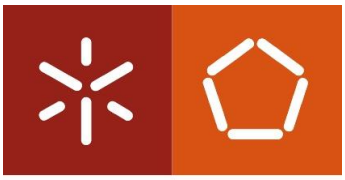

Universidade do Minho

Departamento de Informática

Computação Gráfica
Animações

Bárbara Cardoso (a80453) Márcio Sousa (a82400)
Pedro Mendes (a79003)

12 de março de 2022

Conteúdo

1 Introdução

Este trabalho foi proposto no âmbito da unidade curricular de Computação Gráfica, e tem como objetivo desenvolver um motor gráfico genérico para representar objetos a 3 dimensões. Este relatório tem como contexto ambos os relatórios anteriores.

Nesta terceira fase do projeto estendemos as capacidades do motor gráfico anteriormente definido com a adição de animações e de mais opções de camera.

O gerador de modelos foi também estendido para processar *Bezier Patches* de forma a ser mais fácil adicionar modelos complexos sem ser necessário o uso das primitivas.

Por fim, o modelo de teste utilizado (O Sistema Solar) foi alterado para demonstrar estas capacidades, fazendo com que os planetas orbitassem o Sol e com que as luas orbitassem os planetas, contando ainda com a adição de um cometa e de uma cintura de asteroides.

2 Arquitetura do Projecto

2.1 Model

Os modelos passaram a ser desenhados com *Vertex Buffer Objects* (VBOs). Estes permitem reduzir o número de pedidos feitos à placa gráfica, aumentando assim, significativamente, a performance do programa.

Por outro lado, perderam a possibilidade de ser desenhados com todos os seus triângulos pintados de cores aleatórias, visto que não é possível especificar a cor de cada triângulo quando estes são todos enviados para o GPU simultaneamente.

Esta desvantagem irá, no entanto, ser mitigada na próxima fase com a adição de luz e texturas.

2.2 Transformations

Para que fosse possível animar os modelos, para cada transformação foi criada uma nova versão *animada*: `TranslateAnimated` (2.2.1), `RotateAnimated` (2.2.2), `ScaleAnimated` (2.2.3).

Todas estas recebem o tempo que demoram a completar a animação. Quando uma animação termina recomeça imediatamente.

2.2.1 TranslateAnimated

Esta transformação recebe um conjunto de pontos que definem um caminho pelo qual o modelo deve viajar.

Depois, fazendo uso do número de milissegundos que já passaram desde o início do programa, calcula, usando o algoritmo de *Catmul-Rom*, o ponto no caminho em que o modelo deve estar para esse milissegundo.

2.2.2 RotateAnimated

A rotação animada, ao contrário da sua versão estática, recebe uma duração em vez de receber um ângulo, mantendo os outros parâmetros. Esta duração é mais tarde usada para determinar quanto tempo demora o objeto a efetuar uma rotação de 360° graus.

Depois, simplesmente calcula um ângulo a partir do tempo que passou desde o início do programa para determinar o ângulo para esse instante.

$$angle = \frac{elapsed \times 360}{duracao}$$

Figura 1: Formula de calculo do ângulo da rotação para um dado instante

2.2.3 ScaleAnimated

A escala animada adiciona aos antigos rácios para x, y, z um novo triplo de rácios, e para animar vai, simplesmente, "progredindo" de um rácio para outro durante o tempo que a animação demora. Sendo que, no início e no fim da animação se encontra no primeiro triplo.

$$t = |2 \times \frac{elapsed}{dur} - 1|$$
$$R = (x_i + t(x_f - x_i), y_i + t(y_f - y_i), z_i + t(z_f - z_i))$$

Figura 2: Formula usada para determinar o rácio R da animação para o instante *elapsed*.

2.2.4 Group

O método de desenho recebe agora o tempo que passou desde o início do programa, para que o possa passar às várias transformações, assegurando assim que todas usam o mesmo valor de tempo.

2.3 *Follow mode*

Neste momento os models deslocam-se, o que torna mais complicado de os ver. Assim, para resolver este problema foi implementado o *Follow mode*. Neste o utilizador pode escolher um *model* para a câmara seguir. Sendo que, o comportamento da câmara neste modo é igual ao antigo *explorer mode* apenas com a diferença do centro, em que no modo antigo era na origem do referencial e agora passou a ser objeto.

Para conseguir este efeito, foram feitas duas grandes alterações fundamentais aos módulos *Group* e *Câmara*.

2.3.1 Group

Para poder focar a câmara no objeto é necessário obter a sua posição no espaço 3D. O problema com este requisito é que os *models* em si não têm coordenadas próprias, são transladados, rodados e escalados para as suas posições. Logo, é necessário que o *Group* tenha a capacidade de calcular as coordenadas de cada modelo.

Para este fim, foi implementado um método que dado um índice vai multiplicando todas as matrizes de transformação até chegar ao modelo, e por fim usa a matriz final para extrair as coordenadas do modelo.

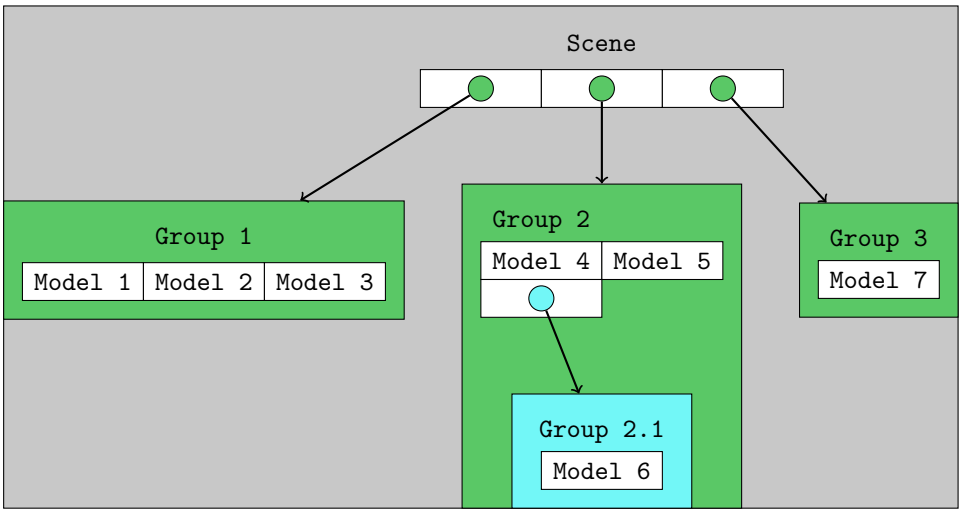

Figura 3: Representação gráfica da numeração dos modelos

2.3.2 Câmara

A câmara, por outro lado, sofreu uma grande reestruturação do seu antigo modo *Explorer*. Foi então concluído que o *Follow mode* era o caso geral deste, ou seja, o *Explorer mode* permitia ao utilizador focar-se na origem enquanto se deslocava na superfície de uma esfera (invisível), podendo-se alterar o raio desta esfera. Por outro lado, o *Follow mode* permite ao utilizador fazer tudo isto, mas focando-se em qualquer ponto do espaço 3D. Desta forma, o *Explorer mode* foi substituído pelo novo, *Follow mode*.

3 Generator

Uma nova funcionalidade foi adicionada ao `generator`. Este é agora capaz de ler *patch files* que definem superfícies `Bezier` e transforma-los nos ficheiros `.3d` que o `engine` está preparado para receber.

Um *patch file* contém duas secções: *patches* e pontos de controlo.

Os pontos de controlo contém triplos de coordenadas que irão ser usados pelos *patches*, enquanto que os *patches* contêm um *patch* por linha.

3.1 Patch

Um *patch* é um conjunto de 16 pontos que definem uma superfície de Bezier (cada linha contém, na verdade, 16 índices que referenciam os pontos de controlo da superfície).

Cada patch destes representa uma superfície e os 16 pontos são divididos em grupos de 4 que descrevem arcos desta superfície. Depois, para cada arco definem-se os pontos da curva usando o nível de *tessellation* para definir quantos pontos criar. Por fim, utilizando estes pontos aplica se o mesmo algoritmo para definir N curvas (uma para cada 4 pontos de cada arco) unindo os 4 arcos e formando uma rede de arcos. Os quadrados desta rede são mais tarde partidos em dois triângulos e serializados como 6 pontos no espaço.

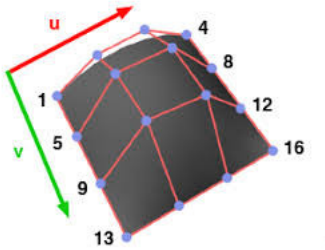

Figura 4: A superfície gerada por pontos de controlo

4 Alterações dos ficheiros de input

O `XML` de input utilizado foi estendido para possibilitar a utilização das novas funcionalidades, no entanto, foi mantida a *backwards compatibility* para o formato anterior. De forma a que modelos antigos continuem a ser validos. Uma ligeira diferença, no entanto, é que o parâmetro `RAND` agora apenas define um cor aleatória para o grupo, ao contrario do comportamento anterior em que desenhava todos os triângulos de todos os seus modelos de cor diferente (ver secção 2.1)

4.1 Translação Animada

Para definir uma translação animada cria-se um bloco *translate* com o parâmetro `TIME`, e dentro deste define se a lista de pontos que a translação ira seguir.

Por exemplo:

```
<translate time="1" >
    <point X="1" Y="1" Z="1"/>
    <point Y="1" />
    <point Z="1" />
    <point X="1.4" />
</translate>
```

Todos os parâmetros podem ser omitidos, pois todos tem valores por defeito: 0 para as coordenadas dos pontos, e 1 para a duração da animação.

4.2 Rotação Animada

Uma rotação animada troca apenas o ângulo por uma duração.

```
<rotate X="1" TIME="1" />
```

Tal como na translação todos os parâmetros podem ser omitidos, com excepção da duração, que é usada para determinar se a rotação é estática ou animada.

4.3 Escala Animada

Uma escala animada tem de receber a escala inicial e a escala final, denominada no `XML` como `XI, YI, ZI` e `XF, YF, ZF` respectivamente. Junto destas espera também uma parâmetro `time` para a duração.

```
<scale XI="20" YI="20" ZI="20" XF="30" YF="30" ZF="30" TIME="1"/>
```

Todos estes tem valores por defeito e podem ser omitidos.

5 Key Bindings

Novas *key bindings* foram adicionadas:

- `Shift+I` e `Shift+O`: Similares ao `I` e `O` permitem diminuir e aumentar o raio do *Follow mode*, mas a uma velocidade mais alta.
- `{` e `}`: Permite seleccionar o modelo a seguir.
- `,`: Liga/Desliga as rotas dos modelos em translação.
- `(` e `)`: Aumenta e Diminui a velocidade com que o tempo passa.
- `P`: Liga/Desliga pausa.

6 Sistema Solar

No Sistema Solar cada um dos astros tem a sua própria órbita e esta é animada, à excepção do sol obviamente. As órbitas dos planetas são órbitas elípticas, tentando assim retratar o sistema solar real. As órbitas das luas de cada planeta são órbitas circulares com inclinações aleatórias.

Além destes astros, foi adicionado também ao sistema solar uma cintura de asteróides que se situa entre Marte e Júpiter e um cometa com uma órbita elíptica muito alongada. O cometa tem a forma de *teapot* para demonstrar também as capacidades do gerador.

7 Conclusões e Trabalho Futuro

Em suma, todas as funcionalidades pedidas foram implementadas assim como alguns extras. No entanto, algumas optimizações não foram implementadas, como por exemplo, o uso de índices no VBOs. Tencionamos adicionar isto na próxima fase.