



Universidade do Minho
Escola de Engenharia

Licenciatura em Engenharia Informática 2021/2022

Computação Gráfica

Trabalho prático - Fase 4 **- Normals and Texture Coordinates -**

5 de junho

Benjamim Miranda Costa (A87985)

Maria Sofia Rocha Gomes (A93314)

Marisa Ferreira Soares (A92926)

Miguel Rodrigues Santa Cruz (A93194)

Índice

Índice	2
Introdução	3
Classes criadas/atualizadas	3
Cálculo das coordenadas da normal	5
2.1. Vetor normal do cone	5
2.2. Vetor normal da esfera	5
2.3. Vetor normal do plano	5
2.4. Vetor normal da caixa	5
Iluminação	6
Demo Sistema Solar	7
Conclusão	8

1. Introdução

No presente relatório será explicado a estratégia da conceção desta quarta fase, as principais alterações em relação às fases anteriores, uma vez que foi necessário adicionar novas funcionalidades.

O principal objetivo desta fase do trabalho é obter as coordenadas dos vetores normais e de textura para cada vértice. Também é pretendido que se crie funcionalidades de iluminação e de textura.

2. Classes criadas/atualizadas

Para suportar as funcionalidades correspondentes a esta fase foi atualizada a classe *Model*. Foi adicionado um campo correspondente ao nome do ficheiro que contém a textura a aplicar ao modelo.

Foram também adicionados campos correspondentes aos diferentes componentes da cor do objeto (difusa, ambiente, especular, emissiva e valor de “shininess”).

```
class Model {
public:
    std::string filename = "NaN";
    std::list<Point> points;
    std::list<Point> normals;
    std::list<Transformation> transformations{};
    std::string texture = "NaN";
    float diffuse_color[4] = { [0]: 50, [1]: 50, [2]: 50, [3]: 1 };
    float ambient_color[4] = { [0]: 50, [1]: 50, [2]: 50, [3]: 1 };
    float specular_color[4] = { [0]: 0, [1]: 0, [2]: 0, [3]: 1 };
    float emissive_color[4] = { [0]: 0, [1]: 0, [2]: 0, [3]: 1 };
    float shininess = 0;
    void drawModel() const;
    void printOut();
};
```

Figura 1. Classe *Model* atualizada

Para implementar as funcionalidades de iluminação foi criada a classe *Light*. Esta classe possui 2 vetores correspondentes à direção e a posição da luz em questão, bem como o parâmetro “*cutoff*”, a aplicar no caso de se tratar de uma *spotlight*.

```
class Light {  
public:  
    std::string type;  
    float posicao[3] = { [0]: 0, [1]: 10, [2]: 0 };  
    float direcao[3] = { [0]: 1, [1]: 1, [2]: 1 };  
    float cutoff = 45;  
    void printOut();  
};
```

Figura 2. Classe Light

3. Cálculo das coordenadas da normal

2.1. Vetor normal do cone

Como a base do cone está contida no plano XZ todos os seus pontos têm o mesmo vetor normal $(0, -1, 0)$. Quanto aos pontos pertencentes à superfície lateral consideramos um triângulo retângulo formado pelo ponto que pretendemos obter a normal (ponto P), o ponto de altura máxima, o vértice do cone (ponto V) e o ponto obtido pela projeção ortogonal de P no eixo Y (ponto R). Através da lei dos senos é possível obter o valor de R e assim o vetor normal é $P - R$ normalizado.

2.2. Vetor normal da esfera

Para cada ponto, o vetor normal é obtido pela diferença entre o ponto que se pretende obter a normal e o centro da esfera. Por fim, o vetor obtido é normalizado.

2.3. Vetor normal do plano

As normais dos vértices do plano possuem todas o valor $(0, 1, 0)$, correspondente ao vetor a apontar para cima.

2.4. Vetor normal da caixa

Para as normais da caixa utiliza-se o mesmo método que o plano tendo em conta a direção do vetor normal de forma a que esta esteja em direção para fora.

4. Iluminação

Após estarem definidas as normais e ter sido implementada a classe *Light* foi criada uma lista denominada *lightsToDraw* que contém todas as luzes que foram lidas do ficheiro XML. Durante a inicialização são ativadas as luzes lidas, até um máximo de 8.

```
GLfloat dark[4] = { [0]: 0.2, [1]: 0.2, [2]: 0.2, [3]: 1.0};
GLfloat white[4] = { [0]: 1.0, [1]: 1.0, [2]: 1.0, [3]: 1.0};
float amb[4] = { [0]: 1.0f, [1]: 1.0f, [2]: 1.0f, [3]: 1.0f };
glLightModelfv( pname: GL_LIGHT_MODEL_AMBIENT, params: amb);
int index = 0;
for (Light l : lightsToDraw) {
    if(index >= 8) break;
    // Enable LIGHT0(0x4000) to LIGHTi(0x400i)
    glEnable( cap: 0x4000+index);
    glLightfv( light: 0x4000+index, pname: GL_AMBIENT, params: dark);
    glLightfv( light: 0x4000+index, pname: GL_DIFFUSE, params: white);
    glLightfv( light: 0x4000+index, pname: GL_SPECULAR, params: white);
    index++;
}
glEnableClientState( array: GL_VERTEX_ARRAY);
glEnableClientState( array: GL_NORMAL_ARRAY);
```

Figura 3. Inicialização das luzes

Após a inicialização das luzes foram definidos os materiais para os respectivos modelos, dentro da função *renderScene()*, antes do desenho dos mesmos.

```
glMaterialfv( face: GL_FRONT, pname: GL_DIFFUSE, params: diffuse_color);
glMaterialfv( face: GL_FRONT, pname: GL_AMBIENT, params: ambient_color);
glMaterialfv( face: GL_FRONT, pname: GL_SPECULAR, params: specular_color);
glMaterialfv( face: GL_FRONT, pname: GL_EMISSION, params: emissive_color);
glMaterialf( face: GL_FRONT, pname: GL_SHININESS, param: shininess);
```

Figura 4. Definição dos materiais antes do desenho dos respetivos modelos

5. Demo Sistema Solar

Nesta fase foi atualizado a demo do sistema solar que temos vindo a utilizar (*SolarSystemPhase4.xml*). Foi definido no ficheiro XML informação relativa às diversas componentes da cor dos diferentes planetas e foram definidas as luzes a utilizar na cena construída.

O resultado de executar o engine tendo este ficheiro como argumento.

```
./engine SolarSystemPhase4.xml
```

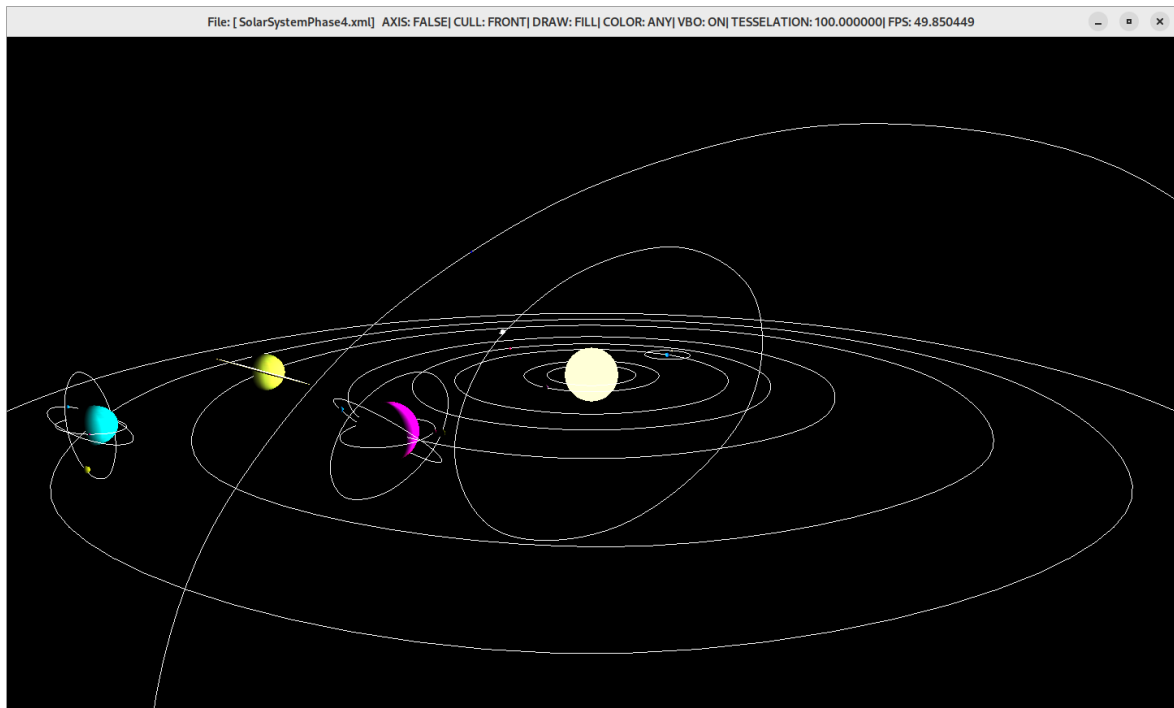


Figura 5. Demo Solar System

```

CG@CG2022 $ ./engine SolarSystemPhase4.xml
XML File: SolarSystemPhase4.xml
==Camera=====
Camera Configurations:
Position x: 300 y: 100 z :300
LookAt x: 0 y: 0 z :0
Up x: 0 y: 1 z :0
FOV: 60
Near: 1
Far: 1000
==Lights=====
Type : point
Position: 0.000000 0.000000 0.000000
-----
==Model=====
Model Filename: teapot.3d
Number of points: 15552
Number of normals: 0
Texture File: asteroid.jpg
COLOR INFO -----
Emissive R: 0.000000 G: 0.000000 B: 0.000000
Specular R: 0.000000 G: 0.000000 B: 0.000000
Ambient R: 0.000000 G: 0.000000 B: 0.000000
Diffuse R: 0.784314 G: 0.784314 B: 0.784314
Shininess: 0.000000
Transformations List -----
Rotate angle: 45.000000 x: 1.000000 y: 0.000000 z: 0.000000
Translate time: 14.000000 align: 1
List of points -----
Point x: 130.000000 y: 0.000000z: 0.000000
Point x: 91.000000 y: 0.000000z: 91.000000
Point x: 0.000000 y: 0.000000z: 130.000000
Point x: -91.000000 y: 0.000000z: 91.000000
Point x: -130.000000 y: 0.000000z: 0.000000

```

Figura 6. Output do comando anteriormente referido

6. Conclusão

Depois de realizada esta quarta fase do trabalho prático, conseguimos implementar a iluminação. No entanto, as normais do cone possuem ligeiras anomalias que impedem a correta iluminação deste. No caso dos patches de b ezier n o foram implementadas as normais dos mesmos pelo que a ilumina  o destes objetos n o   apresentada corretamente.

Relativamente  s implementa  o de texturas dos respetivos modelos, foi implementada a leitura dos nomes dos respectivos ficheiros, no entanto, o carregamento e apresenta  o das mesmas n o foi realizada.

Conclu mos que o trabalho realizado nesta fase n o foi o esperado, uma vez que certos objetivos desta fase n o foram implementados ou ficaram aqu m do expect vel definido pelo grupo.