Departamento de Informática Mestrado Integrado em Engenharia Informática

Índice

[Introdução 3](#_Toc449909629)

[VBOs 3](#_Toc449909630)

[Superfícies de Bézier 4](#_Toc449909631)

[Superfícies de Catmull-Rom 4](#_Toc449909632)

## Introdução

Nesta terceira fase do projeto proposto no âmbito da unidade curricular de Computação Gráfica damos seguimento ao trabalho desenvolvido na primeira e segunda fase. Nesta fase pretende-se estender a aplicação geradora de primitivas para que seja possível criar listas de pontos correspondentes a superfícies de Bézier. A aplicação receberá como parâmetros um ficheiro (*patch*) onde se encontram definidos os vértices relativos aos pontos de controlo e o grau de tesselação pretendido. Tal como para as outras primitivas, o resultado será um ficheiro com uma lista de pontos.

No que diz respeito ao motor 3D, as translações passam a ser definidas através da inclusão das curvas e de uma noção de tempo. Mais especificamente, uma translação é definida como um conjunto de pontos de controlo de uma curva e um tempo (em segundos, por exemplo) necessário para percorrer toda essa curva. Por isso, a animação passa a ser cíclica, isto é, quando a curva terminar regressa-se ao início da mesma, e com isto, o motor 3D passa a dispor de um relógio, ou contador de tempo (por exemplo: glutGet(GLUT\_ELAPSED\_TIME)).

Ainda sobre o motor 3D, as rotações também têm agora associado um tempo que surge no mesmo contexto que o ângulo na fase anterior.

Para além destas alterações, em vez do modo imediato, o motor 3D desenha as primitivas a partir de VBOs (*Vertex Buffer Object*).

Engine

A nível de alterações, o Engine foi mais modificado as novas funcionalidades do que o gerador. A começar pelo *parsing* que engloba mais dois termos, pois é necessário ler as normas e texturadas do .3d.

Ficheiro XML de input

De seguida, expomos um breve excerto de um ficheiro XML responsável pela estruturação da cena de desenho. Trata-se do já conhecido Sistema solar dinâmico, agora com luzes, sombras e texturas.

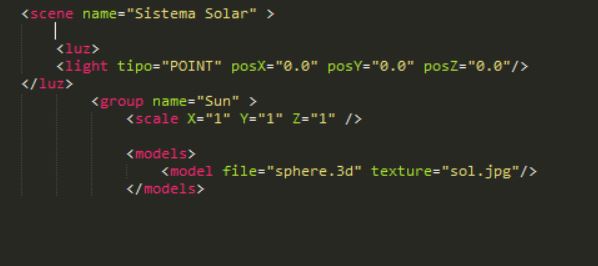


Figura Excerto do xml

O ficheiro XML evoluiu bastante ao longo das fases, passando de um simples ficheiro com primitivas geométricas, a um ficheiro complexo onde existem transformações geométricas, curvas, texturas e luzes.

Classe Luz

Com a introdução da luz foi necessário criar uma nova classe para lidar com a mesma .

A classe luz tem como objetivo guardar, neste caso, o único ponto de luz existente: o sol.

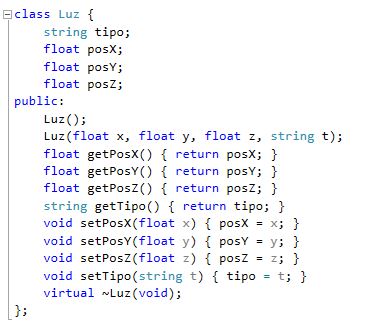


Figura Classe luz .h

Note que esta classe também admite os outros tipos de luzes dependendo do valor do campo “tipo”. As coordenadas 𝑝𝑜𝑠𝑋, 𝑝𝑜𝑠𝑌 𝑒 𝑝𝑜𝑠𝑍 correspondem à localização da fonte de luz na

cena.

No nosso caso especifico será o ponto (0,0,0). Acontece que como o sol emite luz em todas as direções, o seu tipo será de “ponto de luz”.

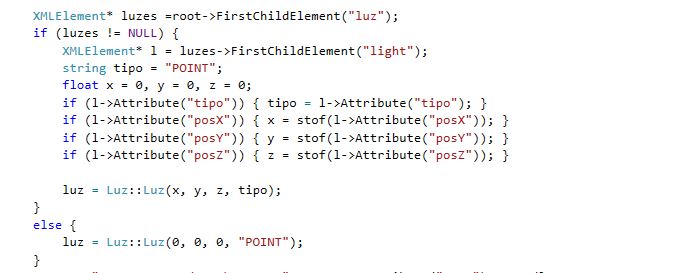


Figura parsing da Luz

Classe Model

A classe que mais evolui foi a Model, de maneira a suportar todos os atributos que lhe podem ser atribuídos .



Figura excerto do Model.h

A classe modelo passa agora a contemplar:

1. Três *buffers* para as VBOs (triângulos, normais e texturas).

2. O número de vértices. Na leitura das VBOs é fundamental saber o número de vértices de cada modelo. Ora, como é gerada normal para cada vértice, a leitura das VBOs de vértices e das normais é feita de três em três coordenadas. Contudo as texturas são lidas de dois em dois, devido às coordenadas cartesianas a duas dimensões.

3. O *nome* da imagem de textura. 4.

4. O código da difusão de cor.

5. variáveis da textura e da própria textura já carregada

6. Os vetores que contêm os pontos dos vértices, normais e texturas.



Figura novas estruturas

Resultados obtidos

Nesta secção exibimos os resultados obtidos na conclusão da última fase deste projeto.

## 

Figura Sistema solar 1

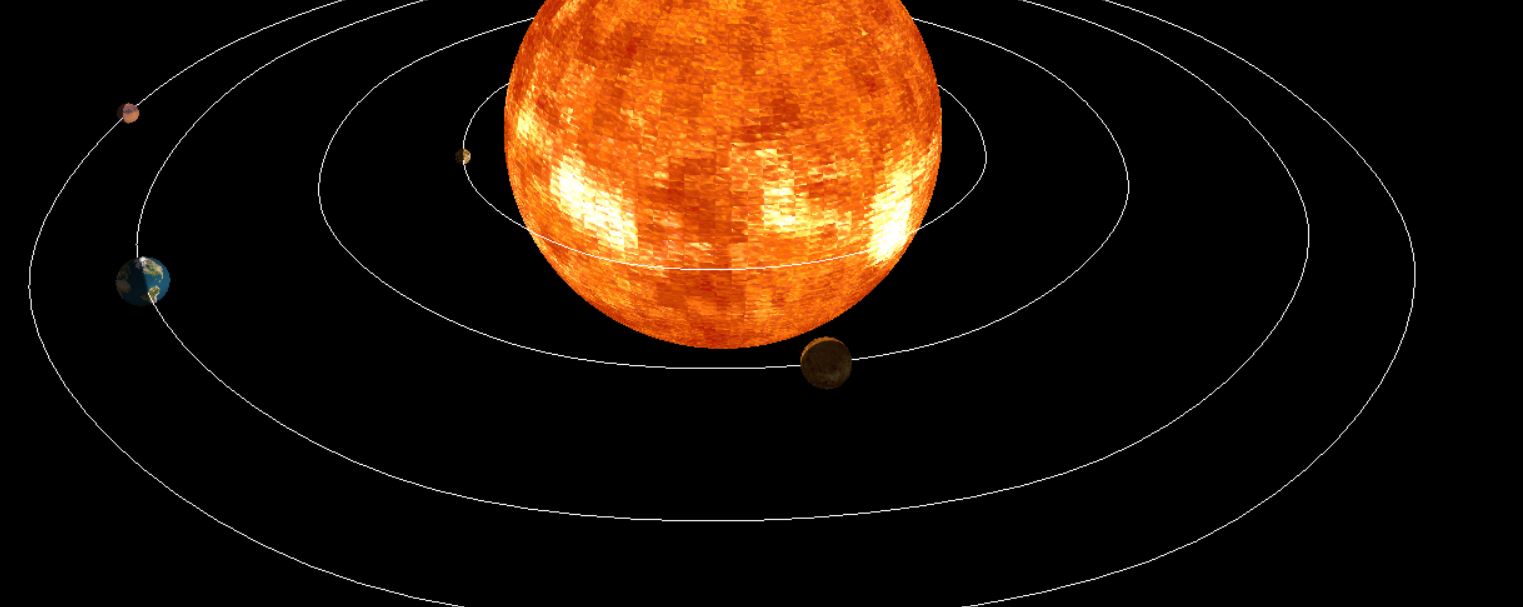


Figura Sistema Solar 2

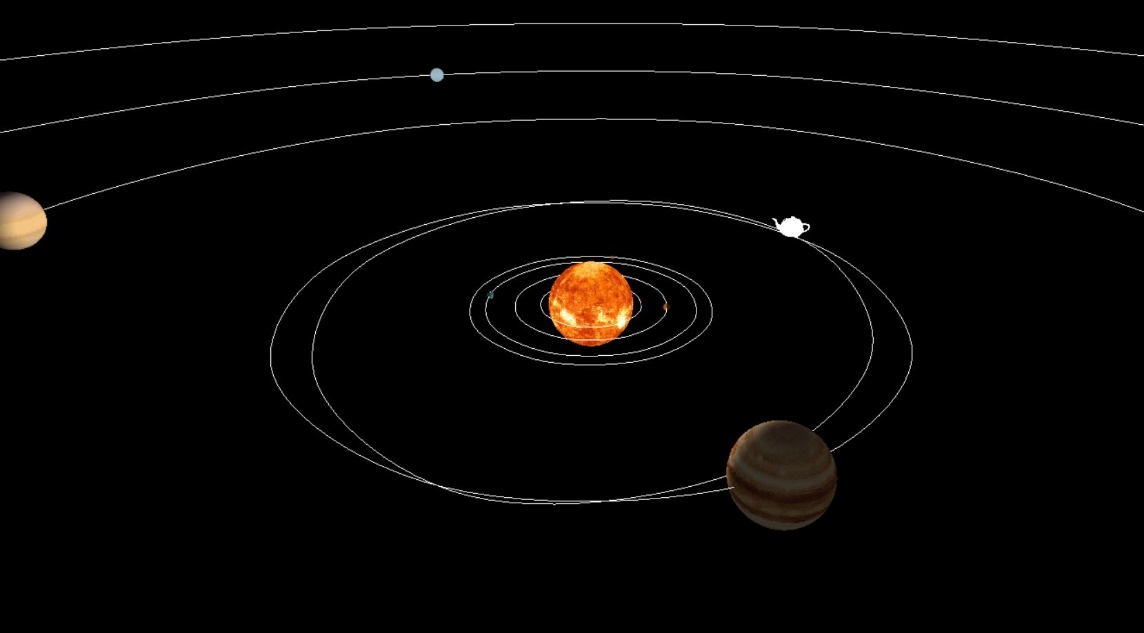


Figura Sistema Solar 3

No entanto ocorreram erros inesperados de ultima hora, o que causou algumas falhas nomeadamente nas luas.