Computação Gráfica
(3º ano de LCC)

Trabalho Prático - Parte 4

Relatório de Desenvolvimento

Francisco Oliveira	Luís Costa	Maria Luísa Silva	Tierri Monteiro
(A82066)	(A70070)	(A82124)	(A76359)

1 de Junho de 2020

Conteúdo

Capítulo 1

Introdução

1.1 Contextualização

O presente relatório foi elaborado no âmbito da Quarta Fase do Trabalho Prático da Unidade Curricular de Computação Gráfica, que se insere no 2º semestre do 3º ano de Ciências da Computação.

Nesta quarta fase foi proposta a geração de modelos com coordenadas de textura e normais para vértice, sendo que o motor gráfico deverá ser capaz de ler e interpretar essas coordenadas, ativando assim as funcionalidades de iluminação e textura. Para além disso, também deverá ser possível definir cores e fontes de luz no ficheiro de configuração da cena.

1.2 Resumo

Uma vez que todos os modelos deverão passar a ter texturas e normais, todas as primitivas terão de ser alteradas para se poder gerar as respetivas coordenadas. No entanto, devida a uma má optimização da nossa parte na geração das figuras iremos também mudar a forma como elas são geradas.

Capítulo 2

Gerador

Ao fazer pesquisa para gerar coordenadas para a textura e normais de cada vértice reparamos que o nosso código de gerar índices era pouco otimizado, por isso ,decidimos mudar a maneira como gerávamos os modelos e a criação dos seus índices.

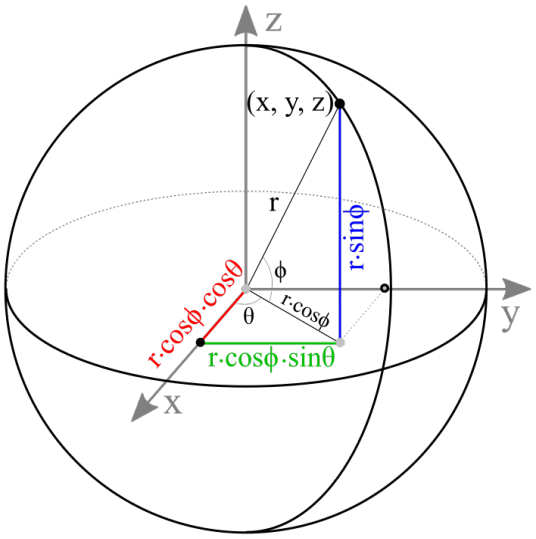
2.1 Sphere

2.1.1 Gerar Esfera

A nova forma de calcular os pontos é a que podemos observar nas figura 2.1 .

$$
\begin{aligned}
x &= (r \cdot \cos\phi) \cdot \cos\theta \\
y &= (r \cdot \cos\phi) \cdot \sin\theta \\
z &= r \cdot \sin\phi \\
\theta &= 2\pi \cdot \frac{\text{slice}}{\text{numslices}} \\
\phi &= \frac{\pi}{2} - \pi \cdot \frac{\text{stack}}{\text{numstacks}}
\end{aligned}
$$

(a) Calculo das coordenadas da esfera



(b) Representação do calculo

Figura 2.1: Geração dos pontos da Esfera

2.1.2 Indices

De forma a desenhar a superfície da esfera temos de triangular os vértices adjacentes dos polígonos. Cada setor requer 2 triângulos no primeiro índice do vértice a stack atual é k1 a seguinte k2 e a próxima sempre em sentido contrário do relógio. K1 $\rightarrow k2 \rightarrow k1 + 1.k1 + 1 \rightarrow k2 \rightarrow k2 + 1$.

No entanto a parte de cima e baixo apenas requer um triangulo por slice.

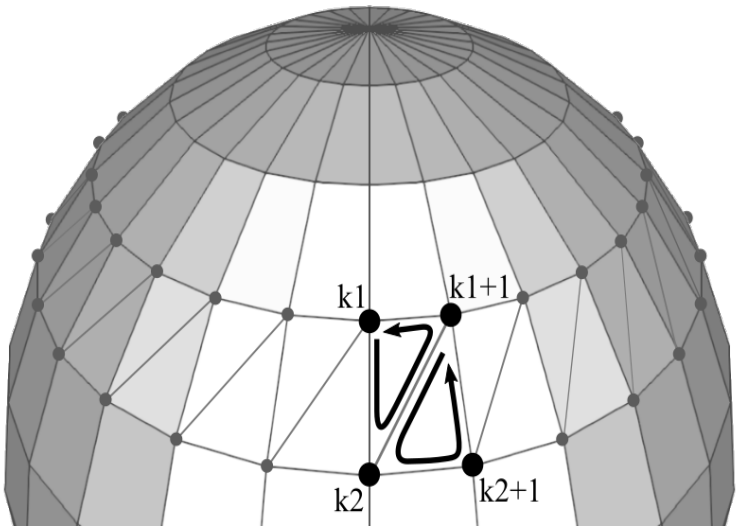


Figura 2.2: Calculo dos indices

2.1.3 Texturas e normais

Quanto às texturas e normais da da esfera, estas são facilmente obtidas através da normalização dos componentes x, y e z de cada vértice, isto é, dividindo o seu valor pelo raio. Quanto às coordenadas de textura, para cada vértice a sua textura será ((slices - slice) / slices, (stacks - stack) / stacks).

2.2 Plano

2.2.1 Índices

Quanto aos índices do plano apenas foi necessario guardar os dois triângulos que o formam , ou seja C $\rightarrow B \rightarrow A, B \rightarrow C \rightarrow D$.

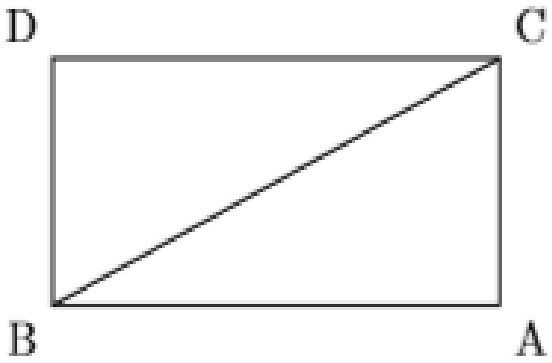


Figura 2.3: Cálculo dos indices do plano

2.2.2 Normais e Texturas

As normais do plano são obtidas de forma muito simples, uma vez que são iguais para os 4 vértices(0,1,0). Quanto às coordenadas de textura, para cada extremidade do plano, corresponde uma extremidade da textura a aplicar, ou seja , (1,1), (0,0), (0,1), (1,0).

2.3 Caixa

2.3.1 Normais e Texturas

Quanto há caixa devido à falta de tempo não foi possivel optimizar o código anterior e apenas colocamos as texturas e as respetivas normais sendo que as suas normais são (0,1,0) para a face de cima , (0,-1,0) para a face de baixo, (-1,0,0) para a face da esquerda, (1,0,0) para a face da direita, (0,0,1) para a face da frente e (0,0,-1) para a face de trás.

Já relativamente às texturas cada vértice (x,y) irá corresponder à coordenada (x/divisions) na textura.

2.4 Cone

2.4.1 Gerar Cone

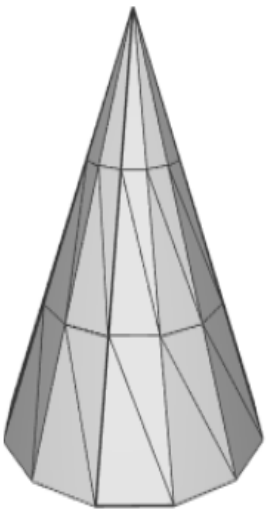
Para a geração do cone dividimos em duas partes , a base e a face lateral. Quanto ao calculo da base utilizamos o método representado na figura 2.4. Optamos para colocar a figura centrada ou seja ter como extremos em y height/2 e -height/2.

```
// base
float step_angle = (2 * M_PI) / slices;
float angle = 0.0f;
float heightc = height / 2;
// base
vertices.push_back(Vertice(0.f, -heightc, 0.f));
for(int slice = 1; slice <= slices; slice++) {

    float x1 = radius * sin(angle + step_angle);
    float y1 = -heightc;
    float z1 = radius * cos(angle + step_angle);
```

Figura 2.4: Cálculo da base do cone

Quanto à face lateral utilizamos o seguinte :



```
for (int slice = 0; slice < slice; slice++) {
    for (int stack = 0; stack < stacks + 1; stack++) {


        float div = stack / stacks;
        float alpha = (1.0 - (div)) * radius;

        float px = alpha * cos(angle + step_angle);
        float py = stack * div - heightc;
        float pz = alpha * sin(angle + step_angle);
```

Figura 2.5: Geração dos pontos da face lateral do cone

Quando div = 1 , ou seja, chega a última stack irá apenas gerar um triangulo em cada slice na parte superior do cone.

2.4.2 Indices

Tal como nos pontos , estará dividido em duas partes , base e face lateral. Na base irá existir os seguintes índices :

```
for (int slice = 0; slice < slices; slice++) {
    indice.push_back(0);
    indice.push_back(slice*2+2);
    indice.push_back(slice*2+1);

}
```

Figura 2.6: Indice da base do cone

E na parte lateral :

```
for (int slice = 0; slice < slices; slice++) {
    for (int stack = 0; stack < stacks; stacks++) {
        indice.push_back((slices + 1) * 2 + slice * (slices + 1) + stack);
        indice.push_back((slices + 1) * 2 + slice * (slices + 1) + stack + 1);
        indice.push_back((slices + 1) * 2 + (slice + 1) * (slices + 1) + stack);

        indice.push_back((slices + 1) * 2 + (slice + 1) * (slices + 1) + stack);
        indice.push_back((slices + 1) * 2 + slice * (slices + 1) + stack + 1);
        indice.push_back((slices + 1) * 2 + (slice + 1) * (slices + 1) + stack + 1);

    }
}
```

Figura 2.7: Indice da base do cone

Ou seja , vai escrever o ponto 1 , 2 , 3, 2, 4 , 3 como demonstrado na figura 2.8.

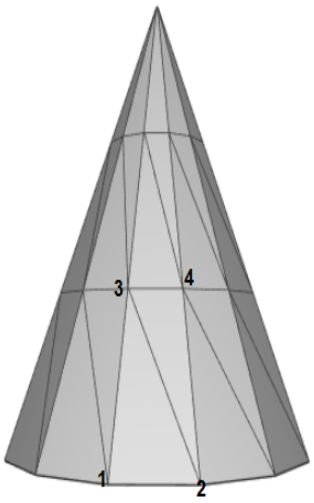


Figura 2.8: Indice da base do cone demonstração

2.4.3 Normais e Texturas

A normal de todos os vértices da base do cone é (0,-1,0). Já nas texturas cada vértice da base (alturaBase ,slice) é associada a coordenada da textura (0, 1) caso do vertice do centro e (slice / slices,0) no caso dos outros .

Calculado a componente Y que é constante como podemos observar pela figura 2.9 o resto das coordenadas são calculadas usando as equações da figura 2.10

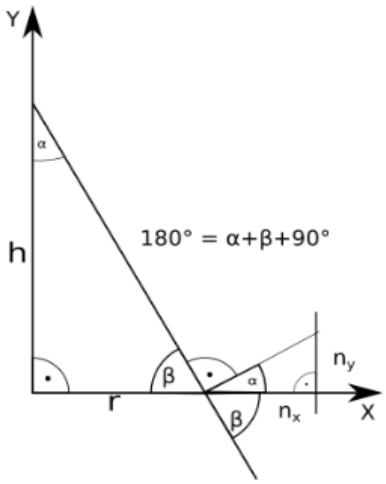


Figura 2.9: Cálculo de y normal da superfície lateral do cone

```
float nx = cos(angle + step_angle);
float ny = sin(M_PI - atan(height / radius));
float nz = cos(angle + step_angle);
```

Figura 2.10: equações para obter x,y,z

Quanto às coordenadas de textura, estas têm valor (slice/ slices, stack / stacks).

Capítulo 3

Engine

O Engine é responsável pelo armazenamento da informação dos modelos a representar, assim como as respetivas transformações geométricas de cada um desses modelos, que no seu conjunto, irão formar o sistema solar que queremos representar. Esta também resultou na implementação de novas características no motor do modelo. Findando o processo de leitura do ficheiro, passamos para a fase seguinte, a qual é responsável pela renderização da informação, previamente obtida.

3.1 VBOs e Texturas

Em paralelo com a fase anterior continuamos a fazer uso de VBOs (Vertex Buffer Object).
Com a implementação de textura e luz tornou-se necessário criar mais 2 VBOs, um com as normais e outra com a textura dos pontos.

3.2 Iluminação

A iluminaçãao dos modelos gerados no projecto é corretamente obtida através do cálculo das várias normais do mesmo, sendo que é através destas que se pode obter a intensidade da luz.

Capítulo 4

Resultados Obtidos

Nesta parte podemos visualizar o resultado da criação do Sistema Solar. Infelizmente o resultado obtido não foi o esperado. Na imagem seguinte conseguimos ver as órbitas escurecidas devido á iluminação no entanto os planetas nao estão a ser renderizados corretamente.

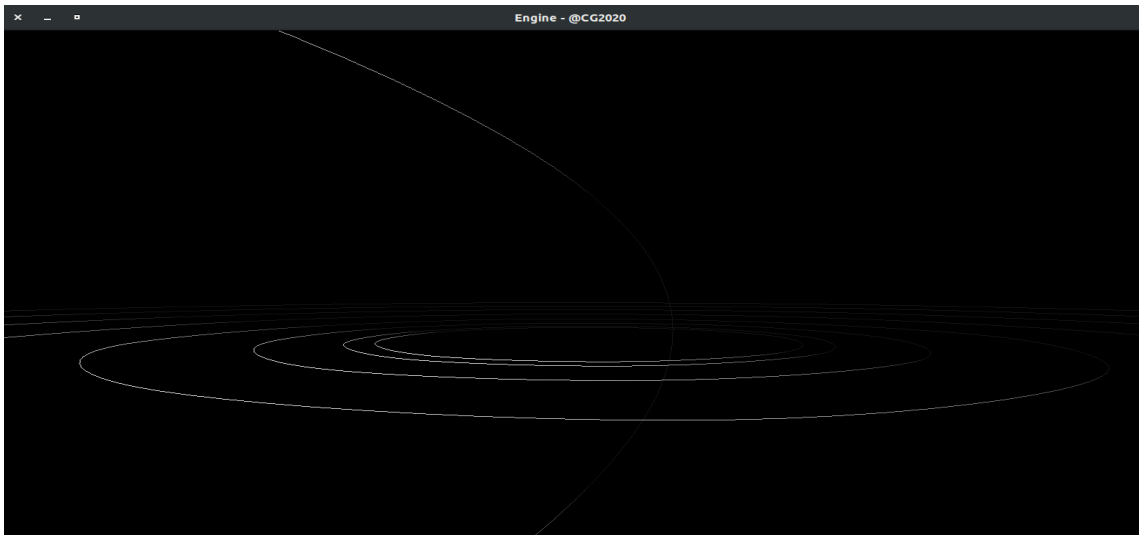


Figura 4.1: Resultado final

Capítulo 5

Conclusão

Depois de concluído, olhamos para este projeto de forma positiva. Apesar das partes iniciais terem sido executadas com sucesso, exixtiram algumas dificuldades nas duas últimas fases do trablho, não conseguindo atingir o objetivo pretendido.

No entanto, apesar dos diversos problemas encontrados, com o desenvolvimento deste trabalho prático consideramos que o nosso conhecimento em relação à disciplina e àrea de Computação Gráfica foi enriquecido, bem como a utilização das diferentes ferramentas usadas, como por exemplo o *OpenGL* ou o *GLUT*.

Capítulo 6

Bibliografia

http://www.songho.ca/opengl