Computação Gráfica

Terceira Fase: Curvas, Superfícies Cúbicas e VBOs

Ciências da Computação

Bruna Alvarães	Igor Braga	Guilherme Santos	Ricardo Teixeira
(A84729)	(A85965)	(A85115)	(A85688)

2 de maio de 2021

Resumo

No seguimento das aulas deste semestre, o seguinte trabalho continua a elaboração de um motor 3D e fornece um modelo do Sistema Solar para mostrar as suas potencialidades.

Conteúdo

Capítulo 1

Introdução

Nesta terceira fase do trabalho, alteramos o nosso ficheiro XML de maneira a acomodar as translações e rotações dos planetas usando curvas de Catmull-Rom. Além disso, usando superfícies de Bézier construímos um cometa que também orbita o sol. Estas são alterações necessárias para conseguir produzir o modelo dinâmico do Sistema Solar.

1.1 Estrutura do Relatório

Neste relatório começamos por analisar o enunciado da terceira fase, e identificamos os requisitos necessários nesta fase. Depois apresentamos a nossa abordagem ao problema. De seguida falamos da implementação do código. Por fim, apresentamos o nosso modelo do Sistema Solar.

Capítulo 2

Análise e Especificação

2.1 Descrição e Enunciado

2.1.1 Gerador

Nesta fase o gerador tem de conseguir criar um novo modelo com base em superfícies de Bézier. Os restantes modelos são desenhados usando VBOs, algo que já foi implementado na fase 2.

2.1.2 Motor

No motor queremos actualizar os elementos de translate e rotate. Para a translação vamos usar curvas cúbicas de Catmull-Rom, o objectivo é criar animações com base nestas curvas. Para a rotação, o objectivo é os planetas rodarem à volta do seu próprio eixo.

2.2 Especificação do Requisitos

2.2.1 Gerador

O gerador irá receber um ficheiro com os pontos de controlo. Depois disto o gerador precisa de ter a capacidade de escrever num ficheiro a lista de triângulos necessários para desenhar o cometa.

2.2.2 Motor

O motor depois de ler o ficheiro XML, além de desenhar os vários modelos com as transformações pela ordem correta, tem de aplicar as diversas animações aos vários astros de maneira a desenhar correctamente o sistema solar.

Capítulo 3

Concepção/desenho da Resolução

3.1 Abordagem ao Problema

Antes de começarmos a implementar a leitura de ficheiros do formato *.patch* e da implementação dos novos tipos de *rotate* e *translate*, decidimos que era melhor fazer uma reestruturação do código de ambos o gerador e o motor, tendo colocado as funções relevantes em ficheiros separados, para uma melhor organização.

No gerador, isto traduziu-se em ficheiros que contêm as funções relevantes de cada figura no seu próprio ficheiro.

No motor, esta reestruturação incluiu uma mudança na forma como implementamos as transformações e as guardamos para as aplicarmos no devido momento.

3.1.1 Superficies de Bézier

A partir do ficheiro *.patch* fornecido na Blackboard, iremos ler os pontos de controlo da curva, criar novas curvas a partir dessas e por fim obter os pontos necessários para a criação de um patch de Bézier.

3.1.2 Curvas Catmull-Rom

Iremos obter os pontos relevantes das curvas lendo os pontos de controlo do nosso *XML*, calculamos as suas posições e desenhamos as curvas e transladamos os modelos de acordo com esses pontos.

Capítulo 4

Codificação e Testes

4.1 Código e Ferramentas Utilizadas

Para obter os diversos pontos da trajectória de translação dos vários planetas, criámos um programa em *Phyton3* que partindo da equação da elipse nos dá pontos na circunferência da mesma espaçados igualmente entre si. Este programa já dá output dos pontos formatados de acordo com o nosso .XML .

```
import math

def PontosElipse(r,ang,flag,a,b):
    i = 0
    while(i<360):
        if (flag < 0):
            aux = 360-i
        else:
            aux = i

        x =  a + r * math.cos(math.radians(aux))
        z = b + 2 * r * math.sin(math.radians(aux))
        y = 0

        print("<point X=\"" + str(round(x,9)) + "\" Y =\""+ str(round(y,9)) +"\" Z=\"" + str(round(z,9)) + "\" />")

        i = i + ang
    return
```

4.2 Alterações ao Gerador

4.2.1 Superficies de Bézier

É lido o ficheiro *.patch* dado como argumento, onde iremos guardar um vetor de vetores de vetores de floats. Isto é, iremos ter um vetor de tamanho igual ao número de patches, em que cada indíce desse vetor irá conter um vetor de tamanho 16 , equivalente ao número de pontos de controlo por cada patch de Bèzier. Cada um desses vetores que agora equivale a um ponto de controlo tem em si o vetor de coordenadas do ponto, ficando neste formato: *[Patch][Ponto][Coordenadas]*.
Com os pontos e as suas coordenadas guardadas é uma questão puramente matemática agora. Guardando os pontos 4 a 4 e aplicando a fórmula de Bèzier, iremos criar as curvas necessárias para a criação do patch de acordo com a *tessellation* pedida e guardamos os seus pontos. Estes pontos são escritos para um ficheiro para ser posteriormente lido pelo motor.

4.3 Alterações ao Motor

4.3.1 Curvas Catmull-Rom

Pontos de Controlo para as Curvas de Catmull-Rom

Obtendo os pontos com o nosso programa , estes são lidos pelo nosso parser de XML e guardados num vector de floats.

Translações

Durante a leitura da árvore de grupos que utilizamos para manter a hieraquia dos grupos, aplicamos as transformações de cada grupo em conjunto. Isso inclui a distinção entre um Translate simples e um Translate que leva a criação de Curvas de Catmull-Rom. Quando esta última translação é detetada através da presença da variável 'time', começamos o cálculo dos pontos finais da curva pretendida. Obtemos os pontos para a translação das figuras e estas são aplicadas.

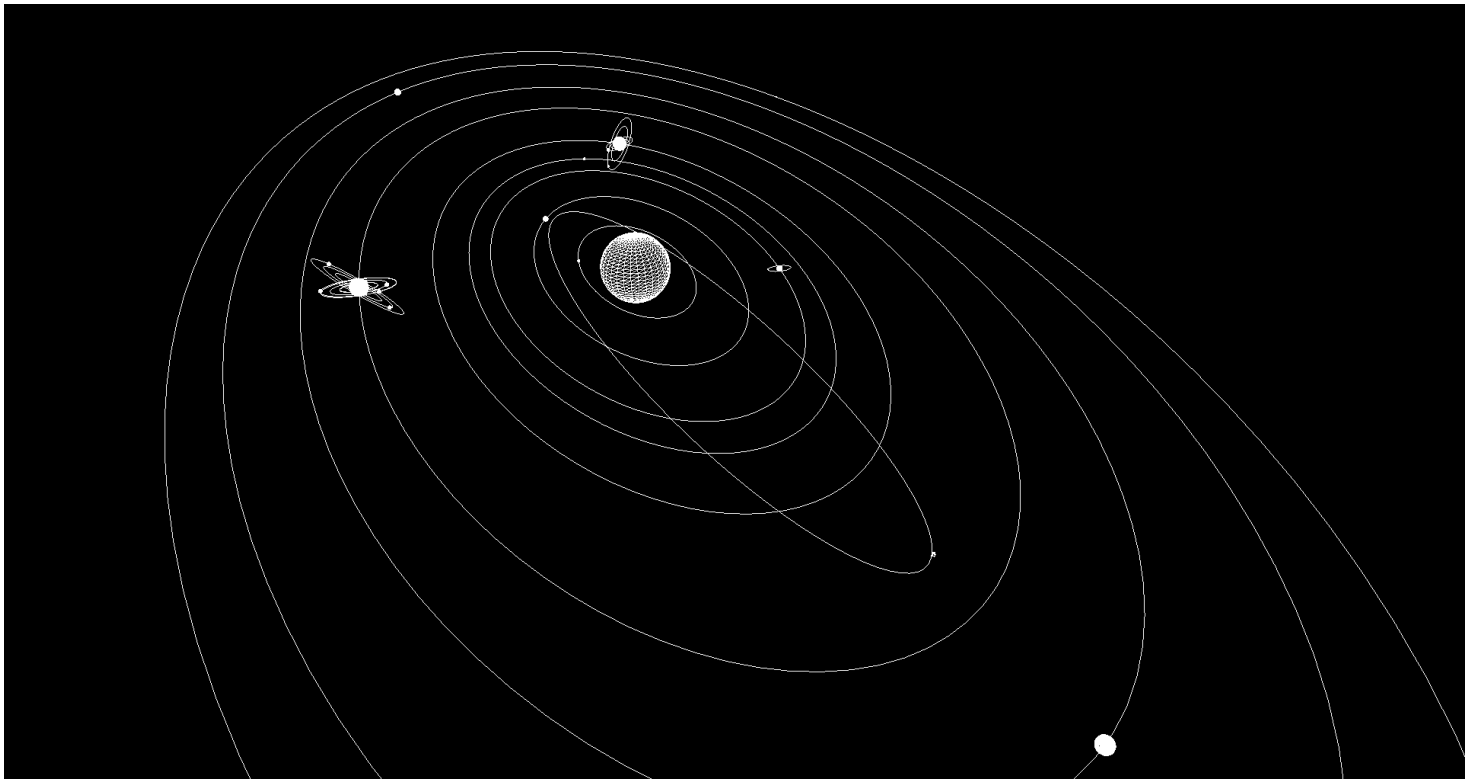
Rotações

Da mesma maneira que as translações são detetadas, se a variável 'time' se encontrar numa rotação, o ângulo é calculado a partir da divisão de 360 pelo tempo, assim temos o mesmo ângulo para todas as rotações.

4.4 Testes realizados e Resultados

Estes foram os resultados obtidos:

4.4.1 Sistema Solar



Capítulo 5

Conclusão

A realização desta terceira fase ajudou-nos a por em prática os conhecimentos que aprendemos nas aulas.