Universidade do Minho

Departamento de Informática

Computação Gráfica

Phase 4 – Normais e Coordenadas de Textura

Grupo 39

Gonçalo Araújo Brandão A100663
Maya Gomes A100822
Luís de Castro Rodrigues Caetano A100893

26 de maio de 2024

Conteúdos

1 Introdução

A produção dos modelos deste trabalho prático em OpenGL envolve diversos temas, entre os quais, transformações geométricas, curvas, superfícies cúbicas, iluminação e texturas. Neste relatório da quarta fase do projeto serão relatados os procedimentos da mesma, sendo mais orientada para as Normais e Coordenadas de Textura. Para isso teremos de ativar as funcionalidades referentes à iluminação e textura, assim como ler e aplicar as normais e coordenadas de textura de ficheiros modelo.

Nesta fase, o objetivo fazer com que o Gerador que desenvolvemos até esta fase seja capaz de calcular as normais e coordenadas das diversas primitivas e que o a nossa Engine tire proveito desses parâmetros para aplicar a iluminação e as texturas.

2 Descrição do problema

Nesta quarta fase do projeto, devemos então implementar as novas funcionalidades no Gerador e na nossa Engine: calcular vetores normais e as coordenadas de textura para cada vértice e ler características seja de textura e iluminação do ficheiro XML, assim como aplicá-las, respetivamente a cada parte. Para este efeito, os nossos ficheiros XML, que irão ser lidos pela Engine, devem permitir a definição da cor difusa, especular, emissiva e ambiente, bem como brilho. Deve também ser possível permitir a definição das fontes de luz.

3 Estrutura da aplicação

3.1 Resolução do problema

3.1.1 Novas estruturas

Nesta fase do trabalho, foi necessário a adaptação da nossa struct "Point"de forma a que para além das coordenadas do ponto sejam também armazenados os três pontos para a normal e dois valores para as coordenadas da textura.

```
struct Point {
    float x, y, z;
    float nx, ny, nz;
    float tx, tz;
};
```

Figura 1: Estrutura *Point*.

Para além disso, no Engine alteramos algumas das nossas estruturas, adiciononando o Point2 para os pontos e as normas nos VBOS e a Point3 para lidar com as coordenadas das textutras:

```
struct Point2 {
    float x, y, z;
};

struct Point3 {
    float x, y;
};
```

Figura 2: Estrutura *Point2* e *Point3*.

Estas duas estruturas de pontos foram criadas de forma a facilitar as nossas interações com as normas (Point2) e com as texturas (Point3).

Para as cores, foram também necessárias *structs*:

```
struct Color
{
    float R, G, B;
    Color(float R = 1.0f, float G = 1.0f, float B = 1.0f) : R(R), G(G), B(B) {}
};
```

Figura 3: Estrutura *Color*.

```
struct ColorProperties
{
    Color diffuse;
    Color ambient;
    Color specular;
    Color emissive;
    float shininess;

    ColorProperties() : diffuse( R: 200,  G: 200,  B: 200), ambient( R: 50,  G: 50,  B: 0),
                        specular( R: 0,  G: 0,  B: 0), emissive( R: 255,  G: 0,  B: 0), shininess(0) {}
};
```

Figura 4: Estrutura *ColorProperties*.

Para a luz, criamos as seguintes estruturas:

```
struct Light {
    std::string type;
    PointLight position;
    PointLight direction;
    float cutoff;
```

Figura 5: Estrutura *Light*.

```
struct PointLight {
    float x, y, z;
    float position[3];
    PointLight() : x(0.0f), y(0.0f), z(0.0f) {
        position[0] = x;
        position[1] = y;
        position[2] = z;
    }
};
```

Figura 6: Estrutura *PointLight*.

Inicialmente, percorremos a lista de luzes no mundo 3D e configuramos as suas propriedades usando funções do OpenGL como glLightfv e glLightf. Definimos as componentes de iluminação ambiente, difusa e especular para cada luz, e ajustamos as suas posições e direções dependendo do tipo (pontual, direcional ou spot). As luzes pontuais emitem em todas as direções a partir de uma posição específica, as luzes direcionais simulam uma fonte distante como o sol e as luzes spot são direcionadas com um ângulo específico, semelhante à de um holofote.

Depois, no initGL inicializamos o estado do OpenGL configurando capacidades necessárias para renderização 3D. Ativámos os arrays de vértices, normais e coordenadas de textura, habilitamos o teste de profundidade (GL_DEPTH_TEST) para renderização correta em 3D e o face culling (GL_CULL_FACE) para melhorar o desempenho. Também configurámos a iluminação, ativando até oito fontes de luz (GL_LIGHT0 a GL_LIGHT7), e habilitámos a normalização automática de normais (GL_RESCALE_NORMAL), tal como o uso de texturas 2D (GL_TEXTURE_2D). Estas configurações preparam o ambiente de renderização para desenhar cenas 3D com iluminação e texturização apropriadas

Também criámos uma estrutura denominada ModelData com o objetivo de de conseguirmos guardar os nossos pontos com as suas normais e coordenadas de texturas, de forma a utulizá-los na construção dos nossos VBO's na funcao initVBO. Esta associa 3 buffers, para vértices, normais e coordenadas de textura, cada um para o tipo de vector. Estes buffers são associados com o glBindBuffer e preenchidos com os dados apropriados através de glBufferData. Estes pontos vértice, normais e textura do modelo, armazenados em modelData, são transferidos para o GPU como buffers estáticos. Após a transferência, a função armazena os identificadores dos buffers no array buffers e desassocia o buffer atual com glBindBuffer(GL_ARRAY_BUFFER, 0).

Finalmente, foi também necessário alterarmos o nosso *World*, acrescentando a modelData e a lights:

```
struct ModelData{
    std::vector<Point2> vertexPoints;
    std::vector<Point2> normalPoints;
    std::vector<Point3> texturePoints;
    int vertexCount ;
};
```

Figura 7: Estrutura *ModelData*.

```
struct World {
    Camera camera;
    vector<Group> groups;
    map<string, vector<Point>> points;
    map<string, ModelData> modelData;
    vector<string> files;
    vector<Light> lights;
    std::vector<CatmullCurve> catmullCurves;
};
```

Figura 8: Estrutura *World*.

3.2 Generator

De forma a aplicármos as novas funcionalidades, foi necessário alterar o conteúdo dos nossos ficheiros .3d. Desta forma, cada linha do ficheiro deve agora conter três pontos (coordenadas), três pontos para a normal e dois valores para as coordenadas da textura.

Plano No caso do plano, como este último está virado para cima, bastou verificar que a normal de cada ponto seria (0, 1, 0). Sobre as texturas, temos que a imagem se vai repetindo ao longo da grelha do plano.

Cone No caso do cone, mais uma vez, iremos dividir o cone em dois. Divide-se as normais em dois segmentos: a base do cone e a superfície lateral. Quanto à base, todos os pontos têm a normal como (0, -1, 0). No entanto, não conseguimos a implementação totalmente correta do cone, mas para efeitos de teste usou-se um xml funcional.

Caixa Relativamente à caixa, sendo que esta é formada por 6 planos, o número de vetores das normais será 6 (sendo um para cada face). Além disso, duas faces são perpendiculares com o plano xy. As outras duas são com o plano yz e as restantes são com o plano xz. Assim, a única diferença de cada "par"de faces perpendicular com qualquer plano é que as faces possuem direções opostas. Desta forma, concluimos que as normais serão (0, 0, 1), (0, 0, -1) para as faces de frente e de trás e (0, 1, 0) para as faces de cima e de baixo. As faxes laterais terão (0, -1, 0) e (1, 0, 0), (-1, 0, 0).

Esfera Na esfera, cada ponto é calculado iterativamente com base nas coordenadas esféricas. As normais de cada ponto são derivadas diretamente das coordenadas esféricas, indicando a direção radial em relação ao centro da esfera. Cada ponto possui coordenadas tridimensionais, bem como coordenadas de textura para mapeamento.

É importante referir que no resto do trabalho, ou seja, no Engine, utilizamos dois ficheiros .3d (cone e bezier) que não foram gerados por nós. Como não conseguiamos gerá-los corretamente, prosseguímos para o resto do trabalho utilizando os ficheiros fornecidos por uns colegas (de outro grupo) para o efeito de teste e, consequentemente, conseguirmos visualizar se o funcionamento da nossa Engine, em geral, estava bem.

3.3 Engine

Na parte do Engine, como referido anteriormente, definiram-se a cor difusa, especular, emissiva e o ambiente, brilho e fontes de luz, sendo necessário alterar o *parser* do ficheiro XML.

A estrutura World possui agora um vetor de Light.

É tambem importante referir que de modo a suportar as definições de luzes, (depois de ler o ficheiro XML) as opções de luz são ativadas.

Além disso, na função de renderização, a seguir à criação da câmara, ativa-se cada luz, isto é, para cada elemento da lista de Light, dependendo do tipo, define-se os parâmetros de fonte de luz.

3.3.1 Resultados

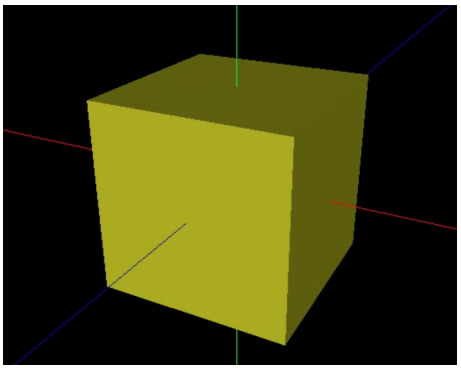

Figura 9: Resultados teste 1.

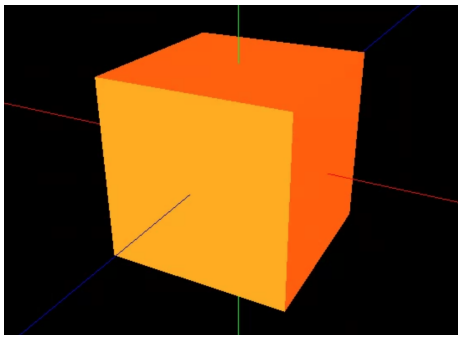

Figura 10: Resultados teste 2.

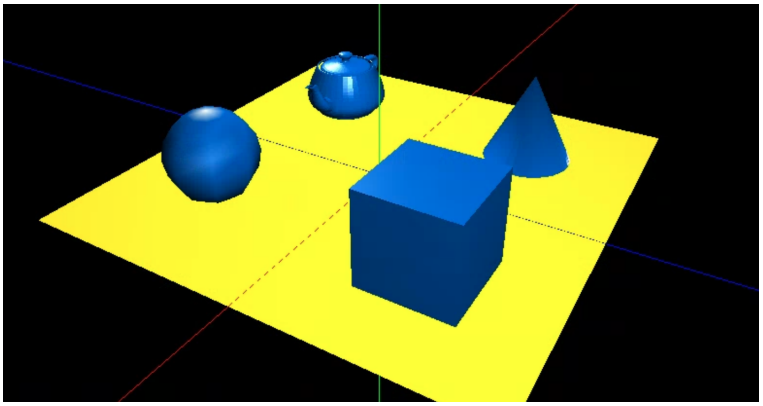

Figura 11: Resultados teste 3.

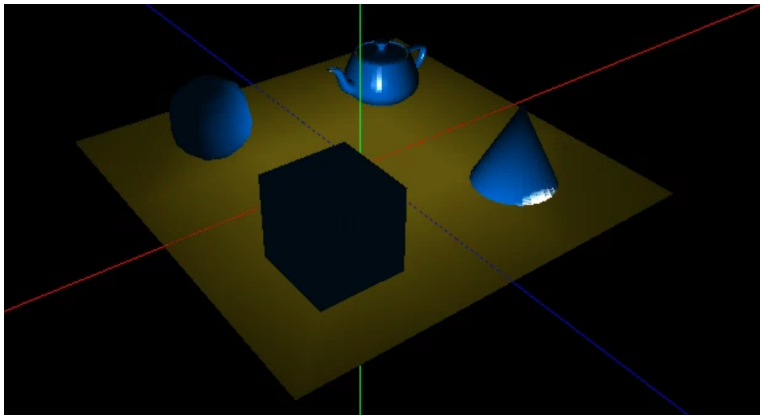

Figura 12: Resultados teste 4.

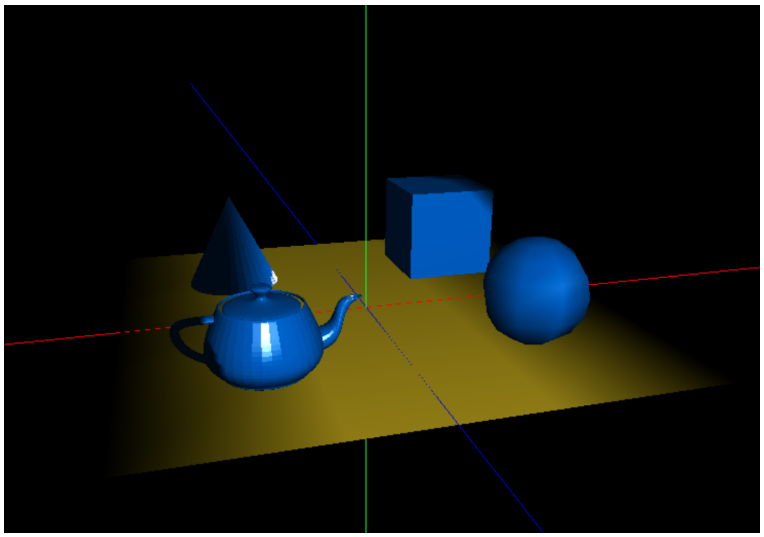

Figura 13: Resultados teste 5.

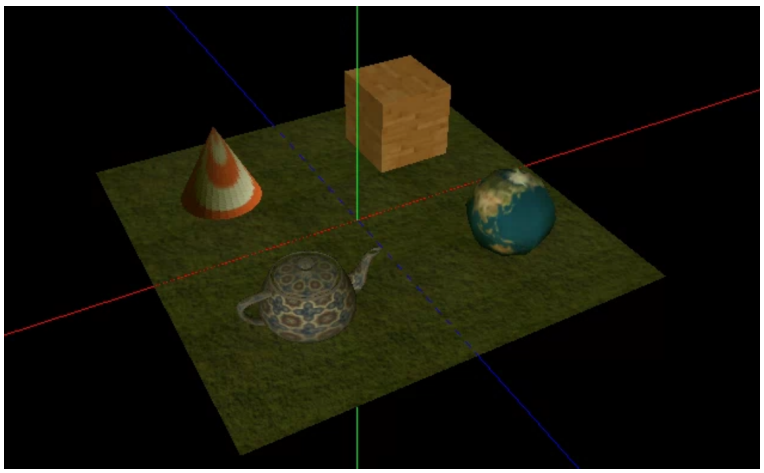

Figura 14: Resultados teste 6.

3.4 Sistema Solar

Para o Sistema Solar implementámos texturas aos vários planetas, obtidas pela documentação obtida pela equipa docente. No que toca a iluminação, utilizou-se uma fonte de luz do tipo POINT posicionada na origem. Desta forma, a luz comporta-se como o sol, sendo emitida em todas às direções. Não utilizamos o *directional* pois seria apenas numa so direção, nem o *spot* pois seria numa direção com ângulo "aberto"tendo o mesmo problema que o anterior.

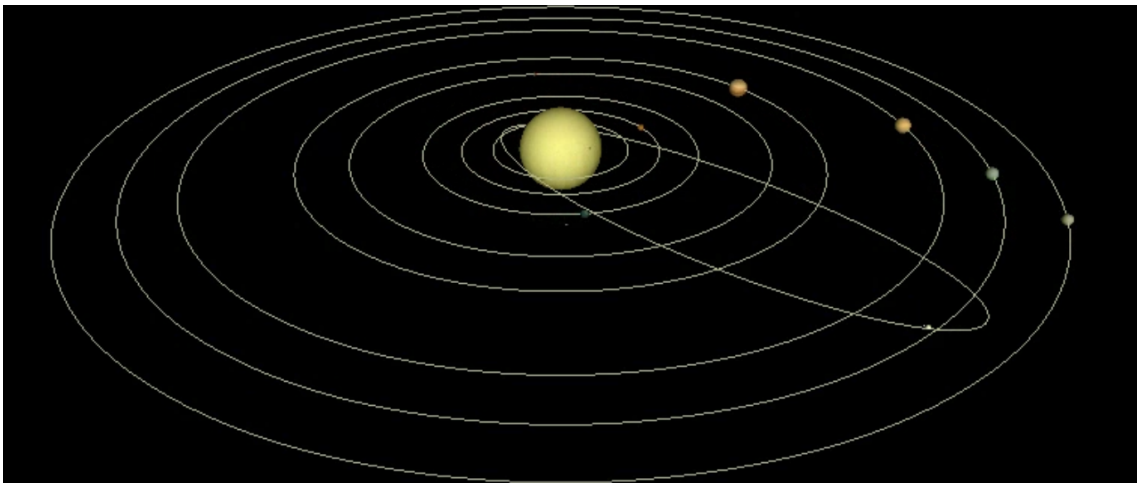

Figura 15: Resultados sistema solar.

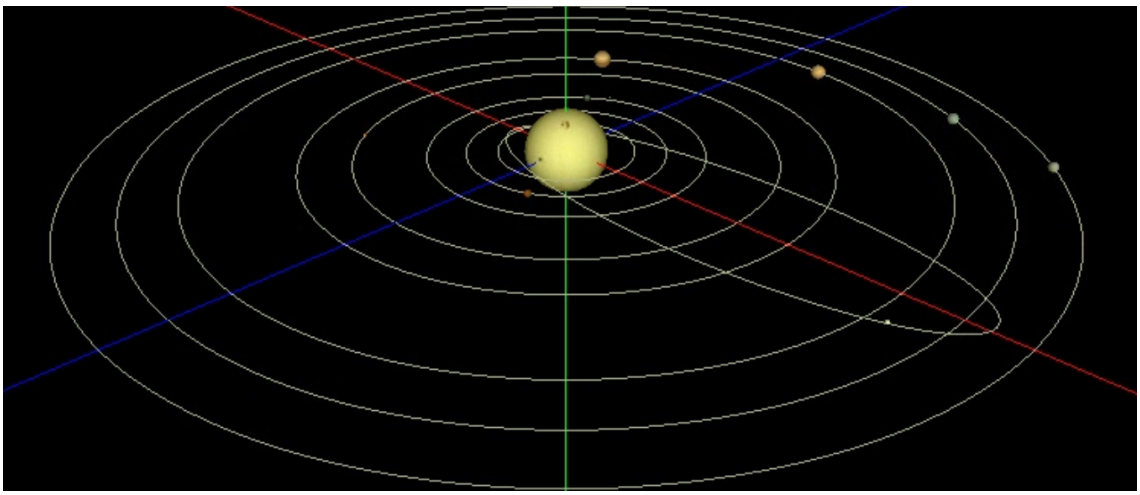

Figura 16: Resultados sistema solar com eixos.

4 Conclusão

Nesta fase não conseguimos implementar corretamente o cone e o bezier devido a má gestão de tempo. Decidimos no entanto avançar e prosseguir com o trabalho testando sempre com ficheiros .3d corretos.

Sobre o trabalho em geral, tínhamos como objetivo a melhoria de alguns parâmetros, como os da câmera, que ficaram para esta fase, mas que não conseguimos concluir, pois preferimos focar na realização das novas propostas (da fase 4).

No que toca à execução dos testes, conseguimos cumprir com os objetivos, sendo que esses funcionam como pretendido. O sistema solar final também está funcional com as suas texturas e a sua iluminação.

Este trabalho revelou-se um grande desafio para o grupo, pois apesar de certas fases serem mais acessíveis, algumas, nomeadamente as duas últimas, foram bastante trabalhosas. Surgiram muitas dificuldades, mas praticamente todas, foram ultrapassadas. Também concluímos que as diversas matérias lecionadas foram devidamente aprofundadas com a concretização dos diferentes problemas que nos foram sendo propostos ao longo das diversas fases.

Desta forma, concluímos este trabalho com um balanço positivo, pois acreditamos ter cumprido com as espectativas da equipa docente.