Universidade do Minho
Escola de Engenharia

COMPUTAÇÃO GRÁFICA

FASE 4

Alexandre Martins A77523

André Vieira A78322

Eduardo Rocha A77048

Ricardo Neves A78764

ÍNDICE

INTRODUÇÃO

Neste relatório iremos apresentar e discutir o trabalho realizado pelo grupo, no âmbito da Unidade Curricular de Computação Gráfica, do 3º ano do Mestrado Integrado em Engenharia Informática.

Nesta ultima fase do trabalho prático, tivemos como objetivo a atualização da cena dinâmica criada na fase anterior, adicionando texturas aos planetas do sistema, bem como a adição de luzes. De salientar que todos os modelos foram desenhados com recurso aos VBOs.

Portanto, neste relatório, iremos apresentar detalhadamente todo o processo realizado, de modo a cumprir os objetivos previamente estabelecidos pelo docente da Unidade Curricular.

Para isto, apresentamos também neste documento, algumas linhas de pensamento que o grupo seguiu, ferramentas usadas, e excertos de código fonte utilizado (e respetiva explicação), de modo a suportar a perceção do trabalho realizado.

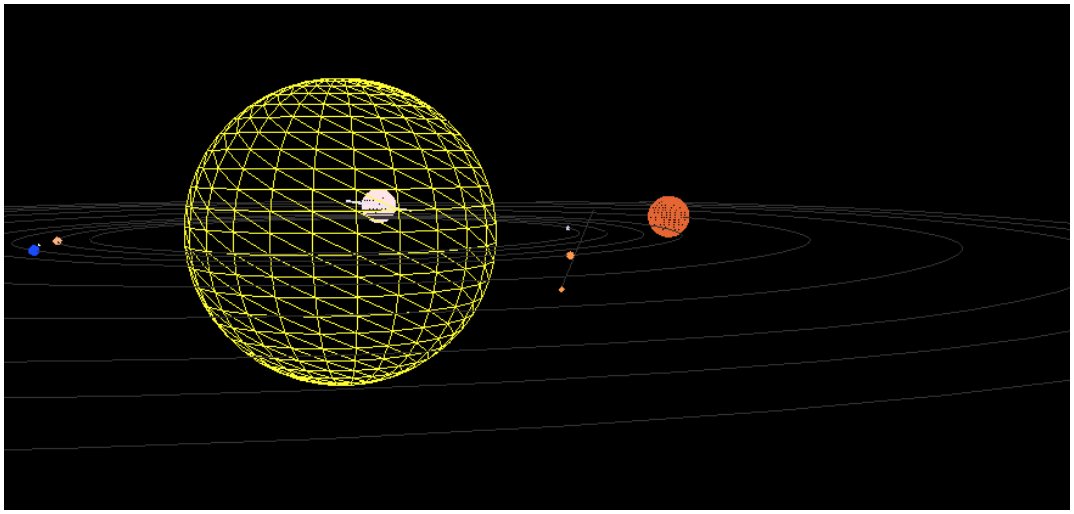

Figura 1 - Modelo da Fase 3

GENERATOR

Para a geração dos pontos para a textura das esferas, foi preciso calcular os pontos onde a textura é aplicada. Aqui, o espaço das texturas varia entre 0 e 1 em cada eixo.

```
double s;
double deltaS = 1.0f / slices;
double t;
double deltaT = 1.0f / stacks;
```

Figura 2 - Variaveis para calculo das coordenadas

Assim, utilizamos duas variáveis para a geração das coordenadas das texturas, que fazem uso das variáveis de ciclo i e j.

t = (stacks - i) * deltaT

s = j * deltas

Á medida do cálculo das coordenadas das texturas, estas iam sendo escritas no ficheiro sphere.3d. Na primeira posição, é escrito um vértice da esfera, na segunda, um vértice que representa as normais, e na terceira, um dos vértices da textura a ser aplicada ao volume.

```
file << p1x_S << " " << p1y_S << " " << p1z_S << "\n";
file << sin(beta) * sin(alpha) << " " << cos(beta) << " " << sin(beta) * cos(alpha) << "\n";
file << s << " " << t << " " << -1 << "\n";
```

Figura 3 - Escrita das coordenadas no ficheiro

Para a aplicação das luzes nos modelos, foi necessário calcular os vetores das normais. Sendo que uma normal é um vetor unitário cuja direção é perpendicular à superfície aplicada. Assim, é necessário calcular este mesmo angulo.

```
p1x_S = radius * sin(beta) * sin(alpha);
p1y_S = radius * cos(beta);
p1z_S = radius * sin(beta) * cos(alpha);
```

Figura 4 - Coordenadas dos vértices

PARSER DO XML

O parser do XML foi alterado de modo a acolher os elementos novos do XML como: as componentes da luz, o nome da textura e cores da luz para cada modelo.

Assim, para acolher o tipo e parâmetros da luz do sistema, criamos uma estrutura de dados simples que guardasse estes dados. Podemos ver essa estrutura em baixo:

```
typedef struct light{
    char* type;
    double params[3];
} Light;
```

Figura 5 - Estrutura de dados da luz

EXTRAÇÃO DAS COORDENADAS

O código de extração das coordenadas foi alterado de modo a guardar estas coordenadas novas. As coordenadas da esfera são guardadas num vetor, as coordenadas das normais no vetor das normais, e as coordenadas das texturas guardadas no vetor das texturas. Aqui, também utilizamos as coordenadas (12345, 12345, 12345), que significam que o modelo terminou.

OPENGL

A luz é adicionada na função addLight(), como o próprio nome indica. Aqui, verificamos qual o tipo de luz que é necessário aplicar ao Sistema Solar, através da comparação com a componente "type" da estrutura Light.

```
void addLight(){

    GLfloat pos[4] = {0.0, 0.0, 1.0, 0.0};
    GLfloat amb[4] = {0.2, 0.2, 0.2, 1.0};
    GLfloat diff[4] = {1.0, 1.0, 1.0, 1.0};
    GLfloat spotDir[3] = {0.0, 0.0, -1.0};

    Light* l = lights.front();

    pos[0] = static_cast<GLfloat>(l->params[0]);
    pos[1] = static_cast<GLfloat>(l->params[1]);
    pos[2] = static_cast<GLfloat>(l->params[2]);
    pos[3] = 1;

    if(strcmp(l->type, "POINT") == 0){

        glEnable(GL_LIGHT0);
        glLightfv(GL_LIGHT0, GL_POSITION, pos);
        glLightfv(GL_LIGHT0, GL_AMBIENT, amb);
        glLightfv(GL_LIGHT0, GL_DIFFUSE, diff);

    }
```

Figura 6 - Função de adição da luz

Como podemos ver acima, foram aplicadas as funções definidas no OpenGL necessárias para a adição de luz ao sistema, dependendo, como é obvio, do seu tipo.

Agora, temos de tratar das luzes individuais de cada modelo. Esta aplicação de cor é efetuada na função addColor(), que trata de verificar o tipo da luz e aplica-la.

```
void addColor(Color* color)
{

    GLfloat amb[4] = {0.2, 0.2, 0.2, 1.0};
    GLfloat dif[4] = {1.0, 1.0, 1.0, 1.0};
    GLfloat spec[4] = {0.0, 0.0, 0.0, 1.0};
    GLfloat emis[4] = {0.0, 0.0, 0.0, 1.0};

    if (strcmp(color->type, "diffuse") == 0) {
        dif[0] = static_cast<GLfloat>(color->colors[0]);
        dif[1] = static_cast<GLfloat>(color->colors[1]);
        dif[2] = static_cast<GLfloat>(color->colors[2]);
        dif[3] = 1;

        glMaterialfv(GL_FRONT, GL_DIFFUSE, dif);
    }
```

Figura 7 - Função de adição de luz a um modelo

O resultado final no planeta Jupiter está disponível na próxima imagem, onde podemos ver a presença da luz proveniente do Sol:

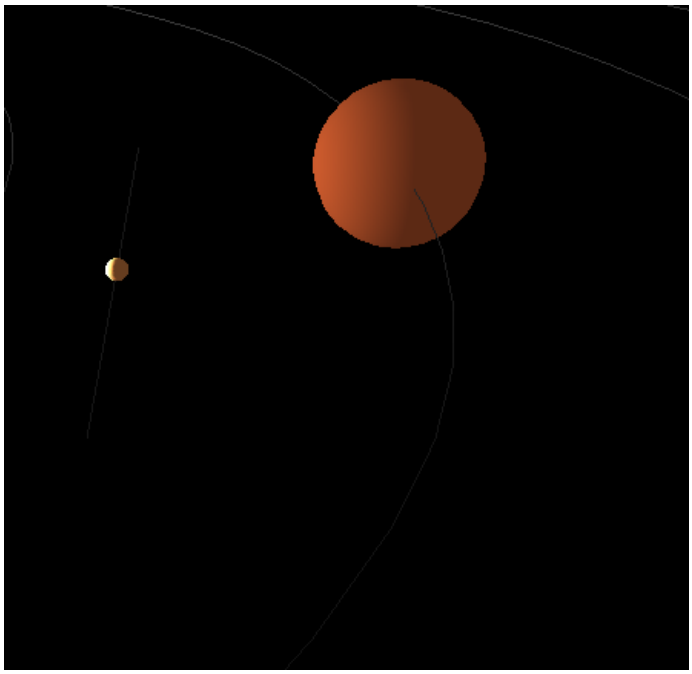

Figura 8 - Imagem de Jupiter

SISTEMA SOLAR FINAL

Este foi o resultado final do trabalho realizado:

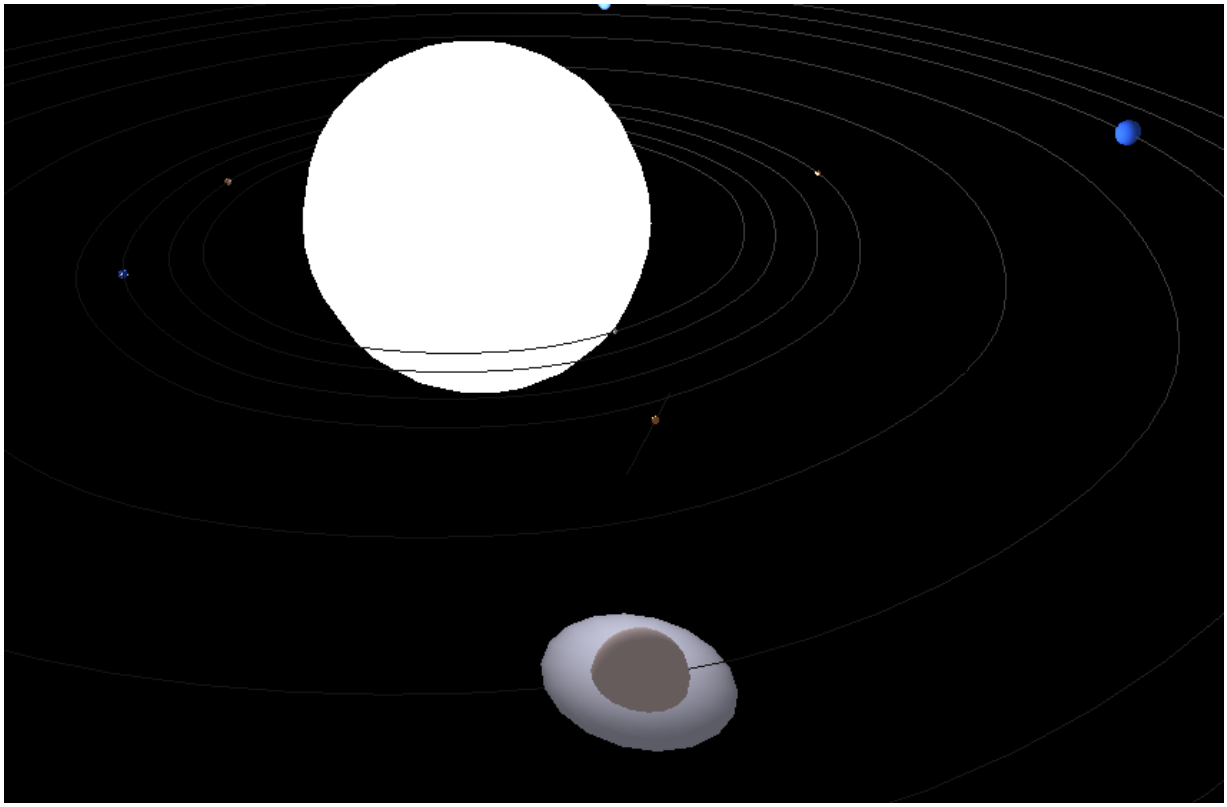

Figura 9 - Sistema Solar final

Apesar das tentativas, não conseguimos colocar texturas nos planetas. No entanto, vemos que os planetas movem-se à volta do Sol, e a luz é dinâmica, ou seja, consoante a posição dos planetas, a luz incidente também é alterado.

CONCLUSÃO E TRABALHO FUTURO

Chegando ao final deste documento, podemos afirmar que esta foi a única fase em que não ficamos satisfeitos com o resultado final, uma vez que não conseguimos implementar as texturas dos planetas, o que daria outro aspeto ao Sistema Solar.

Ao longo deste projeto, apreendemos bastantes conceitos e habilitações sobre OpenGL, ficando preparados para a resolução do teste final da cadeira.

Por fim, consideramos que, numa vista geral, o trabalho realizado foi satisfatório, uma vez que podemos, sem qualquer dificuldade, ver todas as componentes do Sistema Solar, com as suas rotações e luzes.

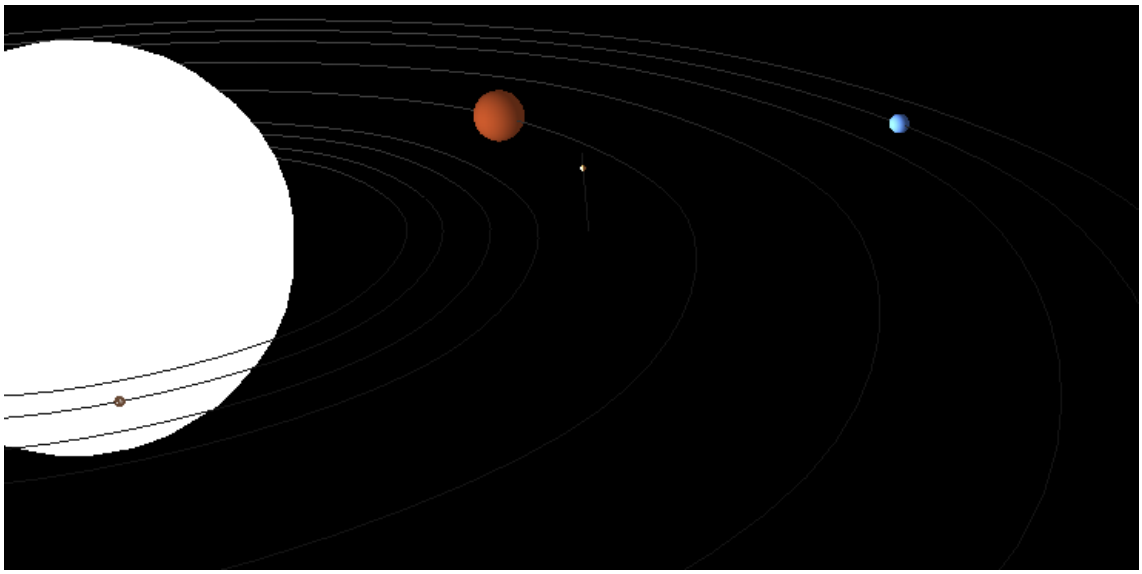

Figura 10 - Imagem do Sistema Solar final