Computação Gráfica

3º ano de LCC/MIEI

4º Fase de Entrega

Relatório de Desenvolvimento

Normals and Texture Coordinates

João Gomes

a70400

João Dias

A72095

Joel Morais

A70841

Luís Ventuzelos

a73002

21 de Maio de 2017

Resumo

Este documento apresenta o desenvolvimento do projeto de *Computação Gráfica* numa colaboração entre o Mestrado Integrado de Engenharia Informática *(MIEI)* e a Licenciatura de Ciências de Computaçao *(LCC)* correspondente ao ano lectivo *2016/2017*, analisando as estratégias utilizadas e as principais funções implementadas em cada uma das tarefas propostas.

O nosso principal objetivo nesta fase remete para a introdução de texturas, juntamente com a iluminação, do nosso respetivo Sistema Solar.

Terão que ser feitas alterações, tanto na parte do motor, como no gerador, sendo que este irá gerar as coordenadas nas normais e de textura, para posteriormente o motor conseguir ler essas mesmas coordenadas.

Capítulo 1

Introdução

Nesta última fase do projeto serão feitas alterações tanto no nosso motor como no nosso gerador.

O gerador irá estar encarregue coordenadas das normais e de textura, como dito anteriormente, para isso será necessário criar duas funções que tratem do cálculo e da normalização do vetor normal a uma superfície.

Já no caso do motor, este irá servir para ler estas mesmas coordenadas e para carregar a textura que irá fazer parte de cada planeta e do Sol.

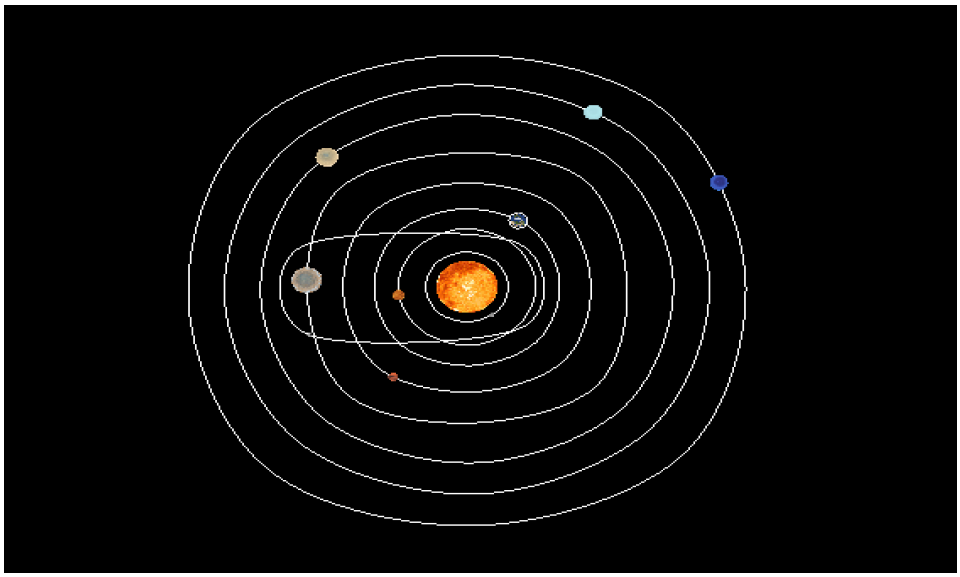

Figura 1.1: Sistema Solar

Capítulo 2

Texturas

2.1 Texture mapping

O objetivo principal pela qual iremos usar texturas visa frisar que a aplicação das mesmas irá deixar o nosso trabalho com um aspeto muito mais aproximado com a realidade, sem ser apenas um conjunto de objetos iguais e sem qualquer tipo de apelo visual.

As nossas texturas irão ser representadas em duas dimensões, e através de algumas operações, iremos ver texturas 2D em objetos 3D.

Para conseguirmos aplicar estas texturas irá ser necessário que a nossa imagem seja repartida em várias divisões, formando uma grelha, sendo que cada triângulo desta imagem 2D tem que corresponder a um triângulo da superfície da esfera, tal como exemplificado nesta seguinte imagem:

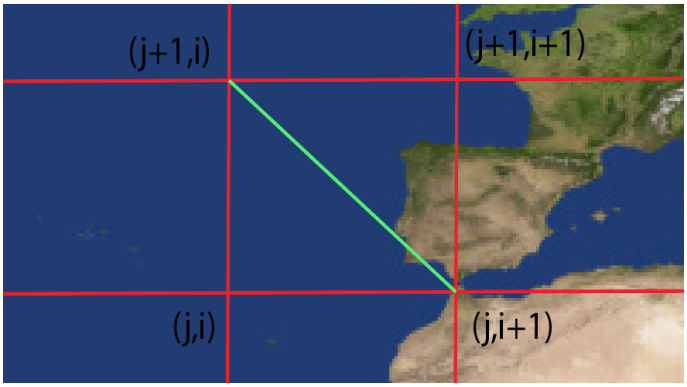

Figura 2.1: Textura aplicada numa esfera

É esta dita aplicação das texturas nos nossos objetos que irá ter o nome de texture mapping.

2.1.1 Como aplicar as texturas

Para cada vértice irão ser fornecidas as coordenadas de textura, de modo a serem posteriormente tratados.

Felizmente, com a utilização do OpenGL, visto que cada vértice vai conter a sua própria coordenada da textura, não nos temos de preocupar com certos problemas relativamente à colocação da textura, como por exemplo, se ela cabe ou não no objeto, não criando efeitos indesejados.

Será o gerador que irá ser encarregado de criar as coordenadas de textura. Para calcularmos as coordenadas de textura, começamos por dividir a imagem 2D em quadrados, com tantas linhas como stacks e colunas como slices. Através da nossa equação da esfera, conseguimos saber a ordem em que desenhamos cada um dos pontos da superfície da esfera logo, para sabermos os pontos de textura de uma certa stack e de uma certa slice, basta desenharmos o quadrado ilustrado acima, seguindo a mesmo ordem dos pontos da esfera. Estes passos todos descritos anteriormente são feitos na função *drawSphere* do nosso gerador. Nesta função além de calcularmos os pontos para o desenho da esfera, calculamos em simultâneo os pontos de textura, do seguinte modo, sendo i o iterador do ciclo exterior e j o iterador do ciclo interior, e **aux_T** um vector de *double*:

```
aux_T.push_back((j+1.0f)/stacks);
aux_T.push_back((i+1.0f)/slices);
aux_T.push_back((j+1.0f)/stacks);
aux_T.push_back((i*1.0f)/slices);
aux_T.push_back((j*1.0f)/stacks);
aux_T.push_back((i+1.0f)/slices);
aux_T.push_back((j+1.0f)/stacks);
aux_T.push_back((i*1.0f)/slices);
aux_T.push_back((j*1.0f)/stacks);
aux_T.push_back((i*1.0f)/slices);
aux_T.push_back((j*1.0f)/stacks);
aux_T.push_back((i+1.0f)/slices);
```

A função *drawSphere* devolve um *map*, que contem todos os pontos de textura e os pontos da esfera.

A partir da criação destas coordenadas, o nosso motor já está pronto para criar a nossa textura através de várias funcionalidades do OpenGL que fomos aprendendo durante as aulas.

2.2 Função read_File3D

Esta função sofreu algumas alterações, agora para além de ler os pontos para desenhar a figura,lê também as normais, pontos que servem para ser usados no calculo da luz, e lê também os pontos da textura.

```
    while(i<(n)){
        input_file >> a >> b >> c;
                    m->pontos.push_back(a);i++;
                    m->pontos.push_back(b);i++;
                    m->pontos.push_back(c);i++;

    }
    while(j<(n)){
        input_file >> a >> b >> c;
                    m->normal.push_back(a);j++;
                    m->normal.push_back(b);j++;
                    m->normal.push_back(c);j++;

    }
```

```
while(t<(texturas)){
    input_file >> a >> b;
                m->textura.push_back(a);t++;
                m->textura.push_back(b);t++;

}
```

2.3 Funções drawFigures

Para os desenhar fazemos uso das funções glBindBuffer() e glVertexPointer() para os pontos e glTexCoordPointer() para as texturas. Por fim, para desenhar a figura fazemos glDrawArrays(). Temos no entanto de, antes de desenhar a figura com a função glDrawArrays, chamar glBindTexture(GL_TEXTURE_2D, texID) e depois de desenhar glBindTexture(GL_TEXTURE_2D, 0).

```
    glBindBuffer(GL_ARRAY_BUFFER,p->mod[i]->buffer);//activar o vbo
        glVertexPointer(3,GL_FLOAT,0,0);
        glBindBuffer(GL_ARRAY_BUFFER,p->mod[i]->textCoord);
        glTexCoordPointer(2, GL_FLOAT,0,0);
        glBindTexture(GL_TEXTURE_2D,p->mod[i]->textId);

        float white[4] = {1,0,0,0};
        glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, white);
        glDrawArrays(GL_TRIANGLES,0,(p->mod[i]->numeroVertices)/3);
        glBindTexture(GL_TEXTURE_2D,0);
        }
```

2.4 Função groupToVBO

A nossa função grouptoVBO, que está representada de seguida, sofreu uma alteração relativamente à fase 3, tendo como objetivo principal a abertura da imagem que irá corresponder às texturas que estão contidas em cada figura.

```
unsigned int t,tw,th;
unsigned char *texData;

ilInit();
ilEnable(IL_ORIGIN_SET);
ilOriginFunc(IL_ORIGIN_LOWER_LEFT);
ilGenImages(1,&t);
ilBindImage(t);
ilLoadImage((ILstring)(m->img));
tw = ilGetInteger(IL_IMAGE_WIDTH);
th = ilGetInteger(IL_IMAGE_HEIGHT);
ilConvertImage(IL_RGBA, IL_UNSIGNED_BYTE);
texData = ilGetData();
```

```
glGenTextures(1,&(m->textId));

glBindTexture(GL_TEXTURE_2D,m->textId);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,GL_REPEAT);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);

glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, tw, th, 0, GL_RGBA, GL_UNSIGNED_BYTE, texData);
}
```

Capítulo 3

Iluminação / Normais

Infelizmente, não conseguimos estruturar uma solução válida para este problema. Porém o plano para este problema seria a definição de normais que nos permitissem determinar o ângulo de partes do modelo em relação á fonte luminosa tendo assim um efeito mais realista e seriamos assim capazes de iluminar apenas segmentos específicos de uma figura e não a sua totalidade, caso relativo à não utilização de normais.
Para indicação do local da fonte de luz adicionaríamos um novo elemento ao nosso ficheiro de configuração xml:

```
<!--<Ligths>
        <ligth tipo="POINT" posX=0 posY=0 posZ=0 />
        <ligth tipo="DIRECIONAL" posX=0 posY=0 posZ=0 />
</Ligths>-->
```

Figura 3.1: XML adicionando a luz

Neste novo elemento somos capazes de identificar o tipo de iluminação e o seu ponto de origem, definido pelas coordenadas X,Y e Z.
No caso do sistema solar iniciaríamos a posição da luz como x=0, y=0 e z=0, sendo este ponto representativo do nosso foco luminoso,o Sol, tem o seu centro.Sendo assim simulamos a fonte de luz vinda do sol que iluminaria os restantes astros pelo ângulo que seria esperado.

Capítulo 4

Sistema Solar

Depois de efetuadas todas as nossas transformações, o resultado final será um Sistema Solar dinâmico em que cada planeta tem a sua própria órbita (ou seja gira em torno do Sol),gira em volta de si mesmo e tem a si associado a sua própria textura. Fizemos uma alteração relativamente ao cometa passando agora a ser uma esfera em vez de um teapot e encontra-se também em órbita.

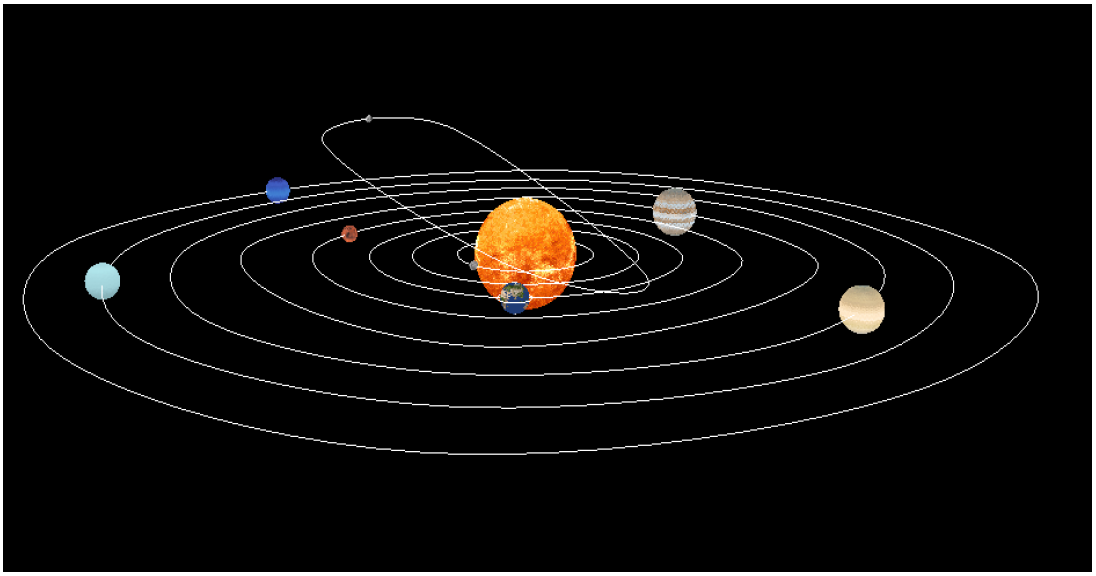

Figura 4.1: Sistema Solar

Capítulo 5

Conclusão

Consideramos que esta fase se dividiu essencialmente em três etapas chave:

- Leitura dos novos parâmetros do XML para a nossa estrutura de dados;
- Cálculo e utilização dos pontos de textura;
- Cálculo e utilização das normais para cálculo da luz;

Infelizmente esta ultima fase foi a par da fase 3 a que consideramos mais complicada, encontramos diversas dificuldades no calculo dos pontos de textura o que não nos permitiu avançar de forma concisa e ágil entre os diversos objetivos deste projeto em particular.
Relativamente à iluminação, outro dos objetivos do projeto, devido a vários fatores externos e também a fatores intrínsecos ao trabalho como a alteração da estrutura dados em que estes serão armazenados e a leitura do próprio ficheiro XML que seria agora caracterizado por novos atributos sendo assim obrigatório uma nova leitura o que nos causa alguns constrangimentos.

Resumidamente, conseguimos realizar a adaptação da nossa estrutura para englobar as texturas porém a sua expressão visual, possivelmente por alguma troca nos valores dos índices relativos as Stacks e Slices e compreendemos que existe bastante margem para melhorar nesse ponto, é ainda com bastante pesar que não conseguimos entregar algo mais sólido relativamente à iluminação e nos mantivemos apenas por um pensamento teórico e não pelo seu investimento pratico mais por factores externos como a falta de tempo causado pelo acumular de entregas características desta etapa do semestre sendo que com mais tempo julgamo-nos capazes de finalizar o que era pretendido nesta fase.

Bibliografia

[1] Documentação fornecida pelo professor

[2] http://www.grinninglizard.com/tinyxmldocs/annotated.html

[3] http://www.grinninglizard.com/tinyxml2/

[4] https://www.mvps.org/directx/articles/catmull/

[5] http://www.cplusplus.com

[6] https://en.wikipedia.org/wiki/VertexBufferObject

[7] http://joshworth.com/dev/pixelspace/pixelspace_solarsystem.html

[8] http://www.songho.ca/opengl/glvbo.html

[9] https://web.cs.wpi.edu/ matt/courses/cs563/talks/surface/bez_surf.html