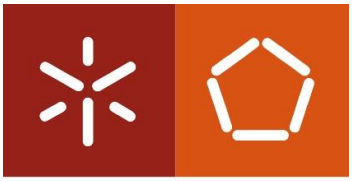

Universidade do Minho

Licenciatura em Ciências da Computação

Computação Gráfica (3º ano de Curso)

Fase 3

Relatório de Desenvolvimento

Diogo Fernandes (A87968) | Luís Guimarães (A87947) | Ivo Lima (A90214)

4 de maio de 2021

Conteúdo

Capítulo 1

Introdução

Nesta terceira fase finalizamos mais uma parte da *engine*, na medida em que possibilitamos aos planetas do nosso Sistema Solar realizarem a sua translação à volta do Sol e ainda incluímos, faltando-nos portanto a última fase do desenvolvimento que será a 4ª fase que pedirá o acrescento de texturas e luzes à *scene*. Como tal, podemos estabelecer um conjunto de tarefas que serão aplicadas tanto no *generator* como na *engine* para suportarem os novos requisitos indicados no enunciado.

Capítulo 2

Atualização do *generator*

2.1 Novas *features*

O engine *gen_Bezier points_Bezier read_Bezier*

Nesta secção pretendemos apresentar por alto tudo aquilo que foi implementado de novo no *generator* sendo um dos novos recursos as *Patches de Bezier*. Uma vez que o programa teve uma nova *feature* tivemos de atualizar o seu método de identificação, pois esta nova implicação tornou a comparação através de uma única letra impossível e confusa (exemplo de conflito: tanto a palavra *box* como *bezier* começavam por um *b*), foi portanto adotado o método de verificar a palavra toda e acresentado um pequeno manual com os parâmetros a utilizador, sendo que este apenasdeve digitar *generator info* aparecendo no seu ecrã as seguintes ajudas:

- plane: generator plane outfile
- box: generator box X Y Z outfile
- sphere: generator sphere radius slices stacks outfile
- cone: generator cone radius height slices stacks outfile
- bezier: generator bezier in-file tessellation-level outfile

Através desta ajuda é facilmente compreendido quais os parâmetros que devem ser dados para cada método for testado.

2.2 Geração dos vértices com os *patches* de Bezier

Pretende-se apresentar o algoritmo adotado sabendo que é dado um conjunto de índices em cada *patch* que mapeiam 16 pontos de controlo. A *tesselation* será uma suavização da superficie, pelo que as divide em pontos intermédios.

Para a criação do *Teapot* foram criadas três funções distintas, a *gen_Bezier*, a *getBezierPoint* e a *read_Bezier*. A *gen_Bezier* é responsável pela junção dos pontos que foram calculados pela *getBezierPoint* que utiliza inicialmente dois polinómios de *Bernstein* tanto para *u*, como para *v*. Tudo isto é possível pois a *read_Bezier* leu todos os pontos de controlo do *teapod.patch* e armazenou-os num vector sendo depois processados os patches através dos indices deste vetor para um novo vetor x que armazena por ordem os pontos de controle de um patch percorrendo um ciclo for que lê todos os patches do ficheiro.

O cálculo do ponto segundo *Bezier* segue a seguinte equação:

$$p(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_i^n(u)\, B_j^m(v)\, K_{ij}$$

Onde *Bu* e *Bv* correspondem aos polinómios de *Bernstein* e *Kij* os pontos de controlo.

Capítulo 3

Atualização da *engine*

3.1 Novas transformações

O ficheiro de configuração do *xml* que temos vindo a utilizar dá suporte a um novo tipo de translações, utilizando curvas (interpolação) de *catmull-rom* , e rotações de modelos para a simulação das órbitas dos planetas ou a rotação dos mesmo sob o seu próprio eixo durante um certo período de tempo.

Temos portanto as seguintes atualizações:

Translações com base em curvas cúbicas de *catmull-rom*, fornecendo:

1. O tempo total para percorrer a curva toda;
2. Um conjunto de pontos de controlo de P0 a Pn com n $\geq$ 4;

Partindo da estrutura seguinte:

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

Para que isto fosse possível, foram adicionadas as seguintes variáveis:

```
struct FIGURE {
  float translation_time;
  float catmull_points[10][3];
  int catmull_points_size;
};
```

Estas variáveis guardam, respetivamente, o tempo de duração da translação (em segundos), os pontos de controlo para a curva de catmull (no máximo 10) e o número de pontos guardados. Na função *renderScene*, antes de desenhar a figura obtem-se o ponto das curvas de catmull e aplica-se a translação associada. Para que a travessia da curva possa demorar *translation_time*, o cálculo dos pontos terá por base um *float* gt, calculado da seguinte forma:

```
fmod(glutGet(GLUT_ELAPSED_TIME), (float)(translation_time *
  1000)) / (translation_time * 1000)
```

Rotações com base no tempo de rotação (360 graus) segundo um eixo:

1. O tempo total, da mesma forma que nas translações;
2. O eixo para orientar a rotação do modelo.

```
<rotate time="..." X="..." Y="..." Z="..."/>
```

Para as rotações foram adicionadas as seguintes variáveis:

```
struct FIGURE {
  float rotation_time;
  float rotation_coordinates[3];
};
```

Estas variáveis guardam, respetivamente, o tempo de duração da rotação(em segundos) e as coordenadas do eixo de rotação. Na função *renderScene*, antes de desenhar a figura aplica-se a rotação no eixo dado por *rotation_coordinates* de ângulo dado por:

```
360 * (fmod(glutGet(GLUT_ELAPSED_TIME), (float)(rotation_time
  * 1000)) / (rotation_time * 1000))
```

3.2 Configuração xml

Com esta implementação, é apenas possível aplicar uma única translação (com base nas curvas de catmull) temporizada e uma única rotação temporizada. Desta forma, no caso de haver conflito, aplicar-se-à a que estiver mais abaixo no ficheiro de configuração.

Uma vez que estas transformações estão implementadas de forma diferente das outras, ao desenhar cada figura, serão primeiro aplicadas estas novas transformações, pela ordem de escrita, e só depois as outras (pela ordem de escrita entre si).

Neste fase, todos os planetas conseguem rodar à volta do Sol. O ficheiro *xmlconfig.xml* foi atualizado de forma que os planetas passassem a orbitar em volta do Sol. Os tempos de rotação foram adicionados estão à escala.

Foi também adicionado um cometa com a forma de um teapot que orbita o Sol mas que possui uma órbita não circular. Neste caso, foram utilizadas as curvas de catmull.

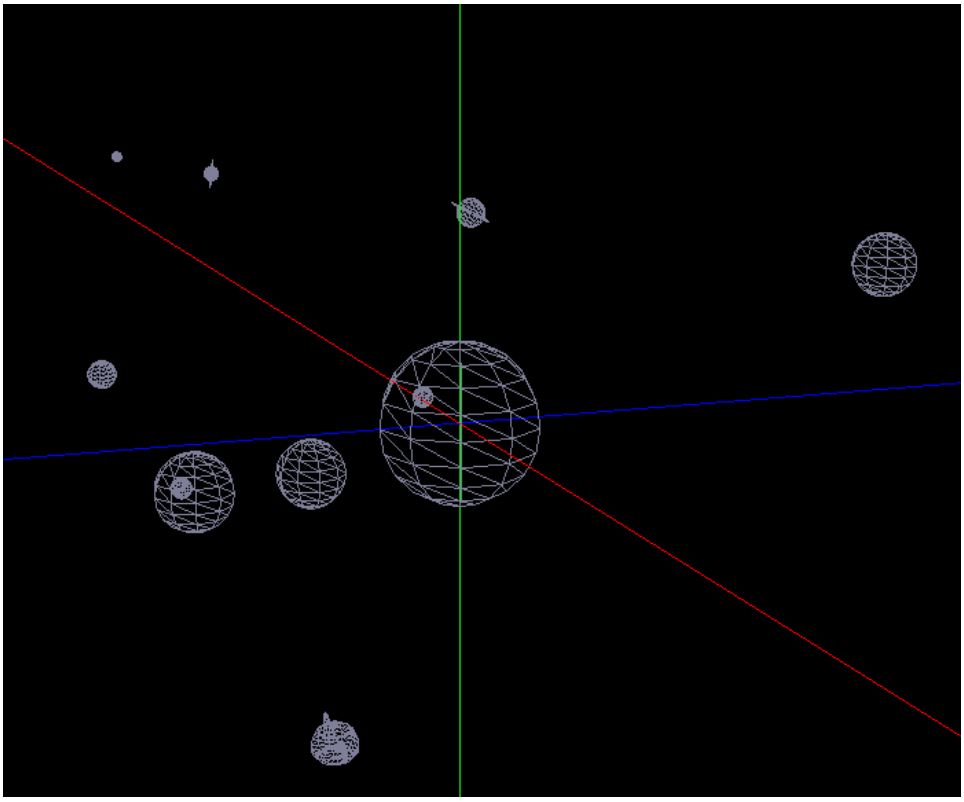

Figura 3.1: Modelo do Sistema Solar

Capítulo 4

Conclusão

Graças a esta terceira fase o nosso projeto ganhou uma maior capacidade de processamento tendo para tal utilizado uma série de conceitos teóricos, que passaram pelas curvas e superfícies de *Bezier* até às interpolações de *Catmull-Rom*.

De facto, com esta fase temos uma nova visão do Sistema Solar, pois o mesmo possui de momento um dinamismo concedido pela rotação e translação dos planetas, sendo que para tal foi necessário calcular os pontos intermédios a partir de valores de controlo, atualizar a configuração do *xml*.

A leitura dos ficheiros patch e a estruturação de uma estratégia para os VBOs com índices foi importante para a consolidação das matérias apresentadas nos guiões práticos.

Em suma, esta fase revelou-se fulcral para o desenvolvimento do projeto, embora esta não tenha sido a fase final, este sistema serve como uma boa base para a última etapa, que pedirá texturas e luzes. Consideramos que os objetivos definidos para a terceira fase foram cumpridos na sua íntegra.