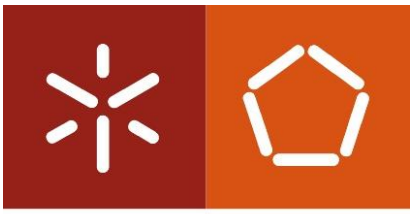

Universidade do Minho

Licenciatura em Engenharia Informática

Computação Gráfica

Fase 3 - Curves, Surfaces and VBOs

Grupo 21

Duarte Parente (A95844) Gonçalo Pereira (A96849)

José Moreira (A95522) Santiago Domingues (A96886)

Ano Letivo 2022/2023

Índice

Capítulo 1

Introdução

O presente relatório visa apresentar a terceira e penúltima fase do projeto a desenvolver no âmbito da Unidade Curricular de Computação Gráfica. Ao longo do documento irá ser explicada a metodologia adotada para a resolução dos problemas, assim como as diversas opções tomadas ao longo desta fase.

O principal objetivo esta fase prendeu-se na implementação de curvas, superfícies cúbicas e VBOs, de forma a criar uma representação dinâmica do sistema solar já desenvolvido anteriormente, conferindo-lhe então um maior grau de realidade.

Contrariamente à fase anterior, foi necessário realizar alterações em ambas as aplicações. No `Generator` foi adicionado o suporte de uma nova primitiva, construída a partir de um conjunto de patches de *Bezier* e o nível de tesselação. Em relação ao `Engine` e de forma semelhante às restantes fases foi necessário atualizar o parser para suportar as novas extensões do ficheiro de configurações. As transformações de translação e rotação foram atualizadas para permitir a animação de curvas Catmull-Rom e rotação de $360^{\underline{o}}$ graus em torno de um eixo, havendo a possibilidade de manipulação da velocidade de duração destas novas funcionalidades, e também do alinhamento do objeto pela curva no caso da translação.

Por fim, procedeu-se finalmente à implementação de VBOs, definido como meta futura na segunda fase, o que melhorou significativamente a qualidade do cenário apresentado.

Capítulo 2

Generator

2.1 Superfícies de Bezier

Como proposto na secção da atual fase do trabalho prático, introduziu-se um novo modelo baseado em Superfícies de Bezier. Com o desenvolvimento destas superfícies, surgiu a necessidade de escrever código que permitisse a realização de cálculos entre matrizes de valores númericos, envolvendo, também, matrizes de pontos.

Inicialmente, houve a necessidade de compreender, integralmente, toda a teoria relativa a curvas e superfícies de Bezier, estudando, portanto, os conceitos de curva, patch, matriz, tesselação, etc. Posteriormente, aquando a implementação, nasceu a ideia de desenvolver dois ficheiros distintos, onde, no primeiro, seria escrito todo o código correspondente à classe "Bezier"e, no segundo, todas as funções responsáveis por parsing e posterior cálculo entre matrizes. Seguindo a lógica proposta no enunciado, todos as patches e pontos de controlo encontram-se escritos num ficheiro, numa pasta à parte, denominada "patches". O algoritmo desenvolvido permite a leitura desse mesmo ficheiro e armazenamento do conteúdo útil em estruturas matriciais onde, posteriormente, são aplicados cálculos, de forma a que os pontos da superfície sejam gerados e, à semelhança de todos os modelos projetados até à data, escritos num ficheiro.

Capítulo 3

Engine

Nesta fase do projeto foi necessário atualizar o `Engine` de forma a suportar as novas funcionalidades de translação e rotação para além da obrigação de desenhar os modelos através de VBOs.

3.1 Alterações ao Ficheiro de Configuração

As novas funcionalidades já enunciadas foram traduzidas em novas extensões XML. Enquanto que anteriormente as transformações de rotação e translação eram descritas da seguinte forma:

```
<translate x="0" y="3" z="0" />
<rotate angle="90" x="1" y="0" z="0" />
```

nesta fase foram acrescentadas as seguintes versões:

```
<translate time = "10" align="true">
    <point x = "0" y = "0" z = "4" />
    <point x = "4" y = "0" z = "0" />
    <point x = "0" y = "0" z = "-4" />
    <point x = "-4" y = "10" z = "0" />
</translate>
<rotate angle="-90" x="1" y="0" z="0" />
```

Na extensão `translate` o atributo `time` traduz o tempo de duração de uma volta completa ao longo da curva, enquanto que o a flag `align` controla se o objeto deverá ser, ou não, orientado na curva. São ainda fornecidos no mínimo 4 pontos para a formação da curva de Catmull-Rom.

Ao contrário da fase anterior estas alterações não se traduziram na necessidade de grandes alterações estruturais no parser.

3.2 Implementação de VBOs

Para além de se apresentar como um dos requisitos para a realização desta fase, a implementação de VBOs foi também uma meta estabelecida pelo grupo para esta etapa. Esta alteração permitiu aumentar significativamente a eficiência do programa na renderização dos cenários, para além da experiência proporcionada ao utilizador.

Procedeu-se à criação de uma classe que irá representar um VBO que ficará associado a um modelo.

```
class VBO{
    public:
        GLuint vertixes;
        int vertixCount;
        GLuint indexes;
        int indexCount;
}
```

A esta estrutura ficará associado o vetor de pontos do modelo (vertixes) assim como o vetor de indíces (indexes). O acesso e armazenamento dos VBOs em associação com o respetico modelo foi facilitado pelo uso do `std::unordered_map` que se apresenta como uma estrutura semelhante a uma Hash Table.

3.3 Implementação das novas Transformações

3.3.1 Translação

Para a implementação da nova possibilidade de Translação foi necessário recorrer a curvas de Catmull-Rom para delinear a trajetória a percorrer pela primitiva visada, tendo em consideração o tempo indicado para a duração de uma volta completa assim como a possibilidade de orientação da primitica na curva.

Isto levou à reestruturação da classe `Translate` que passa a realizar internamente as operações necessárias à construção da curva de Catmull-Rom.

```
class Translate : public Transformation {
    public:
        Point t;        // Usado se for a extensão for do tipo antigo
        float time;
        int flag_g;  // Flag do tipo da translação, 0 = Catmull-Rom , 1 = Antiga
        int flag_align;  // Flag de orientação da primitiva
        static int flag_curves;  // Flag para desenho das curvas
        int point_count;
        vector<Point> cat_points; // Pontos para a construção da curva
}
```

Para o controlo do tempo de duração é usada a função `glutGet(GLUT_ELAPSED_TIME)` tal como advertido no enunciado.

```
float t = ((float)glutGet(GLUT_ELAPSED_TIME) / 1000) / ((float)time);

/* Variàvel t que será passada à função que calcula os pontos de
    Catmull-Rom.   */
```

3.3.2 Rotação

Já em relação à transformação de rotação não foi necessário proceder a tais alterações, sendo apenas necessário detetar se o atributo presente na extensão se tratava de `angle` ou `time`, agindo em concordância com cada um dos casos.

Tal como na transformação acima o controlo do tempo de uma rotação completa é efetuado através da `glutGet(GLUT_ELAPSED_TIME)`.

```
angle = ((float)glutGet(GLUT_ELAPSED_TIME) * 360 / 1000) / ((float)time);

/* Variàvel angle que será passada à função que efetua a rotação
    em concreto (glRotatef). */
```

Capítulo 4

Análise de Resultados

Neste capítulo serão apresentados os resultados do programa quando confrontado com os testes e respetivos ficheiros de configuração disponibilizados pela equipa docente. Foram disponibilizados 2 testes, cada um focando em vários aspetos definidos para esta fase.

Seguem-se representados os resultados obtidos com o programa desenvolvido (metade esquerda da figura), e respetiva comparação com o resultado esperado (metade direita da figura).

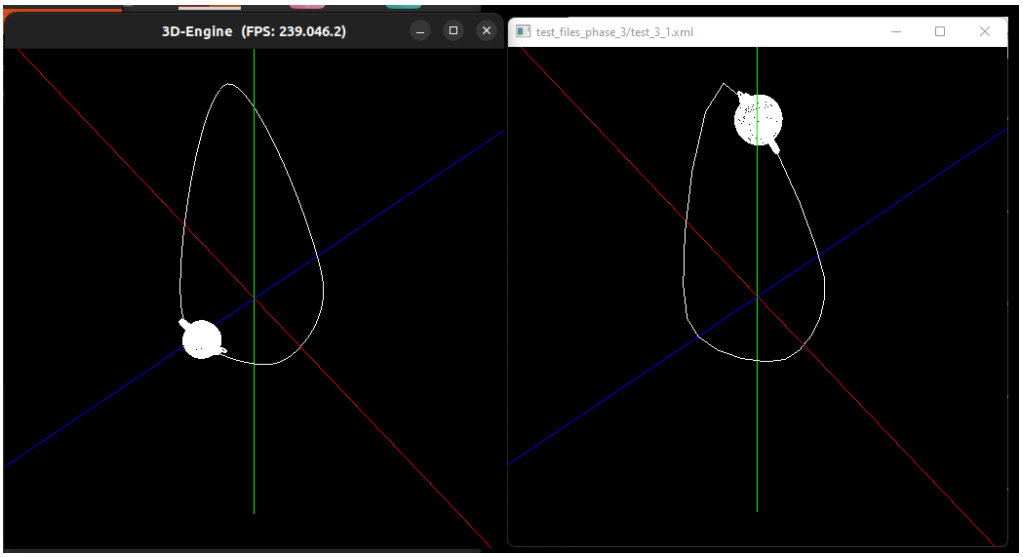

Figura 1: Teste 1

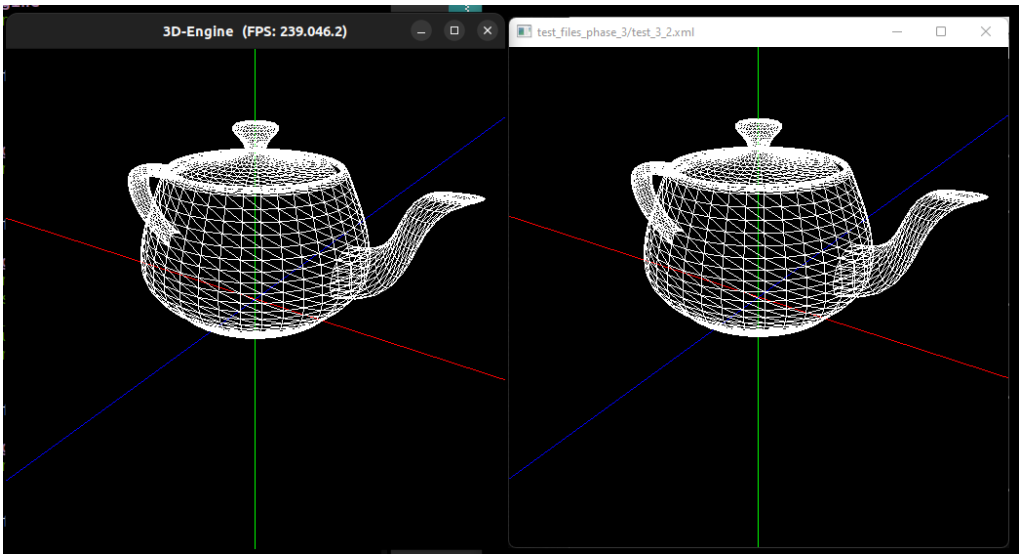

Figura 2: Teste 2

De notar que o resultado do primeiro teste não poderá ser totalmente comprovado através deste relatório dada a impossibilidade de comprovar a rotação da curva e a translação do teapot ao longo da mesma. No entanto através da consulta do vídeo disponibilizado pelo corpo docente foi possível comprovar que este correspondeu com o comportamento esperado.

Capítulo 5

Sistema Solar

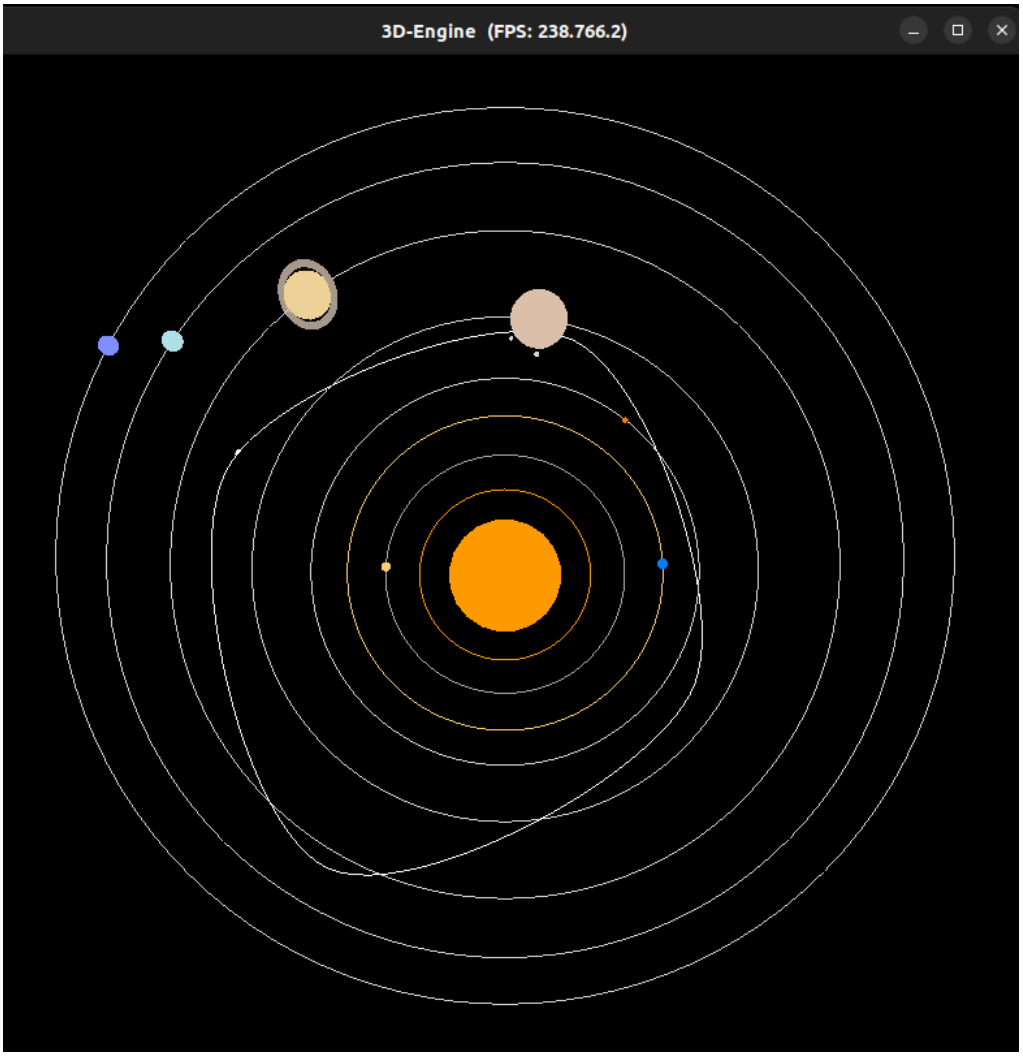

Figura 3: Representação do Sistema Solar com as órbitas

Para esta terceira fase a representação do sistema solar teria de passar a ser representada de forma dinâmica, ao contrário da versão estática desenvolvida na fase anterior. Sendo assim, fizemos uso das curvas de Catmull-Rom para definir as órbitas de cada um dos planetas tal como se encontra representado na figura acima. Para além disso estas serviram também para definir a órbita do cometa, que era também um dos requisitos para esta penúltima etapa. De notar que o cometa foi desenhado usando as superficíes de Bezier, em particular o teapot fornecido pelo corpo docente.

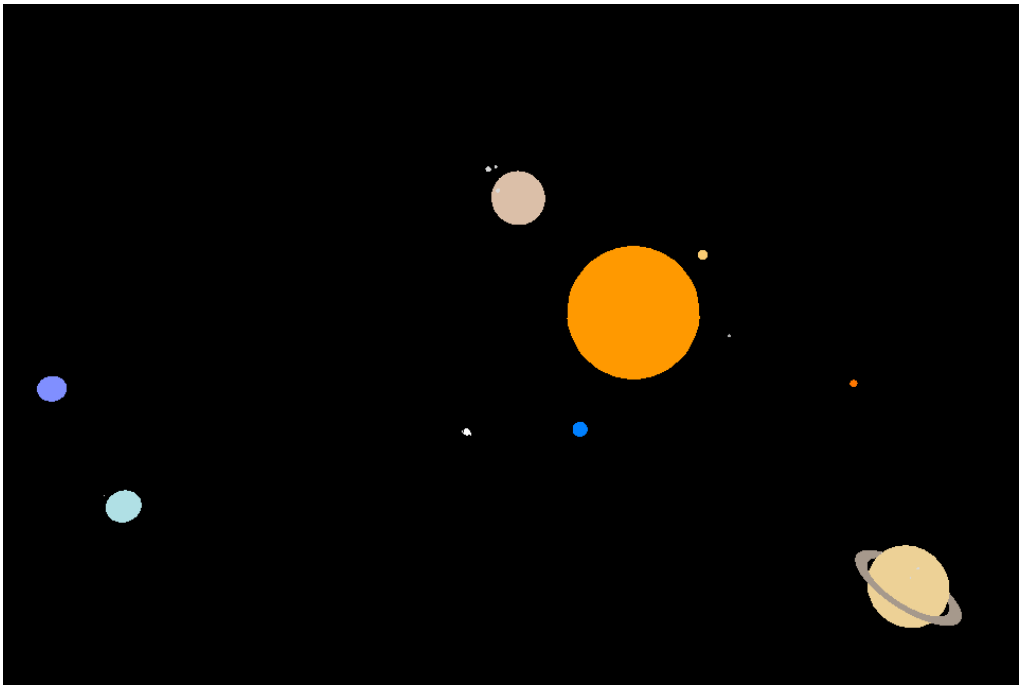

Figura 4: Representação do Sistema Solar

De forma a representar realisticamente o movimento de rotação dos planetas, definimos um valor base para o movimento de rotação de Mercúrio, 600 segundos que correspondem a 88 dias. Após isso, calculamos o valor de translação em proporção com o seu valor real.

Capítulo 6

Conclusão

Dado por terminado o trabalho, conclui-se que foi respondido com sucesso o principal objetivo do trabalho.

Embora se tenham verificado algumas dificuldade referentes à aplicação dos *patches de Bezier*, na construção do novo tipo de primitiva geométrica do Generator e no desenvolvimento das curvas de Catmull-Rom, é possível concluir que estas foram ultrapassadas. Além disso, o grupo conseguiu ser bastante proativo na resolução dos desafios encontrados.

Após alguma ponderação do grupo, decidiu-se implementar um mecanismo para ocultar ou revelar a renderização das curvas de Catmull-Rom, permitindo dar certa realidade à representação do Sistema Solar.

É expectável que o projeto em curso possa sofrer alterações de forma a dar continuidade ao aperfeiçoamento do Sistema Solar, nomeadamente, a adição da cintura de asteroides.