



UNIVERSIDADE DO MINHO
Departamento de Informática

Entrega 4 - Computação Gráfica



Realizado por:

André Mota - A81384 | Gonçalo Braga - A97541 | Rodrigo Pereira - A96561

<https://github.com/acmota2/CG2023>

4 de junho de 2023

Conteúdo

1	Introdução	2
2	Objetivos	2
3	Aplicação de Normais e Texturas no Gerador	3
3.1	Plano	3
3.2	Cubo	4
3.3	Esfera	5
3.4	Cone	6
3.5	Patch	7
4	Alterações no Sistema Solar	8
5	Alterações no motor 3D	9
5.1	Funcionalidades extra	9
6	Novas Funcionalidades	9
6.1	Script Python para executar o programa	9
6.2	Nova cena	10
6.3	Sistema solar alternativo	11
7	Conclusão	13

Conteúdo

1 Introdução

Esta parte do trabalho prático tem como objetivo principal a adição e utilização da iluminação e das texturas nas cenas criadas anteriormente.

Assim nesta parte, o gerador dos ficheiros **.OBJ** tem de ser capaz de acrescentar a cada vértice constituinte de modelo a sua normal (**Normalizada**) e a sua textura. Para além disso, o motor terá de ser capaz de reconhecer as mudanças feitas no gerador, criar fontes de luzes que são requeridas no XML, bem como interpolar a imagem da textura no objeto pedido.

2 Objetivos

Os objetivos para esta fase são:

- O gerador possuir a capacidade de calcular normais e texturas de um vértice;
- O motor 3D possuir a capacidade de aplicar as normais e as texturas à cena;
- Realizar alterações no Sistema Solar anteriormente criado;
- A adição de funcionalidades extra.

3 Aplicação de Normais e Texturas no Gerador

3.1 Plano

O plano é o objeto mais simples de aplicar as normais e as texturas. A normal do plano é um vetor perpendicular à sua superfície. Como este é sempre definido ao longo dos eixos XZ, o vetor normal à superfície terá as seguintes coordenadas: $(0,1,0)$.

Desta forma, todos os vértices pertencentes ao plano irão ter a normal mencionada anterior. As coordenadas da textura dos vértices do plano são calculadas da seguinte forma:

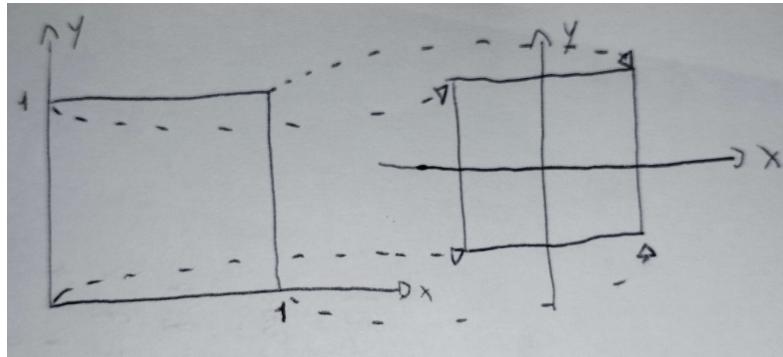


Figura 1: Explicação das coordenadas de textura no plano

Com base na imagem vista em cima, conseguimos perceber que os vértices que estão nas pontas do plano têm as seguintes coordenadas da textura:

- $(0,0)$ - canto inferior esquerdo;
- $(0,1)$ - canto superior esquerdo;
- $(1,0)$ - canto inferior direito;
- $(1,1)$ - canto superior direito.

Assim, com estas alterações, é possível observar o seguinte plano:

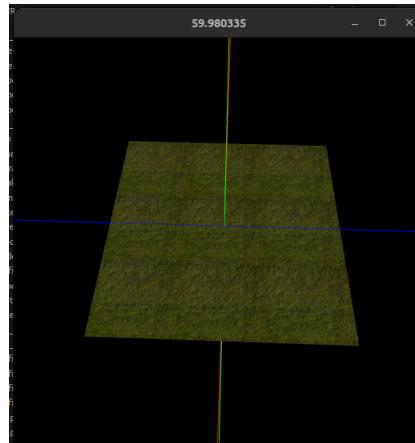


Figura 2: Plano obtido

3.2 Cubo

O cálculo das normais do plano são análogas às do cubo. Sendo assim, tendo em atenção que o cubo possui 6 faces, possuirá uma normal para todos os pontos pertencentes a cada face. Dessa forma, usando como referência eixos tridimensionais, iremos obter as seguintes normais: $(1,0,0)$, $(-1,0,0)$, $(0,1,0)$, $(0,-1,0)$, $(0,0,1)$ e $(0,0,-1)$.

Para uma coordenada de textura, calcula-se da seguinte forma:

```
m.pushTexture( vec3( i / ( float ) divisions , j / ( float ) divisions , 0 ));
```

Reparando no excerto de código acima, verifica-se que as coordenadas de textura são influenciadas pelo número de divisões (**divisions**) que o objeto possui. Assim, o *número de iteração* no qual estamos a dividir pelo número de divisões que o objeto possui, dando as coordenadas de textura desse mesmo vértice da figura.

O raciocínio demonstrado na *figura 1* foi também utilizado para o cálculo das coordenadas de textura para o cubo.

Dessa forma, com base nas alterações efetuadas, o cubo representa-se do seguinte modo:

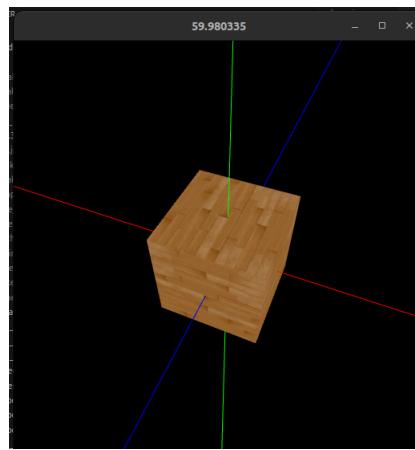


Figura 3: Cubo obtido

3.3 Esfera

As normais da esfera não podem ser calculadas como eram calculadas as normais para o plano e o cubo, uma vez que a superfície esférica não faz um ângulo reto com o vetor normal.

Se olharmos para uma esfera conseguimos perceber que os vetores perpendiculares à superfície esférica começam desde a origem até ao vértice pertencente à superfície esférica.

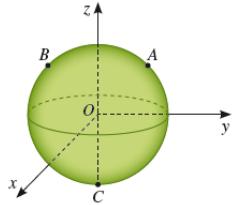


Figura 4: Esfera centrada na origem do referencial

Como podemos ver na imagem 4, um dos vetores normais à superfície esférica é o vetor OA. Como as normais de um determinado vértice são calculadas no mesmo instante de tempo que as dos vértices, é possível referenciar no ficheiro as normais tal como referenciamos as coordenadas dos vértices.

Desta maneira, com base nas alterações que foram feitas, a esfera fica com este aspeto:

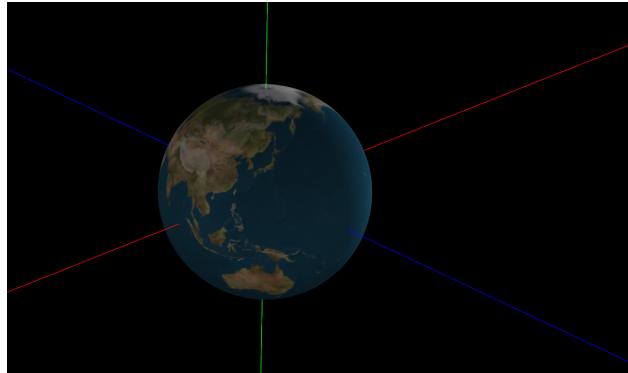


Figura 5: Esfera obtida

3.4 Cone

Para as normais do cone assume-se que o cone possui a altura h e raio r , voltando-se para cima (a sua ponta segue a direção do intervalo de Y's positivos), a lateral da sua face normal depende de dois ângulos: o ângulo da base circular que agrupa o chão, e o ângulo da sua ponta (simplificando, diremos que é o ângulo cônico ou). Este "ângulo cônico" dependerá da razão de h com r .

Olhando para o corte transversal do cone, é possível verificar um triângulo reto de catetos h e r . Assumindo que o cateto h sobe o eixo dos Y's a partir da origem, e o cateto r sobe pelo eixo dos X's, calcula-se a normal da hipotenusa que aponta para fora.

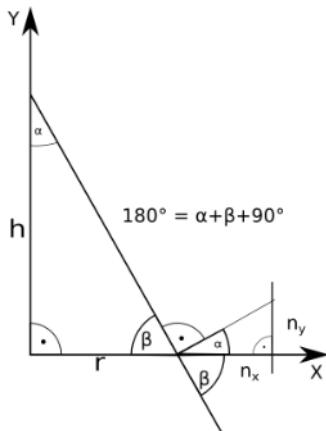


Figura 6: Esquema exemplificativo para o cálculo das normais do cone

Após os cálculos efetuados, é possível determinar que a hipotenusa da normal apresenta-se da seguinte forma:

$$(\cos(\text{angulo_conico}), \sin(\text{angulo_conico}))\}$$

sendo o "angulo_conico" representado pela seguinte fórmula:

$$\text{angulo_conico} = \text{atan}(r / h)$$

Para determinarmos a sua forma tridimensional, foi preciso determinar a normal de um círculo localizado no plano XZ:

$$(\cos(\text{angulo_conico}), 0, \sin(\text{angulo_conico}))$$

Sabendo isto, foi possível combinar ambos numa equação normal. O declive da normal possui partes horizontais e verticais, tendo as verticais automaticamente sido associadas à coordenada Y, enquanto que as horizontais contribuiram para as direções horizontais (X e Z):

$$(\cos(\text{coneAngle}) * \cos(\text{circleAngle}), \sin(\text{coneAngle}), \cos(\text{coneAngle}) * \sin(\text{circleAngle}))$$

Chegando a esta conclusão, foi deduzido que existem realmente dois vetores. O vetor que aponta para cima, para a ponta do cone, e o horizontal, gerado pela normal do círculo. Ambos os vetores formam a base, tendo sido aplicada uma transformação linear dos eixos bidimensionais XY da normal do cone, para espaço criado pela normal do círculo e o vetor que aponta para cima. Para tal, foram multiplicados os componentes XY do plano bidimensional com os respetivos vetores base do espaço tridimensional, resultando na seguinte regra:

$$(\cos(\text{coneAngle}) * (\cos(\text{circleAngle}), 0, \sin(\text{circleAngle})) + \sin(\text{coneAngle})) * (0, 1, 0)$$

As coordenadas de textura para foram feitas de forma análoga à da esfera.

3.5 Patch

Para os modelos que desejamos, partindo dos ficheiros **.patch** possuímos normais e texturas. Dessa forma, as normais do modelo foram calculadas seguindo:

```
puv_uderiv = multMathVec(v_vetor, u_vetorXcalculada_uderiv);
puv_vderiv = multMathVec(v_vetor_deriv, u_vetorXcalculada);
vec3 normal = puv_vderiv.get_cross_product(puv_uderiv);
normal.normalize();
normaisFinais[u][v] = normal;
```

Sendo, deste modo, possível ver que o vetor normal é calculado com base nos vetores existentes no cálculo das superfícies de *Bzier*.

Analogamente aos modelos anteriores, este também possui coordenadas de textura para os vértices pertencentes ao mesmo. Assim, as coordenadas de textura são calculadas da seguinte forma:

```
m.pushTexture(vec3(1-(k/(float)tess), (i/(float)tess), 0));
```

Neste caso, a *tesselation* cria interferências na maneira de calcular as coordenadas de textura de um determinado vértice.

Desta maneira, com base nas alterações que foram feitas, o modelo fica com o seguinte aspeto:

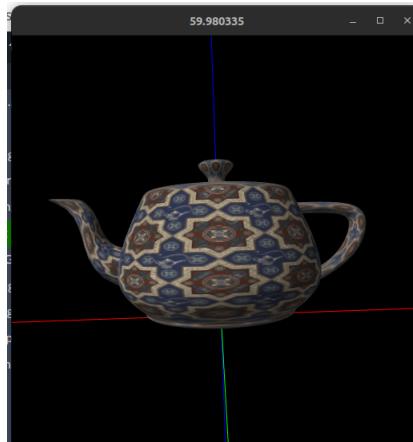


Figura 7: Teapot obtido

4 Alterações no Sistema Solar

O Sistema Solar atinge nesta fase o seu "momento alto", quando uma vez alterado, aproximar-se-á cada vez mais da realidade. Esta aproximação é conseguida através da aplicação das texturas às esferas (onde se aplicará as texturas aos planetas e ao Sol), e da criação da fonte de luz (que, neste caso, é o Sol).

Estas duas alterações foram conseguidas após a adaptação do motor 3D e gerador, de forma a lidar com as normais e as texturas dos modelos em questão.

Assim com as alterações efetuadas no Sistema Solar este fica com a seguinte aparência:

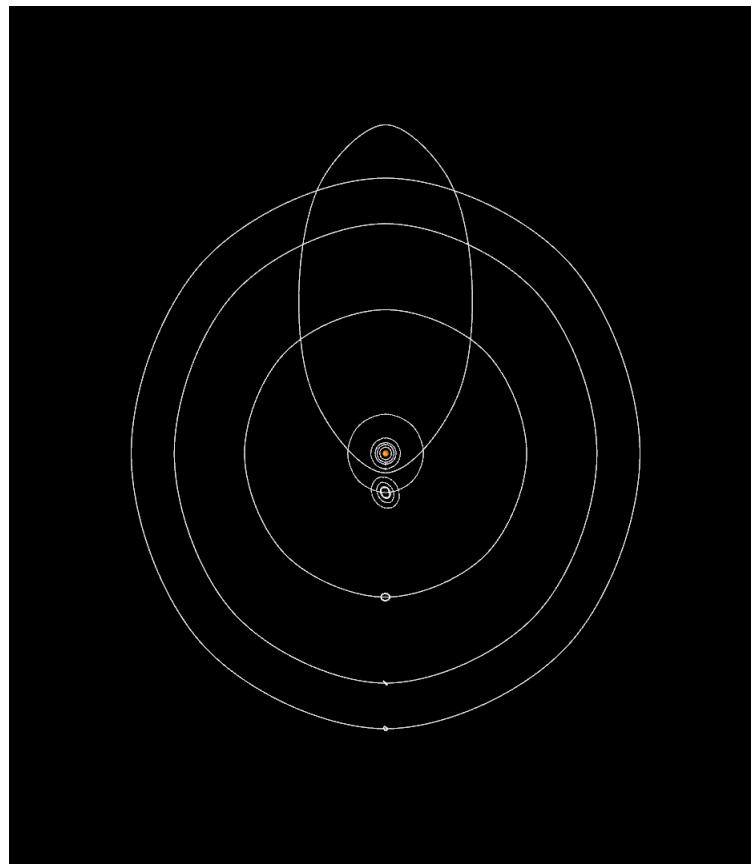


Figura 8: Vista aérea do Sistema Solar

Entre as alterações esperadas para esta fase, foram ainda melhoradas as órbitas, que ficaram mais próximas da perfeição. Isto deve-se ao facto de se utilizarem 8 pontos de controlo, dos quais 4 são os pontos que coincidem com os eixos x e z, sendo os restantes 4 pontos as interceções entre a equação da elipse utilizada e a equação da circunferência de raio correspondente à média do a e b da elipse.

5 Alterações no motor 3D

Baseando nas alterações feitas no gerador, o motor 3D foi capaz de suportar estas alterações, sendo estas:

- Normais
- Texturas
- Materiais

As alterações feitas ao motor 3D prendem-se aos seguintes pontos:

- A criação lógica necessária para ler informação sobre luzes, texturas e materiais;
- A criação de uma classe **VBO** para gerir os buffers de coordenadas, de normais e de texturas;
- A criação de uma classe *light* para gerir a lógica das luzes(ativar a luz e definir ao desenhar);
- A criação de uma classe Texture para carregar as imagens ao inicializar a cena e dar *bind* as imagens das texturas no momento de desenho;
- A criação de uma classe material para carregar o materiais e as suas cores durante a inicialização e para as aplicar antes de desenhar a geometria em que queremos o material;
- A modificação do método para desenhar eixos(passa a haver três *VBO's* com vértices correspondentes a *GL_LINES* para desenhar cada eixo cada um com tamanho 1, após a sua representação e alteração de um *scale* para o tamanho desejado de desenho dos eixos. Foi preciso aplicar materiais da cor dos eixos para estes terem cor quando a iluminação está ligada).

5.1 Funcionalidades extra

Nesta fase, foram acrescentadas ao motor 3D sincronização vertical (*v-sync*) e *anti-aliasing* (MSAA), através das extensões GLX de OpenGL para Linux.

A nível de otimizações do motor 3D, os **VBO** e texturas, uma vez carregados, são carregados apenas essa vez. Isto é possibilitado através do nome do ficheiro carregado, que se já for *conhecido* do motor 3D, apenas volta a ser referenciado.

A nível de parsing de .xml, a tag **group** agora incorpora também a capacidade de carregar ficheiros .xml externos, de forma muito semelhante ao funcionamento de macros em C e C++.

6 Novas Funcionalidades

6.1 Script Python para executar o programa

Para automatizar todos os processos necessários para executar o programa, como configurar a *cmake file* gerar a *makefile* e por fim executar a *makefile*, foi criada uma script que faz estes processos todos ficarem unificados numa só linha, tal como :

```
python3 .. / test_files_custom / helper.py .. / test_files_phase_4 / test_4_1.xml
```

Neste comando é necessário especificar onde está o script a ser utilizado, bem como o ficheiro de teste que queremos testar.

Desta forma, o script que foi criado foi o seguinte:

```

import re
import sys
import os

comment_detector = re.compile(r'(?=<!--\s)(?:\.\.\ /)? generator .*?(?=\s-->)')

f = open(sys.argv[1], 'r')
full_file = f.read()

os.system("make clean")
os.system("make")

command: str
for command in comment_detector.findall(full_file):
    try:
        if(command.startswith("./")):
            os.system(command)
        else:
            os.system("./"+command)
    except:
        print('No such command')

os.system(f"./engine {sys.argv[1]}")

```

6.2 Nova cena

De forma a enriquecer o trabalho, decidiu-se criar uma nova cena. Esta cena utiliza modelos criados com base em seis ficheiros *.patches* tal como :

- Pato de Borracha;
- Banheira;
- Torneira;
- Sanita;
- Lavatório;
- Planos, que servem de paredes de casa de banho.

De salientar, que alguns deste ficheiros foram retirados de outras fontes, tal como [free3d](#) e [cgtrader](#). Desta forma também conseguimos testar o motor 3d, comprovando que este pode receber ficheiros de diversas fontes, além dos gerados por nós. Desta forma, a cena criada foi a seguinte:



Figura 9: Cena adicional

6.3 Sistema solar alternativo

Baseado no tema do trabalho ser o sistema solar, foi criado ainda, como extra, o modelo geocêntrico do sistema solar, no qual se pode observar a terra no centro com todos os planetas existentes até à época e o Sol a girar à sua volta. Sendo assim, o modelo geocêntrico considerado tem, por esta ordem, à volta da terra:

- Lua;
- Mercúrio;
- Vénus;
- Marte;
- Júpiter;
- Saturno.

Esta representação pode ser observada na seguinte imagem:

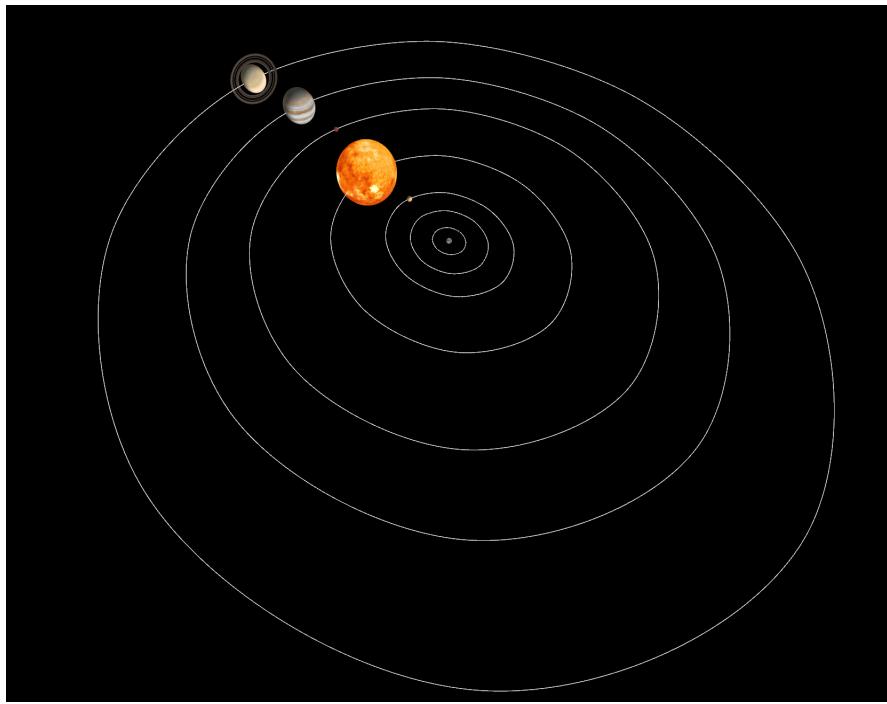


Figura 10: Sistema geocêntrico

7 Conclusão

Tendo em conta o trabalho desenvolvido nesta fase, acreditamos ter atingido os objetivos propostos para esta fase.

Como era sabido, o objetivo desta fase do trabalho prático seria acrescentar normais e texturas aos modelos gerados pelo gerador e fazer com que o motor 3D aplicasse essas alterações à cena.

Com isto, acreditamos que atingimos o objetivo proposto para esta fase do trabalho prático.

Como esta seria a última fase do trabalho prático, e analisando agora todo o trabalho desenvolvido ao longo das diferentes fases do trabalho prático, acreditamos que concluímos este trabalho prático com os objetivos definidos inicialmente alcançados, uma vez que ao longo de todas as fases sempre fomos alcançando o objetivo proposto para cada uma.

Assim acreditamos que este projeto nos ajudou a por em prática a matéria lecionada tanto nas aulas teóricas como nas aulas práticas.