Universidade do Minho

Departamento de Informática

Trabalho Prático

Ciências da Computação - LCC

Computação Gráfica

Grupo 9

Gonçalo Rodrigues A91641

Hugo Sousa A91654

Maio de 2023

Conteúdo

Capítulo 1

Introdução

1.1 Contextualização

Com seguimento no que temos desenvolvido nas fases anteriores chegou o momento de fazer com que o *generator* seja capaz de criar um novo tipo de modelo baseado num ficheiro que contém a informação das curvas de Bezier. Num segundo momento modificamos a *engine* de forma a estender a definição de translação e rotação, o objetivo é que os objetos se movimentam com a trajetória de uma curva repetidamente. O resultado final corresponde a uma cena dinâmica de um sistema solar que inclui todos os planetas, algumas luas e ainda um cometa.

1.1.1 Estrutura do Relatório

Neste relatório será descrita a interpretação efetuada do enunciado do trabalho prático e a abordagem usada para resolver os problemas propostos.

Capítulo 2

Análise e Especificação

2.1 Descrição e Enunciado

2.1.1 Generator

Tendo em conta o enunciado e a fase em que nos encontramos, chega a altura de implementar curvas de bezier. Foi-nos indicado que o nosso *generator* deveria ser capaz de, a partir de um ficheiro *.patch* gerar um ficheiro *.3d* que contenha os triângulos a representar na cena final.

2.1.2 Engine

No caso da *engine* pretendemos nesta fase acrescentar funcionalidades às transformações de rotação e translação. No caso da underlinerotação é agora pretendido que os planetas rodem em torno do seu eixo, e nas underlinetranslações poderem ser usadas curvas cúbicas de Catmull-Rom, com o objetivo de criar movimento.

2.2 Requesitos

2.2.1 Generator

No generator foram adicionadas funções para que a partir de um ficheiro passado como argumento do tipo *.patch*, com informação relacionada com a curvas de bezier, escrevesse num ficheiro do tipo *.3d* os pontos necessários para desenhar um cometa. Para isso foi necessário recorrer aos conhecimentos teóricos da disciplina, mais especificamente à fórmula de bezier para que fosse possível calcular os pontos da curva. O ficheiro final *.3d* contém a lista de triângulos que desejamos desenhar.

2.2.2 Engine

A *engine* deve então agora ser capaz de reconhecer os novos tipos de transformações. São mantidas todas as funcionalidades das fases anteriores, contudo agora é necessário que esta interprete um novo tipo de rotação e underlinetranslação de modo a aplicar animações aos astros, com o objetivo de desenhar um sistema solar.

Capítulo 3

Concepção e Desenho da Resolução

3.1 Transformações geométricas

- **<translate time = t align = "True" />**
  - **<point x= x_1 y= y_1 z= z_1 />**
  - **<point x= x_2 y= y_2 z= z_2 />**
  - **<point x= x_3 y= y_3 z= z_3 />**
  - **...**
  - **<point x= x_4 y= y_4 z= z_4 />**

  Reconhece uma translação sobre uma curvade Catmull-Rom de duração t.

- **<rotate time= t x= x_2 y= y_2 z= z_2 />**

  Reconhece uma rotação de (360/t, x_2, y_2, z_2).

3.2 Curvas de Catmull-Rom

Os pontos de controlo são dados através do ficheiro *XML*, de seguida, utilizando estes mesmos pontos é calculada a posição dos pontos da curva. Por fim, esta é desenhada e transladado um modelo de acordo com os pontos calculados.

Capítulo 4

Codificação

4.1 Código e ferramentas utilizadas

Para a execução deste trabalho foi utilizada a linguagem C++ tanto na construção do *generator* como da *engine*. Para permitir a leitura da configuração do motor foi usada a biblioteca *tinyXML2*.

4.2 Alterações ao generator

Nesta fase foram adicionadas 3 funções. A primeira, formulaBezier, recebe 4 listas de pontos de controlo e aplica a fórmula de bezier estudada nas aulas teóricas retornando a lista com os resultados da aplicação da fórmula aos pontos recebidos. A segunda, bezier, calcula a posição de um ponto na curva de Bezier. Essa curva é formada a partir de um conjunto de pontos de controle que são passados por argumento. Os parâmetros n e m representam valores de t e variam entre 0 e 1. O atributo index é um array que contém os índices dos pontos de controlo e são usados para termos acesso aos pontos de controlo no array points. A função divide os pontos de controlo em grupos de 4 e cada grupo é utilizado para calcular um ponto da curva utilizando a função formulaBezier retornando um novo conjunto de pontos. Por fim é aplicada mais uma vez a função formulaBezier com base nos pontos previamente calculados e o valor de m. A terceira função, drawBezier, vai desenhar uma curva de Bezier usando os dados passados por argumento no ficheiro .patch, para além disso recebe com argumento um inteiro, tessellation. Depois de ler a o ficheiro .patch, a função percorre cada patch e para cada um deles, divide-o em vários quadrados com tessellation pontos em cada lado. Para cada quadrado, a função calcula os quatro pontos de bezier que o definem. Estes pontos são calculados utilizando a função bezier. Por fim os pontos da curva de Bezier são escritos num dinheiro .3d para ser posteriormente escritos. Os pontos são escritos de forma a desenhar dois triângulos por quadrado.

4.3 Alterações à engine

4.3.1 Leitura do ficheiro XML para uma árvore

Para esta fase foi alterado a estrutura do ficheiro com a configuração para poderem ser aplicadas transformações aos modelos, agora cada grupo contém dois valores, o **transform** onde são descritas as transformações e outro **models** onde são descritos os ficheiros a ser lidos.

Foi ainda criado uma classe para guardar as funções que calculam as curvas de Catmull-Rom.

4.3.2 Aplicação das Transformações

Quando uma translação é aplicada, primeiro é testado se existe uma variável "time" associado à mesma, para identificar se é uma translação normal ou uma translação que leva a criação de curvas de Catmull-Rom. Sempre que é o caso desta última é feito o cálculo dos pontos finais da curva pretendida.

Da mesma forma que as translações são distinguidas, as rotações também o são, pois algumas tem um ângulo associado e outras um tempo. Para estas últimas o ângulo é calculado dividindo 360 por o tempo, fazendo assim com que o ângulo seja igual para todas as rotações.

Capítulo 5

Testes

5.1 Sistema Solar

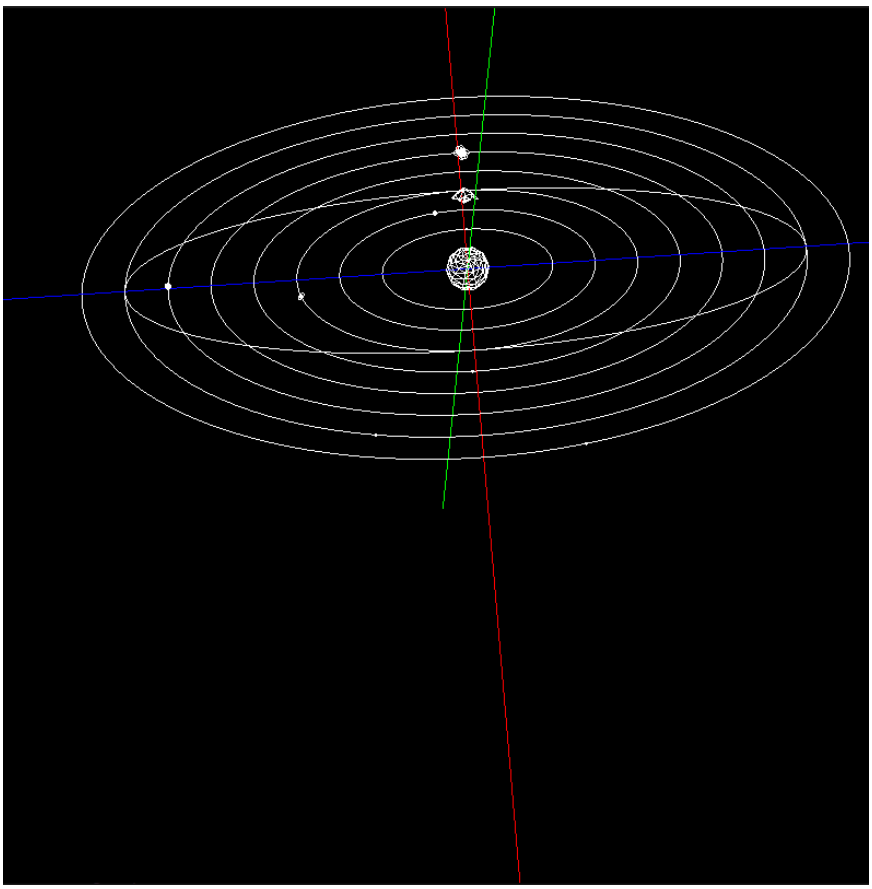

Figura 5.1: Representação do Sistema do Solar 1.

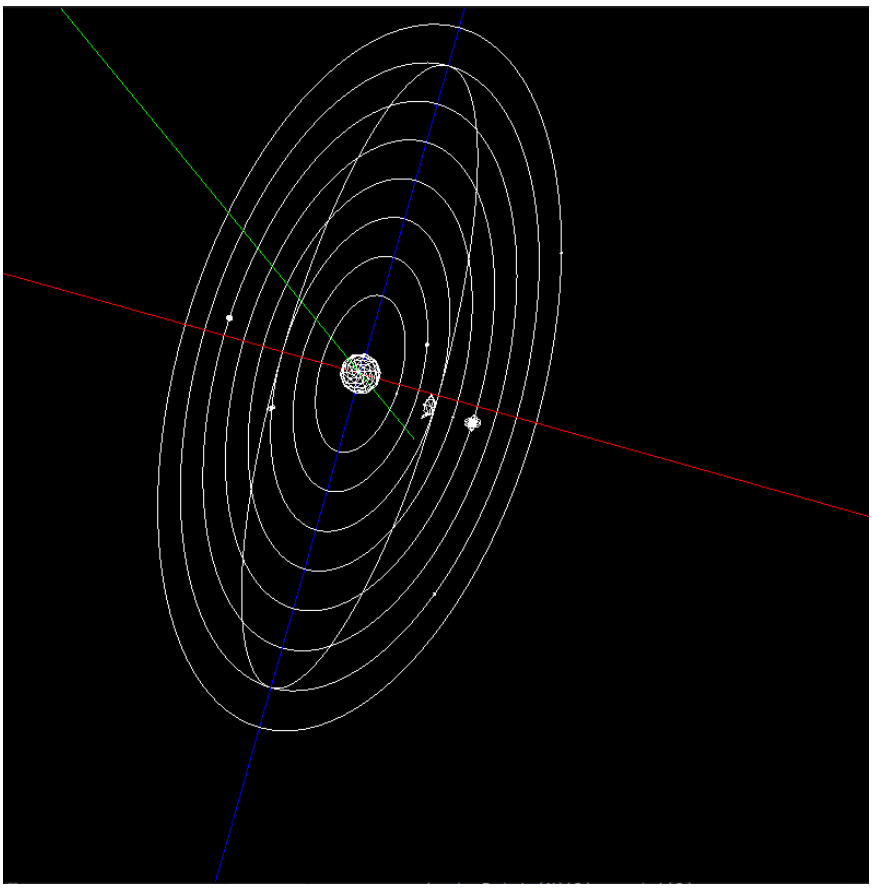

Figura 5.2: Representação do Sistema do Solar 2.

Capítulo 6

Conclusão

Para concluir este relatório, adicionamos nesta fase a capacidade de criação de curvas de bezier no gerador. A engine também se tornou capaz de reconhecer novas transformações, mais especificamente, um novo tipo de rotação e outro de translação. Por fim, tivemos dificuldade em implementar o desalinhamento do objeto nas curvas de Catmull-Rom.