



Universidade do Minho
Escola de Engenharia

COMPUTAÇÃO GRÁFICA

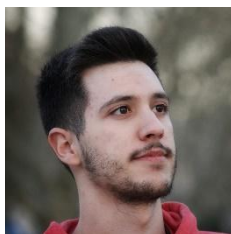
PHASE 4 – NORMALS AND TEXTURE
COORDINATES

GRUPO 30 - MAIO 2021



Luís Miguel Pinto A89506

Ana Luísa Carneiro A89533



Pedro Almeida Fernandes A89574

Ana Rita Peixoto A89612



ÍNDICE

INTRODUÇÃO	2
GENERATOR	3
ESFERA	3
CONE	4
CAIXA	5
PLANO	6
CILINDRO	6
TEAPOT/PATCHES DE BEZIER	7
ANEL PLANO	9
FICHEIRO 3D	10
ENGINE	11
ILUMINAÇÃO	11
TEXTURAS	11
FICHEIRO XML	12
FUNCIONALIDADE EXTRA FPS	12
PALETE DE TEXTURAS	13
DEMO SCENES	15
ESFERA	15
CONE	15
CAIXA	16
PLANO	17
CILINDRO	17
ANEL PLANO	18
TEAPOT/PATCHES DE BEZIER	18
SISTEMA SOLAR	19
CONCLUSÃO	21
ANEXOS	22
solarSystem.xml	22

INTRODUÇÃO

Esta quarta e última fase do trabalho prático teve como objetivo completar o trabalho realizado com a adição de texturas e iluminação, o que permitiu obter uma representação mais realista das cenas.

Assim, nesta fase os temas abordados foram principalmente os vetores normais e as coordenadas de textura em cada vértice, que se refletiram em alterações nos programas *Generator* e *Engine*.

No caso do *Generator*, foi necessário alterar cada primitiva de forma a que a escrita no ficheiro 3d albergue também as coordenadas de textura e as normais a cada vértice. Além disso, também foi alterada a função que anteriormente gerava o anel de saturno.

No programa *Engine* foram efetuadas alterações de modo a ser possível efetuar a leitura dos novos ficheiros 3d, e também para suportar a iluminação e aplicação de texturas às primitivas.

No presente relatório apresentamos explicações detalhadas de todas as etapas que sustentaram esta fase do projeto, acompanhadas por pseudocódigo e imagens de forma a ilustrar o raciocínio usado e manter uma documentação exemplificativa do projeto.

GENERATOR

De forma a possibilitar a aplicação de texturas e iluminação às primitivas foi necessário efetuar o cálculo das coordenadas de textura e das normais de cada vértice gerado para cada primitiva. Estas alterações foram efetuadas nas respetivas funções no programa *Generator*. O procedimento aplicado a cada primitiva será explicado em detalhe em cada uma das seguintes secções.

ESFERA

De forma a calcular as coordenadas de textura e normais da esfera foi necessário rever a função que cria uma esfera no *Generator*.

Para o cálculo das normais de cada vértice, o raciocínio passou por traçar o vetor que parte da origem do referencial até cada vértice. Deste modo, o vetor normal pode ser calculado através da diferença entre o vértice e a origem, tal como explicitado na seguinte equação:

$$\vec{N} = P - O, \text{ sendo que } P \text{ denota o vértice em questão e } O \text{ a origem do referencial.}$$

Na seguinte figura é possível observar uma esfera 3D que explicita o raciocínio adotado no cálculo das normais e apresenta uma visão representativa do vetor normal no ponto P:

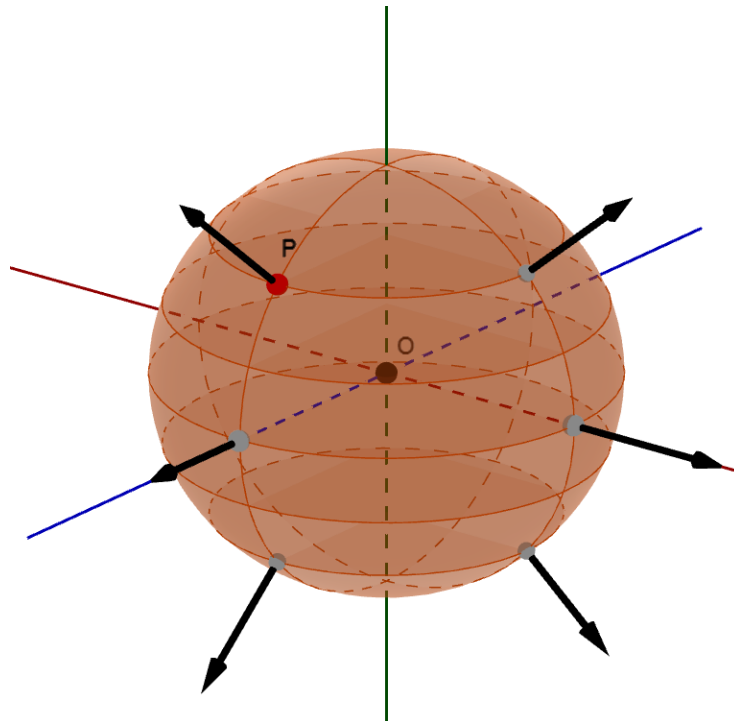


Figura 1: Esfera e normal no ponto P

Para as coordenadas de textura o procedimento passou por observar a esfera dividida nas suas *slices* e *stacks* e transportar essa representação para uma imagem de textura retangular, ou seja, as coordenadas calculadas correspondem a uma pequena porção da imagem que irá ocupar cada pedaço da esfera cortado pelas *slices* e *stacks*. O cálculo das coordenadas de textura teve em consideração o método de cálculo dos vértices da esfera. Por esta razão foram calculadas coordenadas de textura para a parte superior e inferior da esfera seguindo raciocínios análogos.

CONE

Para determinarmos as coordenadas de textura e as normais de cada coordenada do cone foi necessário acrescentar à função *creatCone* que determina as coordenadas dos vários pontos dos triângulos o cálculo das coordenadas de textura e normais de cada ponto.

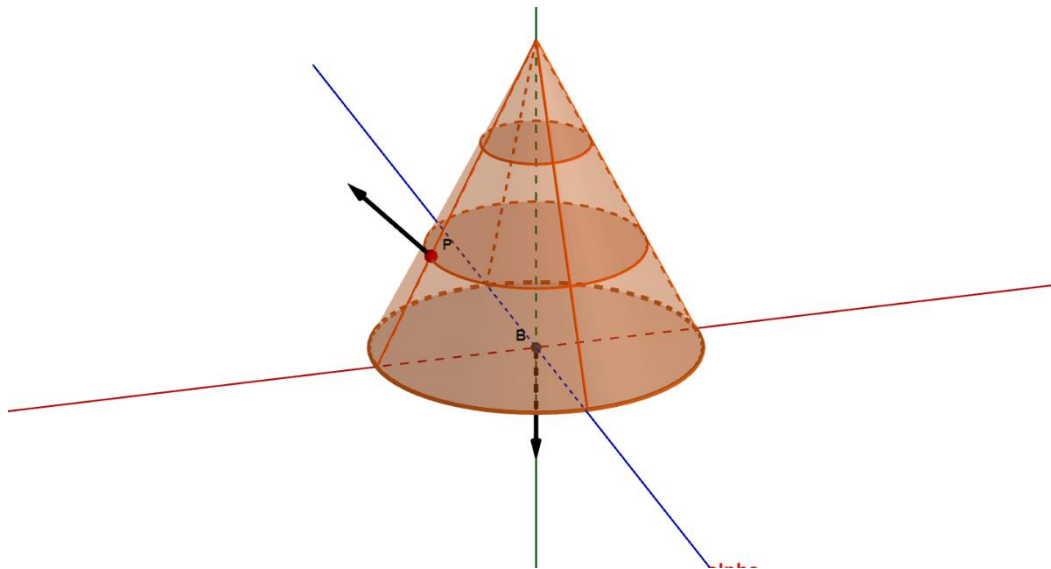


Figura 2: Normais para a primitiva Cone

As normais foram calculadas tal como está na imagem, assim para a base do cone é necessário que as normais apontem para baixo. Contudo, para os pontos da circunferência é necessário que estes tenham duas normais uma vez que estes tanto pertencem à base como à superfície curva. Na que toca às normais da superfície esféricas estas foram determinadas a partir da normalização das coordenadas desse ponto.

Para as coordenadas de textura, estas foram determinadas em função à textura que pretendíamos implementar no cone. Na figura seguinte está representada a textura que foi aplicada ao cone e as respetivas dimensões a serem aplicada a cada coordenada do cone. Desta forma a parte amarela da textura foi utilizada para a superfície do cone e a parte rosa á base do cone.

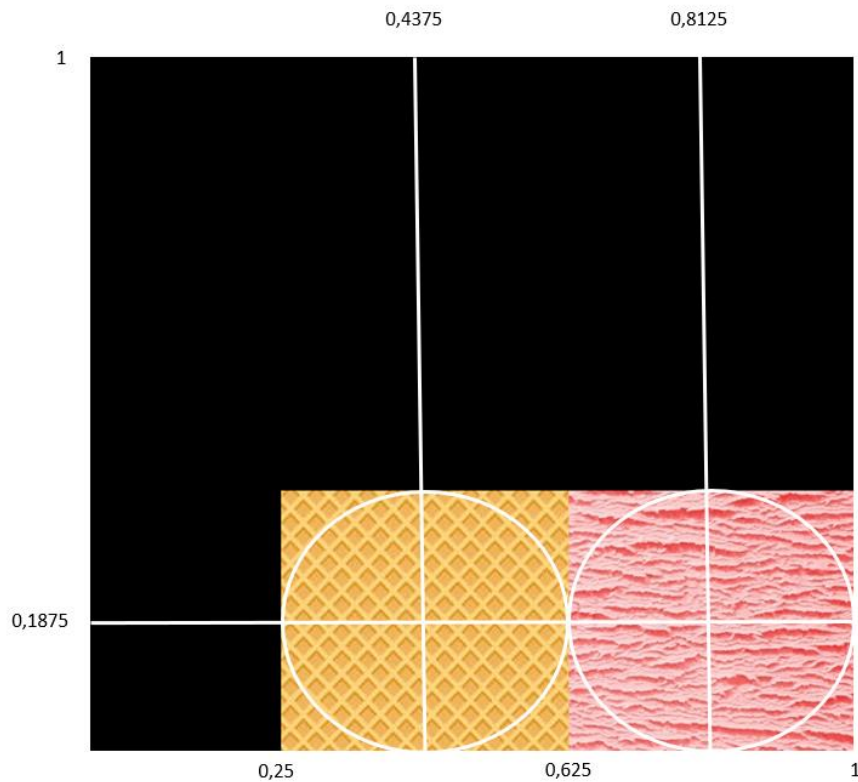


Figura 3: Textura que define as coordenadas de textura para o cone

CAIXA

Para determinarmos as coordenadas de textura e as normais de cada coordenada da caixa foi necessário acrescentar à função *buildPlanes* que determina as coordenadas dos vários pontos dos triângulos o cálculo das coordenadas de textura e normais de cada ponto.

Para determinar as normais decidiu-se determinar os vetores tal como estão na figura x. Assim para cada ponto do quadrado vai existir três normais, uma para cada face a que o ponto pertence.

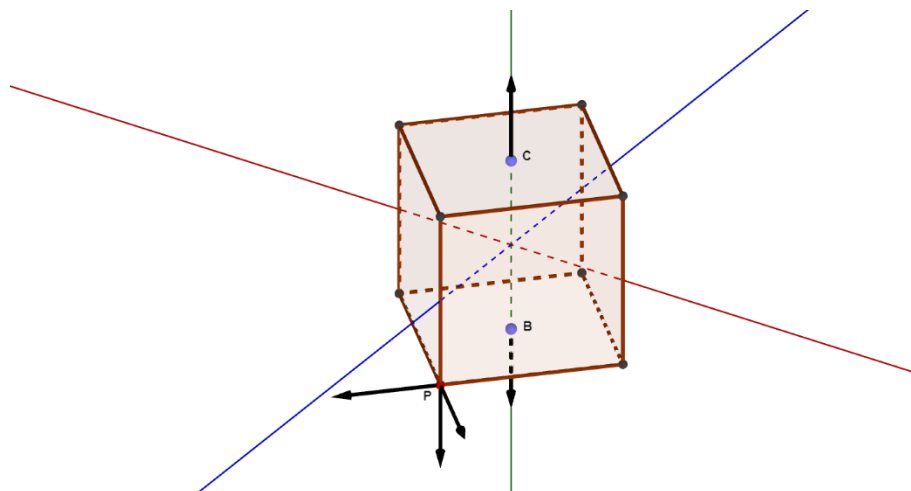


Figura 4: Normais da caixa

Desta forma o ponto P da imagem conterá as seguintes normais $(1,0,0)$, $(0,-1,0)$ e $(0,0,1)$. É necessário ter em conta que as faces opostas vão ter normais simétricas tal como esta representado para os pontos B e C.

Para as coordenadas de textura estas vão ser aplicadas a cada face do cubo por isso cada face vai ter as coordenadas de textura $(1,1)$, $(0,1)$, $(0,0)$, $(1,0)$. No que toca à caixa com edges, isto é, caixa com divisões as coordenadas de textura vão ser definidas para cada divisão assim a textura vai ser implementada em cada divisão em vez de ser na face toda.

PLANO

No caso do plano, foram geradas também coordenadas de textura e vetores normais para cada ponto. Para o cálculo das normais o procedimento é trivial, dado que o vetor normal em cada ponto trata-se do vetor vertical $(0,1,0)$. A título de exemplo, é possível observar na seguinte figura os vetores normais em alguns pontos, nomeadamente do ponto P. É de notar que este vetor possui direção vertical e sentido de baixo para cima.

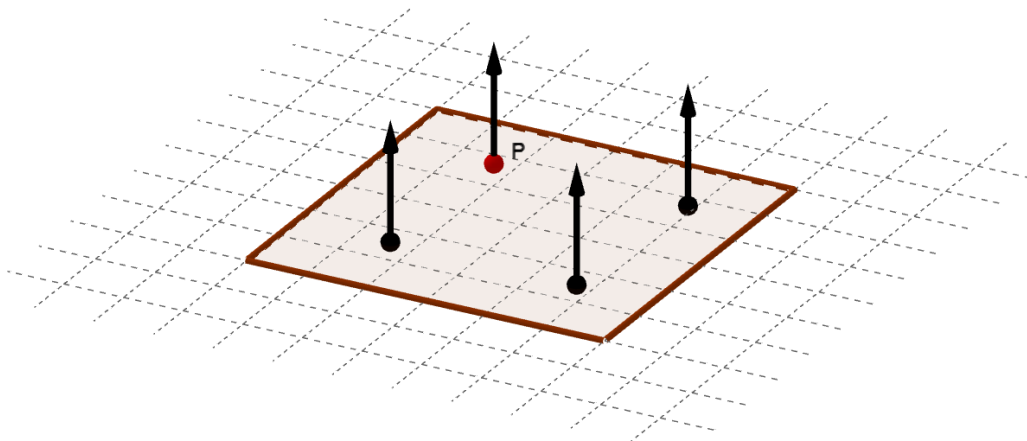


Figura 5: Vetores normais do plano

Para as coordenadas de textura, o cálculo foi simplificado na medida em que o plano já possui a forma da imagem da textura. Deste modo, as coordenadas de textura são simplesmente os pontos $(0,1)$, $(1,0)$, $(0,0)$ e $(1,1)$.

CILINDRO

Foi necessário atualizar a função *creatCylinder* no *Generator* de modo a ter em consideração as coordenadas de textura e vetores normais em cada vértice.

Assim, para o cálculo dos vetores normais foi necessário ter em consideração a localização dos triângulos gerados. Por exemplo, no caso dos triângulos no topo do cilindro, os vetores normais tem a direção vertical e sentido de baixo para cima, podendo assim ser representados pelo vetor $(0,1,0)$. Para os triângulos das laterais, o vetor a adotar para os vértices que os constituem será radial e horizontal. Para a base do cilindro o raciocínio é análogo ao topo e o vetor será $(0,-1,0)$. O raciocínio aplicado neste cálculo é visível através da seguinte representação:

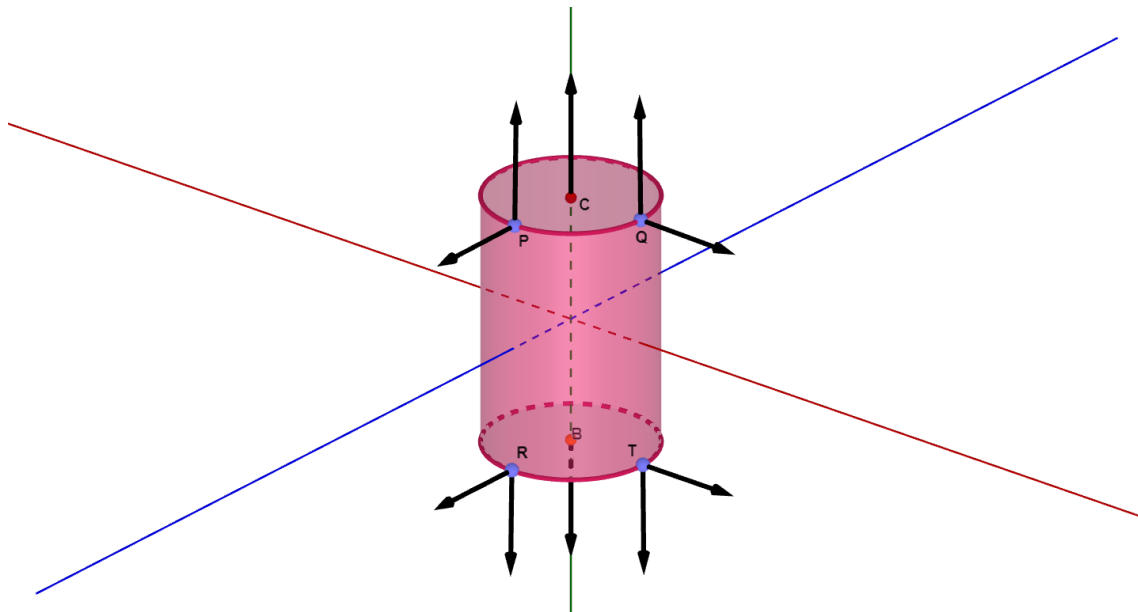


Figura 6: Normais do cilindro

Para o cálculo das coordenadas de textura foi necessário projetar o cilindro para uma representação plana e efetuar o cálculo numa escala de 0 a 1 de cada uma dessas coordenadas, conforme o local onde cada parte do cilindro (topo, base, laterais) se encontra localizada na imagem.

TEAPOT/PATCHES DE BEZIER

Para determinarmos as coordenadas de textura e as normais de cada coordenada do teapot foi necessário acrescentar à função *buildPointsComet* que determina as coordenadas dos vários pontos dos patches as coordenadas de textura e normais de cada um desses pontos.

Ao longo desta função além de serem calculados os pontos do teapot também se calcula juntamente a sua normal e a coordenada de textura. Para as coordenadas de textura derivou-se os vetores u e v , obtendo as seguintes expressões.

$$u' = (3 * u^2 \quad 2 * u \quad 1 \quad 0)$$

$$v' = (3 * v^2 \quad 2 * v \quad 1 \quad 0)$$

Para determinarmos a normal de um ponto no patch bastava fazer o produto vetorial entre os vetores calculados acima, normalizando o resultado final através das funções *cross* e *normalize*, respetivamente.

$$p_{normal}(u, v) = u' \cdot v'$$

$$p_{normal}(u, v) = \text{normalize}(p_n(u, v))$$

Para as coordenadas de textura de cada ponto assumiu-se que a textura iria ser aplicada a cada patch em vez que no teapot no todo. Com cada patch é representado da segundo a grelha da figura x então a textura a ser utilizada vai ser dividida segundo as mesmas divisões que a grelha.

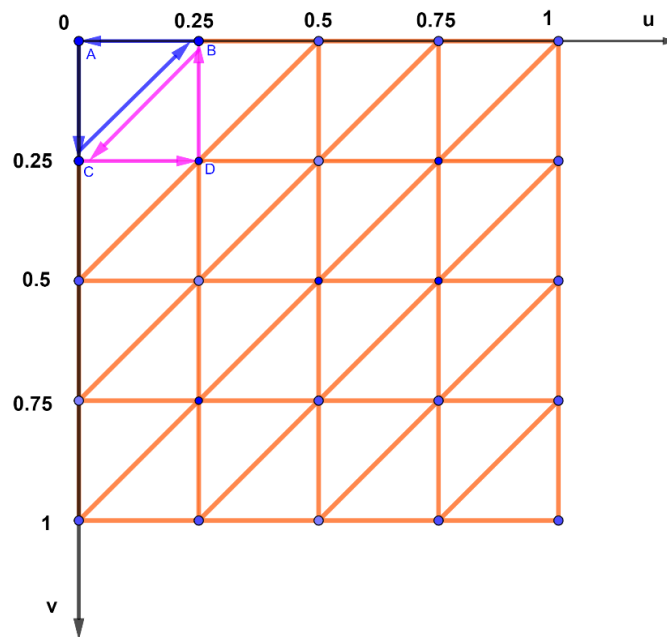


Figura 7: Grelha representativa de um patch

Desta forma cada coordenada de textura será obtida em função de cada patch segundo as seguintes coordenadas.

$$p_{texture} = (u, v)$$

Ambos os resultados da coordenada de textura como da normal serão armazenados numa matriz que e será posteriormente percorrida de forma a formar pontos de um triângulo.

ANEL PLANO

Nesta fase foi adicionada uma nova primitiva que denota um anel plano. Esta primitiva tem como objetivo representar o anel de saturno e vem substituir a representação adotada na fase anterior. Deste modo, tal como nas restantes primitivas foi necessário calcular os vetores normais a cada vértice e as coordenadas de textura.

O cálculo dos vetores normais foi trivial. Para a parte de cima do anel consideramos o vetor vertical $(0,1,0)$ em cada ponto, e para a parte inferior o vetor $(0,-1,0)$. Na seguinte imagem é possível observar o raciocínio para o cálculo das normais:

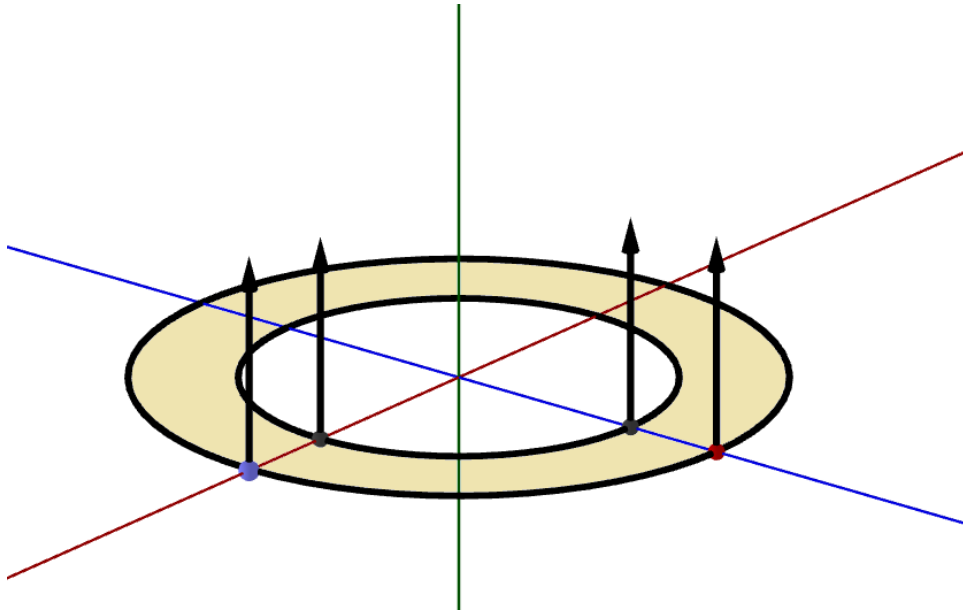


Figura 8: Normais do anel plano

Para o cálculo das coordenadas de textura efetuou-se uma projeção do anel numa imagem de textura retângular e através de cálculos matemáticos, considerando os raios interno e externo, calcularam-se as coordenadas de textura.

FICHEIRO 3D

As alterações efetuadas neste programa condicionaram a estrutura do ficheiro 3d, na medida em que cada linha agora denota um conjunto de 8 pontos, sendo que os 3 primeiros corresponde ao vértice em si, os 3 seguintes denotam as normais desse vértice e os 2 últimos as coordenadas de textura. Este ficheiro é gerado para cada primitiva aquando da criação da mesma. É de notar que a linha inicial deste ficheiro contém o número de vértices presentes no ficheiro para possibilitar a leitura correta no *Engine*. A seguinte figura denota a estrutura típica de um ficheiro 3d gerado pelo programa *Generator*:

```
9360
0.000000,1.000000,-0.000000,0.000000,1.000000,-0.000000,0.000000,-0.000000
0.312250,0.950000,-0.000000,0.312250,0.950000,-0.000000,0.000000,-0.025000
0.308406,0.950000,-0.048847,0.308406,0.950000,-0.048847,0.025000,-0.025000
0.312250,-0.950000,-0.000000,0.312250,-0.950000,-0.000000,0.000000,-0.975000
0.000000,-1.000000,-0.000000,0.000000,-1.000000,-0.000000,0.000000,-1.000000
0.308406,-0.950000,-0.048847,0.308406,-0.950000,-0.048847,0.025000,-0.975000
0.000000,1.000000,-0.000000,0.000000,1.000000,-0.000000,0.025000,-0.000000
0.308406,0.950000,-0.048847,0.308406,0.950000,-0.048847,0.025000,-0.025000
0.296967,0.950000,-0.096491,0.296967,0.950000,-0.096491,0.050000,-0.025000
0.308406,-0.950000,-0.048847,0.308406,-0.950000,-0.048847,0.025000,-0.975000
0.000000,-1.000000,-0.000000,0.000000,-1.000000,-0.000000,0.025000,-1.000000
0.296967,-0.950000,-0.096491,0.296967,-0.950000,-0.096491,0.050000,-0.975000
0.000000,1.000000,-0.000000,0.000000,1.000000,-0.000000,0.050000,-0.000000
```

Figura 9: Estrutura típica de um ficheiro 3d

ENGINE

De modo a suportar os requisitos propostos para a 4ª fase do trabalho prático foi necessário efetuar algumas alterações no programa *Engine*. Assim, este programa agora suporta a leitura de ficheiros XML e 3d mais complexos, isto é, de modo a conseguir albergar as diferentes texturas e iluminação de cada primitiva.

ILUMINAÇÃO

Com o objetivo de implementar a iluminação no projeto foi necessário efetuar algumas alterações no *Engine*. Procedeu-se à ligação das luzes com recurso às funções *glEnable(GL_LIGHTING)* e *glEnable(GL_LIGHT0)* do *OpenGL*, na função *initGL*. De seguida, durante a leitura do ficheiro XML, são armazenados os dados relativos a iluminação, isto é, quer as componentes de cor dos modelos (difusa, especular, emissiva e ambiente) quer as fontes de luz (POINT, DIRECTIONAL ou SPOT). Deste modo, cada primitiva irá possuir um conjunto de variáveis capazes de armazenar os dados contidos no ficheiro XML e, analogamente para as fontes de luz, foi criada uma variável global na classe *main* (*vector<Light> lights*) com o propósito de registar as fontes de luz do XML. A classe denominada por *Light* serve para acomodar os valores provenientes do XML como o tipo de luz e as coordenadas para a luz. Posteriormente, na função *renderScene* acontece a renderização da cena, lá ocorrerá a aplicação das fontes de luz e a invocação da função *drawPrimitives*, responsável por percorrer todas as estruturas de dados e para cada qual aplicar as transformações necessárias, agora também responsável por aplicar as componentes de cor aos modelos.

Adicionalmente, foi considerado um novo VBO para as coordenadas normais, de modo a ser possível aplicar a iluminação. No momento de desenho das primitivas são invocadas as funções *glBindBuffer* e *glNormalPointer* devido à implementação com VBOs.

TEXTURAS

O processo de implementação das texturas ocorre ao mesmo passo que do processo de implementação das iluminações anteriormente abordado, na medida em que são lidas do ficheiro XML as informações, neste caso informações sobre as texturas a aplicar aos modelos. Estas informações são numa etapa seguinte guardadas em estruturas de dados na classe *Primitive* que armazena o caminho para a imagem de textura pretendida.

De forma a aplicar as texturas corretamente a todos os modelos, foi necessário implementar uma estrutura *map<string, GLuint>* texturas em que o seu conteúdo são pares de (Nome textura, ID Textura). Assim, em cada iteração do ciclo *glutMainLoop()* é efetuado o *load* da textura e atualizado o mapa de texturas. No momento de renderização da cena, em concreto quando a função *drawPrimitives* é invocada e são desenhadas as primitivas, verifica-se qual a textura correspondente à primitiva em questão e visita-se o *map* texturas para obter o ID correto.

É de notar que as coordenadas de textura foram armazenadas num VBO específico. No momento de desenho das primitivas é invocada a função *glBindBuffer* que utiliza o ID da textura correspondente à primitiva e as funções *glBindBuffer* e *glTexCoordPointer* para permitir a aplicação de texturas às primitivas.

FICHEIRO XML

O ficheiro XML suportado pelo Engine contém novos identificadores que remetem à iluminação e às texturas. No caso das luzes, é possível definir fontes de luz através da definição do tipo de luz como “POINT”, “DIRECTIONAL” ou “SPOT”. Além disso, também é possível atribuir componentes de cor difusa, especular, emissiva ou ambiente a cada primitiva. Para isso, foi necessário acrescentar uma nova função *parseLights* que é invocada na função de leitura do XML (*parseDocument*) e tem como objetivo ler e armazenar num *vector* os pontos de luz da cena. Em adição, também foi necessário implementar uma nova função *readConfigModel* com o propósito de analisar de uma primitiva possui textura ou componentes de cor.

Foi realizada uma alteração em relação à localização do ficheiro xml lido. Anteriormente era necessário colocar o ficheiro pretendido numa pasta específica, no entanto agora é possível coloca-lo onde se quiser visto que o *engine* passou a receber o caminho para o mesmo ficheiro como parâmetro.

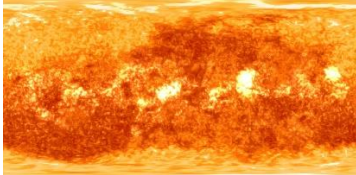



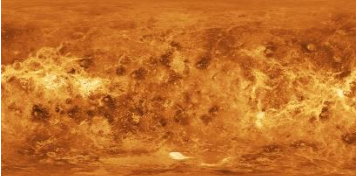







FUNCIONALIDADE EXTRA FPS

Os *fps* mais concretamente *frames per second*, representam a quantidade de *frames*, ou seja imagens, são apresentadas no ecrã por cada segundo. Visto que a maior parte dos monitores do mercado têm 60 Hz de refrescamento do ecrã, para obter algo que corra de forma bastante são necessários de 60 *fps*.

De modo a testar-mos a *performance* do nosso programa decidimos implementar esta funcionalidade que nos demonstra a suavidade do nosso programa. Visto que em médio com o sistema solar obtivemos cerca de 450 *fps*, sendo que as nossas placas gráficas são de gama média-baixo, consideramos que conseguimos obter um bom resultado.

PALETE DE TEXTURAS

Na seguinte tabela é possível observar as diferentes texturas utilizadas para cada primitiva/objeto:

Objeto	Textura	Objeto	Textura
Sol		Júpiter	
Mercúrio		Saturno	
Vénus		Úrano	
Terra		Neptuno	
Marte		Anel de Saturno	
Caixa		Cilindro	



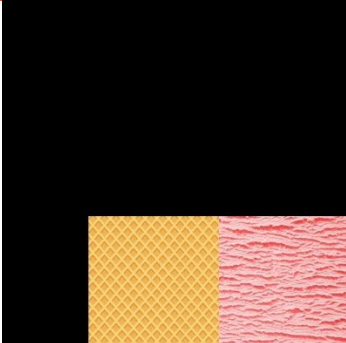

Plano		Teapot	
Cone		Teapot (cometa)	

Tabela 1: Paleta de texturas

DEMO SCENES

De modo a demonstrar de forma visual o trabalho realizado, nesta secção iremos dar exemplos de objetos com luzes e texturas. Serão apresentados pelo menos uma representação de cada primitiva implementada.

ESFERA

De seguida encontra-se uma demonstração de uma esfera. Foi escolhida para apresentação a textura do planeta Terra como podemos observar na imagem seguinte:



Figura 10: Demonstração esfera com textura e iluminação

CONE

Para a demonstração do cone usamos uma textura de cone de gelado para as bordas e gelado de morango na base. Em seguida apresenta-se a representação do mesmo:

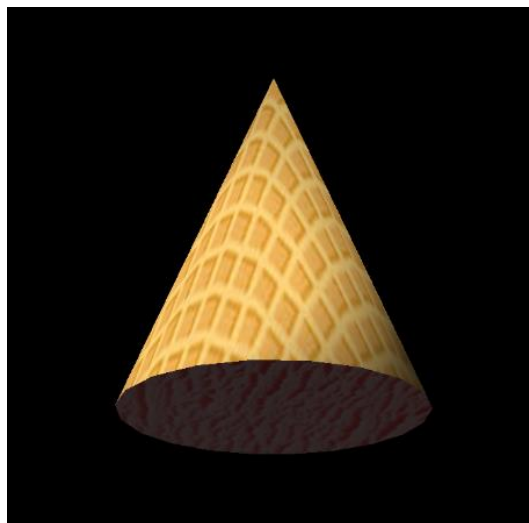


Figura 11: Demonstração Cone

CAIXA

No caso da caixa usamos uma textura de caixa de madeira. Em seguida estão apresentadas duas cenas: uma caixa sem *edges* que se materializa na representação de apenas uma caixa, e de uma caixa com 4 *edges* que dá a aparência de uma série de caixas empilhadas.



Figura 12: Demonstração Caixa com 1 divisão

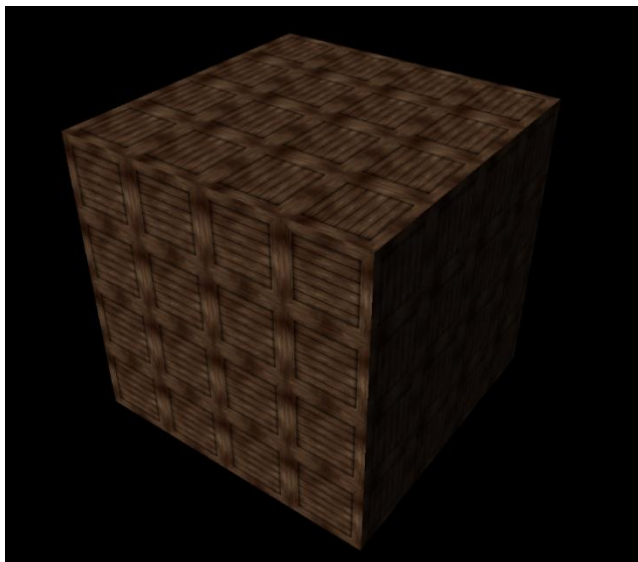


Figura 13: Demonstração Caixa com 4 *edges*

PLANO

Para representar um plano decidimos utilizar uma textura de relva como se encontra na seguinte imagem:

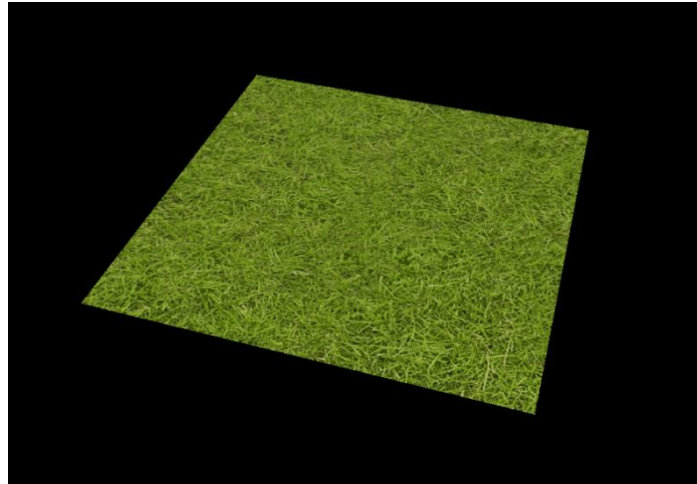


Figura 14: Demonstração Plano

CILINDRO

No seguinte imagem encontra-se a demonstração de um cilindro. Para este caso decididos utilizar a textura de uma lata de coca-cola.



Figura 15: Demonstração Cilindro

ANEL PLANO

De modo a ser possível realizar o anel de saturno foi necessário criar esta primitiva. Em seguida encontra-se uma demonstração do mesmo.

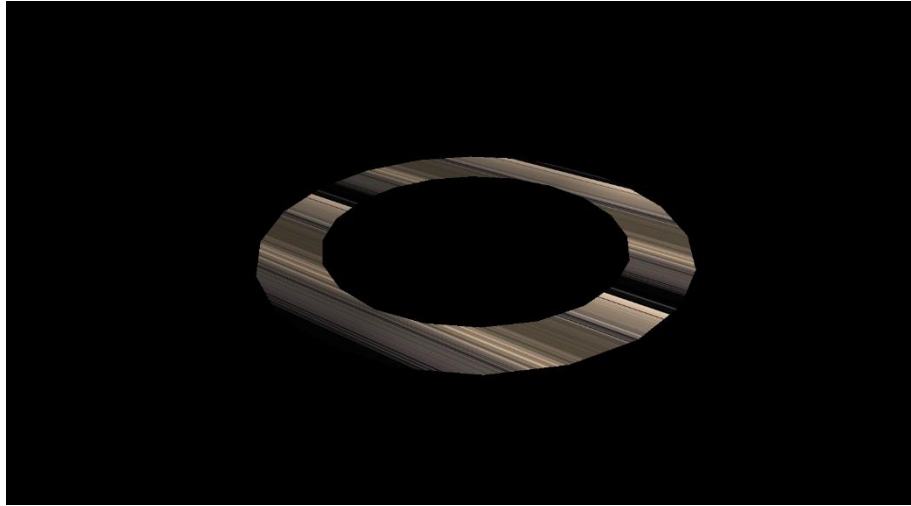


Figura 16: Demonstração Anel

TEAPOT/PATCHES DE BEZIER

A nossa última primitiva desenvolvida foi o famoso *teapot*. Para tal usamos uma textura de porcelana obtendo o seguinte resultado:



Figura 17: Demonstração Teapot

SISTEMA SOLAR

No sistema solar foi aplicado um pouco de tudo o que foi realizado durante as diversas fases do trabalho. Apresenta-se em seguida algumas imagens do mesmo, sendo que na segunda podemos também observar o cometa (*teapot*) a percorrer a sua rota. De modo a ser mais fácil de visualizar as texturas dos planetas na terceira imagem foi usada uma *spotlight* enquanto que nas outras duas apenas existe a luz emissiva do sol.



Figura 18: Demonstração Sistema Solar 1

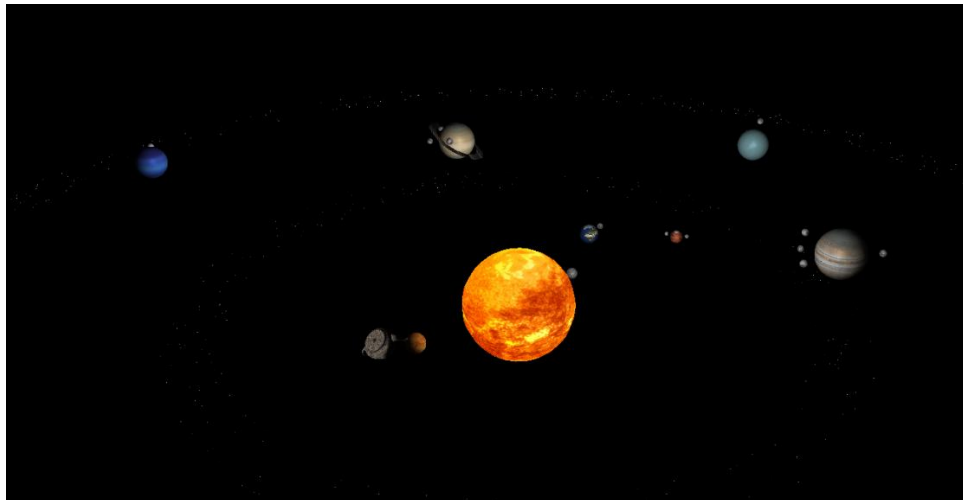


Figura 19: Demonstração Sistema Solar 2

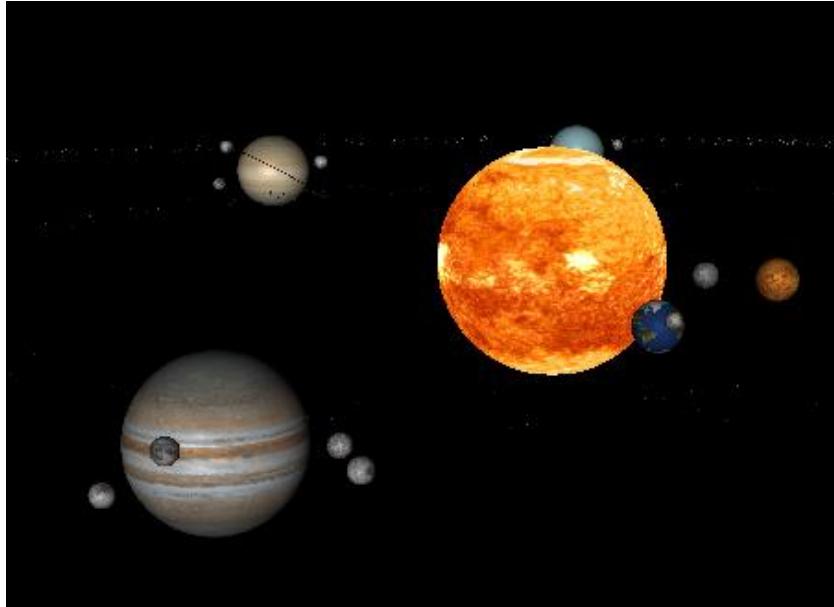


Figura 20: Demonstração Sistema Solar 3

CONCLUSÃO

Dada por concluída a fase 3 do projeto, consideramos relevante efetuar uma análise crítica do trabalho realizado.

No espectro positivo, consideramos relevante destacar o correto funcionamento do programa e o cumprimento de todos os requisitos expectáveis para esta fase, tais como a definição de coordenadas de textura e vetores normais e a aplicação de iluminação e texturas às primitivas.

Por outro lado, também existiram algumas dificuldades, das quais destacamos a gestão da memória utilizada na renderização de uma cena. Apesar disso, conseguimos contornar esse obstáculo.

Para concluir, consideramos que houve um balanço positivo do trabalho realizado dado que as dificuldades sentidas foram superadas e foram cumpridos todos os requisitos.

ANEXOS

solarSystem.xml

```

<scene>

    <lights>

        <light type="DIRECTIONAL" posX="2" posY="2" posZ="2"/>

    </lights>

    <group>

        <rotate time="15" x="0" y="1" z="0" />

        <models>

            <model file="sphere.3d" texture="sol.jpg" emissR="1" emissG="1" emissB="1"/>

        </models>

    </group>

    <group>

        <rotate time="2" x="0" y="1" z="0" />

        <translate x="1.5" y="0" z="0"/>

        <group>

            <rotate time="30" x="0" y="1" z="0" />

            <scale x="0.11" y="0.11" z="0.11"/>

            <models>

                <model file="sphere.3d" texture="mercurio.jpg" specR="1" specG="1"
specB="1"/>

            </models>

        </group>

    </group>

    <group>

        <rotate time="6" x="0" y="-1" z="0" />

        <translate x="2.1" y="0" z="0"/>

        <group>

            <rotate time="80" x="0" y="-0.9" z="0" />

            <scale x="0.18" y="0.18" z="0.18"/>

            <models>

                <model file="sphere.3d" texture="venus.jpg" specR="1" specG="1"
specB="1"/>

            </models>

        </group>

    </group>

    <group>

        <rotate time="12" x="0" y="1" z="0" />

        <translate x="3.1" y="0" z="0"/>

        <group>

            <rotate time="10" x="0" y="0.7" z="0" />

            <scale x="0.19" y="0.19" z="0.19"/>

```

```

        <models>
            <model file="sphere.3d" texture="terra.jpg" specR="1" specG="1"
specB="1"/>
        </models>
        <group>
            <rotate time="16" x="0" y="1" z="0" />
            <translate x="1.5" y="0.5" z="0"/>
            <group>
                <rotate time="9" x="0" y="1" z="0" />
                <scale x="0.33" y="0.33" z="0.33"/>
                <models>
                    <model file="sphere.3d" texture="lua.jpg"/>
                </models>
            </group>
        </group>
    </group>
    <group>
        <rotate time="18" x="0" y="1" z="0" />
        <translate x="4.1" y="0" z="0"/>
        <group>
            <rotate time="10" x="0" y="0.7" z="0" />
            <scale x="0.13" y="0.13" z="0.13"/>
            <models>
                <model file="sphere.3d" texture="marte.jpg" specR="1" specG="1"
specB="1"/>
            </models>
            <group>
                <rotate time="15" x="0" y="1" z="0" />
                <translate x="1.5" y="0.5" z="0"/>
                <group>
                    <rotate time="11" x="0" y="1" z="0" />
                    <scale x="0.33" y="0.33" z="0.33"/>
                    <models>
                        <model file="sphere.3d" texture="lua.jpg"/>
                    </models>
                </group>
            </group>
        <group>
            <rotate time="19" x="1" y="1" z="0" />
            <translate x="1.5" y="-0.5" z="0"/>
            <group>
                <rotate time="11" x="0" y="1" z="0" />
                <scale x="0.33" y="0.33" z="0.33"/>
                <models>

```



```

        <model file="sphere.3d" texture="lua.jpg"/>
    </models>
</group>
</group>
</group>
</group>
<group>
    <rotate time="60" x="0" y="1" z="0" />
    <models>
        <model file="points.3d" />
    </models>
</group>
<group>
    <rotate time="36" x="0" y="1" z="0" />
    <translate x="6.5" y="0" z="0"/>
    <group>
        <rotate time="6" x="0" y="1" z="0" />
        <scale x="0.5" y="0.5" z="0.5"/>
        <models>
            <model file="sphere.3d" texture="jupiter.jpg" specR="1" specG="1"
specB="1"/>
        </models>
        <group>
            <rotate time="22" x="0" y="0" z="1" />
            <translate x="1.5" y="0.5" z="0"/>
            <group>
                <rotate time="10" x="0" y="1" z="0" />
                <scale x="0.15" y="0.15" z="0.15"/>
                <models>
                    <model file="sphere.3d" texture="lua.jpg"/>
                </models>
            </group>
        </group>
        <group>
            <rotate time="21" x="1" y="0" z="0" />
            <translate x="1.5" y="-0.2" z="0"/>
            <group>
                <rotate time="10" x="0" y="1" z="0" />
                <scale x="0.14" y="0.14" z="0.14"/>
                <models>
                    <model file="sphere.3d" texture="lua.jpg"/>
                </models>
            </group>
        </group>
    </group>

```

```

        <group>
        <rotate time="18" x="1" y="1" z="0" />
        <translate x="1.5" y="-0.5" z="0"/>
        <group>
            <rotate time="10" x="0" y="1" z="0" />
            <scale x="0.16" y="0.16" z="0.16"/>
            <models>
                <model file="sphere.3d" texture="lua.jpg"/>
            </models>
        </group>
    </group>

    <group>
        <rotate time="20" x="0" y="1" z="0" />
        <translate x="1.5" y="0.3" z="0"/>
        <group>
            <rotate time="10" x="0" y="1" z="0" />
            <scale x="0.15" y="0.15" z="0.15"/>
            <models>
                <model file="sphere.3d" texture="lua.jpg"/>
            </models>
        </group>
    </group>

</group>

<group>
    <rotate time="48" x="0" y="1" z="0" />
    <translate x="8" y="0" z="0"/>
    <group>
        <rotate time="5" x="0" y="1" z="0" />
        <scale x="0.45" y="0.45" z="0.45"/>
        <models>
            <model file="sphere.3d" texture="saturno.jpg" specR="1" specG="1"
specB="1"/>
        </models>
        <group>
            <scale x="0.48" y="0.48" z="0.48"/>
            <rotate z="0.7" y="0" x="1" angle="30"/>
            <models>
                <model file="anelSaturno.3d" texture="anel.png"/>
            </models>
        </group>
        <group>
            <rotate time="25" x="0" y="1" z="1" />
            <translate x="1.5" y="-0.5" z="0"/>

```

```

    <group>
      <rotate time="7" x="0" y="1" z="0" />
      <scale x="0.18" y="0.18" z="0.18"/>
      <models>
        <model file="sphere.3d" texture="lua.jpg"/>
      </models>
    </group>
  </group>

  <group>
    <rotate time="22" x="0" y="1" z="0" />
    <translate x="1.5" y="0.3" z="0"/>
    <group>
      <rotate time="7" x="0" y="1" z="0" />
      <scale x="0.19" y="0.19" z="0.19"/>
      <models>
        <model file="sphere.3d" texture="lua.jpg"/>
      </models>
    </group>
  </group>

  <group>
    <rotate time="18" x="1" y="1" z="0" />
    <translate x="1.5" y="-0.5" z="0"/>
    <group>
      <rotate time="7" x="0" y="1" z="0" />
      <scale x="0.17" y="0.17" z="0.17"/>
      <models>
        <model file="sphere.3d" texture="lua.jpg"/>
      </models>
    </group>
  </group>
</group>

<group>
  <rotate time="60" x="0" y="-1" z="0" />
  <translate x="9.5" y="0" z="0"/>
  <group>
    <rotate time="7" x="0" y="0" z="1" />
    <scale x="0.38" y="0.38" z="0.38"/>
    <models>
      <model file="sphere.3d" texture="urano.jpg" specR="1" specG="1"
specB="1"/>
    </models>
  </group>
  <rotate time="21" x="0" y="1" z="1" />

```

```

    <translate x="1.5" y="0.2" z="0"/>
    <group>
        <rotate time="10" x="0" y="1" z="0" />
        <scale x="0.19" y="0.19" z="0.19"/>
        <models>
            <model file="sphere.3d" texture="lua.jpg"/>
        </models>
    </group>
</group>

    <group>
        <rotate time="20" x="0" y="1" z="0" />
        <translate x="1.5" y="0.4" z="0"/>
        <group>
            <rotate time="10" x="0" y="1" z="0" />
            <scale x="0.20" y="0.20" z="0.20"/>
            <models>
                <model file="sphere.3d" texture="lua.jpg"/>
            </models>
        </group>
    </group>
</group>

    <group>
        <rotate time="78" x="0" y="1" z="0" />
        <translate x="10.75" y="0" z="0"/>
        <group>
            <rotate time="6" x="0" y="1" z="0" />
            <scale x="0.37" y="0.37" z="0.37"/>
            <models>
                <model file="sphere.3d" texture="neptuno.jpg" specR="1" specG="1"
specB="1"/>
            </models>
            <group>
                <rotate time="20" x="0" y="1" z="0" />
                <translate x="1.5" y="0.5" z="0"/>
                <group>
                    <rotate time="4" x="0" y="1" z="0" />
                    <scale x="0.21" y="0.21" z="0.21"/>
                    <models>
                        <model file="sphere.3d" texture="lua.jpg"/>
                    </models>
                </group>
            </group>
        </group>
    </group>

```

```
</group>
<group>
  <rotate time="60" x="0" y="1" z="0" />
  <models>
    <model file="pointsKuiper.3d" />
  </models>
</group>
<group>
  <translate time="15">
    <point x="-1" y="0" z="-4"/>
    <point x="-5" y="0" z="4"/>
    <point x="5" y="5" z="4"/>
    <point x="5" y="0" z="0"/>
  </translate>
  <scale x="0.1" y="0.1" z="0.1"/>
  <models>
    <model file="comet.3d" texture="lua.jpg"/>
  </models>
</group>
</scene>
```