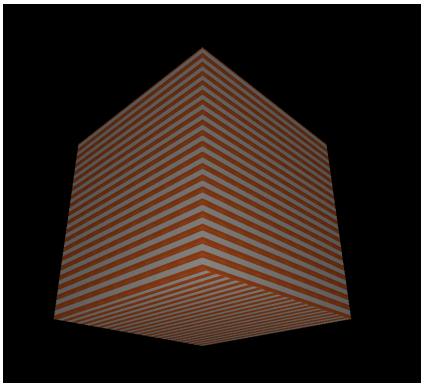

Figure 8: Box com muitas divisões

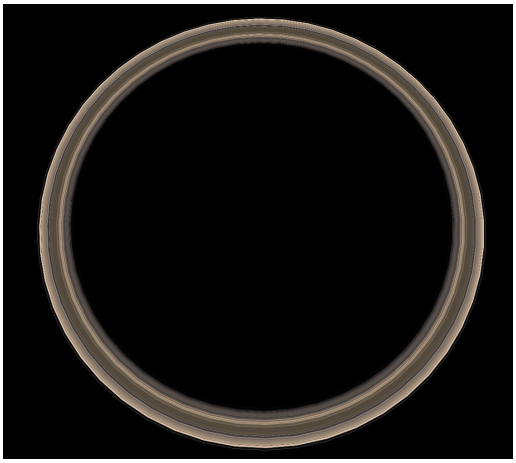

Figure 9: Torus com textura

6 Sistema Solar

Para o Sistema Solar, também adicionamos as respetivas texturas para as primitivas tornando o cenário mais realista.

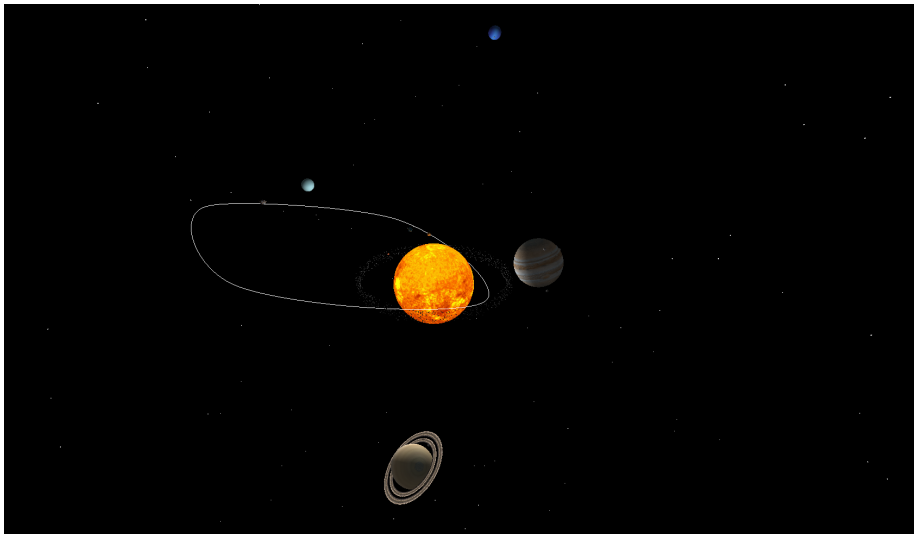

Figure 10: Sistema Solar de longe

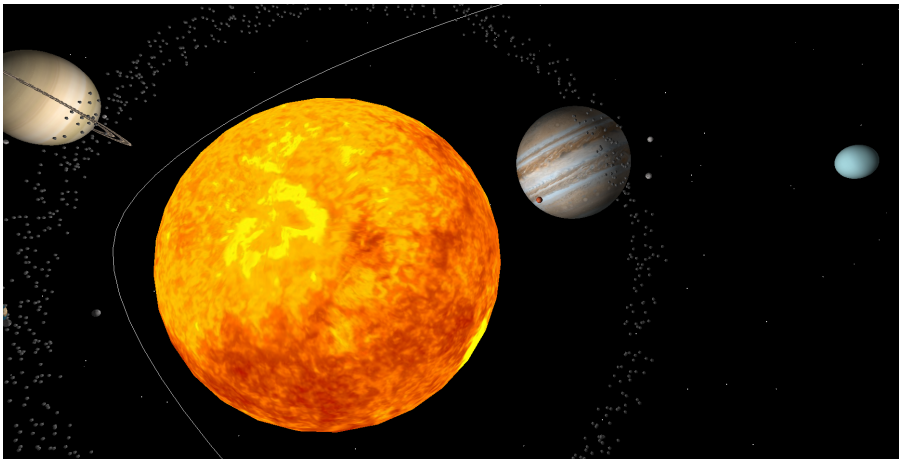

Figure 11: Sistema Solar de perto

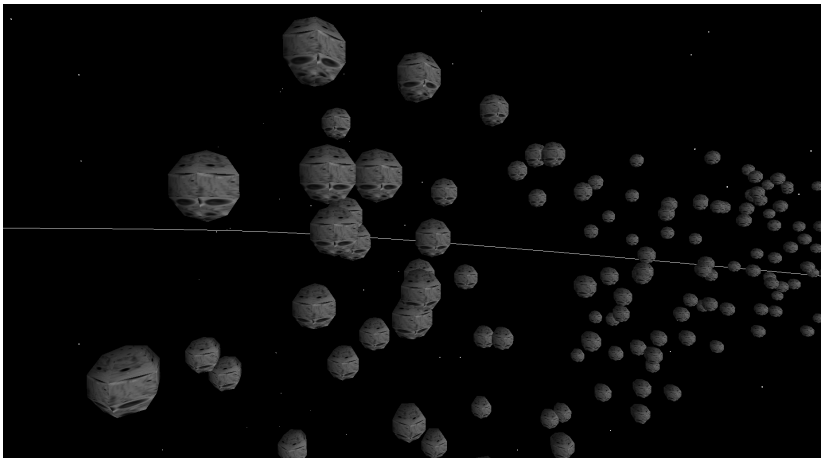

Figure 12: Cintura de Asteróides

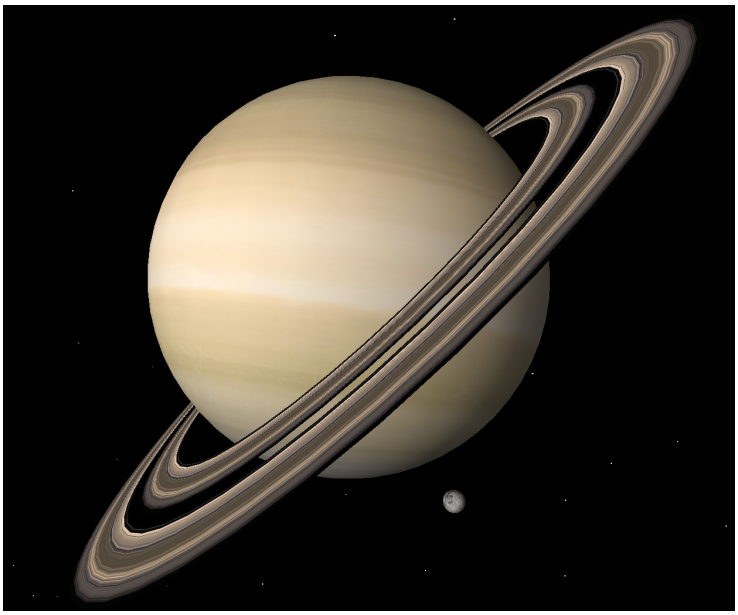

Figure 13: Saturno

7 Conclusão

Dado por completa a nossa fase 4 do projeto, iremos efetuar uma análise crítica do trabalho desenvolvido.

Depois desta quarta fase, consideramos que, no geral, este projeto foi bem conseguido, cumprindo com os requisitos de todas as fases e tentando sempre fazer alguns extras quando havia essa disponibilidade.

Consideramos ter conseguido estruturar bem o nosso projeto de modo a que seja fácil de procurar pela informação e achamos ter conseguido construir um sistema solar bem representativo do real. Surgiram algumas dificuldades ao longo desta fase mas após algum esforço e trabalho de equipa estas foram ultrapassadas.

Para concluir, o balanço deste projeto foi bastante positivo, cumprindo os requisitos e tendo ultrapassado as nossas dificuldades.