Computação Gráfica
Fase 3

Universidade do Minho
Escola de Engenharia

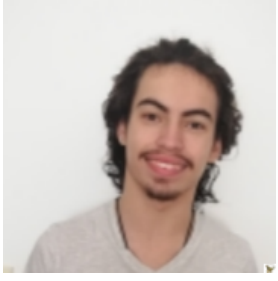
Trabalho realizado por:

Afonso Laureano Barros Amorim
Gonçalo Martins dos Santos
João Carlos Fernandes Novais
Telmo José Pereira Maciel

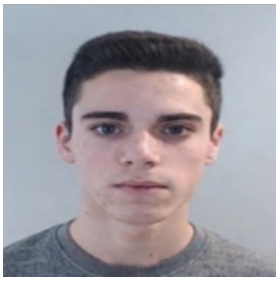
Grupo 26

A97569
Afonso Amorim

A95354
Gonçalo Santos



A96626
João Novais



A96569
Telmo Maciel

Índice

1 Introdução

Na terceira fase do trabalho prático de Computação Gráfica, foi proposto alterar o Generator de maneira a que implemente um novo tipo de modelo baseado em remendos de Bezier (conhecidos normalmente como *Bezier Patches*).

Também foi proposto alterar o Engine de maneira a suportar translações e rotações dinâmicas usando curvas de Catmull-Rom, e de forma a que o desenho dos modelos seja feito com *VBO*s.

Tivemos ainda que desenvolver um modelo dinâmico para o Sistema Solar incluindo um cometa cuja a trajetória é definida por uma curva de Catmull-Rom.

2 Alterações na arquitetura do projeto

Nesta parte do relatório, iremos resumir as alterações introduzidas na arquitetura do projeto para cumprir com o que era pedido.

2.1 Common

2.1.1 MatrixOpp.cpp

Este ficheiro foi adicionado de forma a permitir fazer as operações matemáticas sobre matrizes que necessitamos para calcular os pontos necessários para gerar os modelos através dos ficheiros .patch. Essas operações vão desde o produto vetorial de vetores quer de floats quer de *Point*s, até a multiplicação de matrizes.

2.1.2 Classe Shape

De forma a auxliar o desenho por *VBO*s decidimos adicionar dois atributos a esta classe, chamados *vboStartIndex*, que representa o índice onde aparece o seu modelo no *VBO*, e *vboStopIndex*, que representa o índice onde começa o próximo modelo no *VBO*.

2.1.3 Classe Point

No que toca a esta classe, decidimos sobrecarregar os métodos * (com float) e + (com *Point*) para facilitar as operações sobre matrizes de objetos desta classe.

2.2 Generator

Nesta fase do trabalho, o Generator foi alterado para ser capaz de conveter um ficheiro com Bezier patches (extensão patch) num ficheiro de extensão 3d com todos os pontos necessários para desenhar a figura segundo um certo nível de tesselação.

2.2.1 Bezier.cpp

Ao generator, adicionamos o ficheiro **Bezier.cpp** que serve para gerar ficheiros da extensão 3d através de ficheiros da extensão patch. Passemos para a explicação da maneira de como esta conversão foi implementada.

Começa-se por fazer o parsing do ficheiro, guardando os índices dos patches num *vector* que contém *vector*s de inteiros. Os pontos de controlo são guardados num vetor.

A seguir, temos que calcular os pontos da curva de Bezier, percorrendo os vários patches guardados. Tendo em posse os pontos de controlo, inicializamos as matrizes de Bezier para estes (P e MxPxMT). Para gerar a grelha, demos valores às variáveis u e v, incrementando de acordo com o nível de tesselação e, em conjunto com as matrizes calculadas antes, podemos obter os pontos finais. Com esta grelha, podemos dividir cada um dos quadrados em dois triângulos e guardar esses pontos no ficheiro de extensão 3d.

2.3 Engine

No Engine, foi adicionada a renderização com os VBO's e ainda foi implementada a alteração de modo de visualização entre GL_LINES ou GL_SOLID, usando a tecla 'c'.

2.3.1 Parser.cpp

Tendo em conta que agora as translações e rotações são dinâmicas, o parser teve que sofrer umas alterações na maneira como realiza o seu parse nas transformações, havendo

agora uma função para o tratamento de translações (*parseTranslation*) e outra para o tratamento de rotações (*parseRotation*).

2.3.2 Classe Group

Na classe Group, adicionamos um array que utiliza memória estática, com espaço para 3 floats, que representa o antigo eixo y (y_0), usado para alinhar o grupo caso ele esteja numa translação dinâmica.

2.3.3 Classe Curve

Adicionamos a classe Curve (subclasse de Transformation), que representa uma translação dinâmica sobre uma curva Catmull-Rom, e que usamos para retornar o ponto a que um modelo se encontra na mesma num determinado instante temporal.

2.3.4 Classe Transformation

Na classe Transformation, adicionamos 2 tipos de transformações: TimeTranslate e TimeRotate e ainda um campo do tipo float chamado time.

3 Adição dos VBO

Tal como foi dito na introdução, é pedido que nesta fase os modelos sejam desenhados através de *VBO*s, e esta secção tem como objetivo explicar a estratégia que seguimos para o fazer.

Adicionamos uma função chamada *buildVBO*. Nesta função começamos por definir um *std::vector<float>*, que vai conter a informação que será colocada na memória gráfica no *VBO*. De seguida iteramos sobre todos os modelos que estarão na cena e, caso os pontos necessários para o desenhar não estejam no *VBO*, estes são adicionados ao mesmo, e definimos o índice de início e de fim. Após isso colocamos esses indíces nas diferentes instâncias da classe *Shape*, e acabamos por colocar a informação no *VBO*

4 Alterações no cenário final

Para esta fase decidimos reduzir a distância a que cada planeta está do Sol para metade, e aumentamos a escala de cada um de forma a que sejam mais visíveis.

Fizemos ainda com que cada planeta rodasse sobre o seu próprio eixo e à volta do Sol, com uma escala temporal em que 1 décimo de segundo equivale a 1 dia (por exemplo, a órbita da Terra à volta do Sol demora 36.5 segundos).

Adicionamos ainda um cometa, que tem como modelo um bule, obtido através de curvas de Bezier.

Abaixo vamos deixar umas imagens do cenário e ainda um vídeo de demonstração.

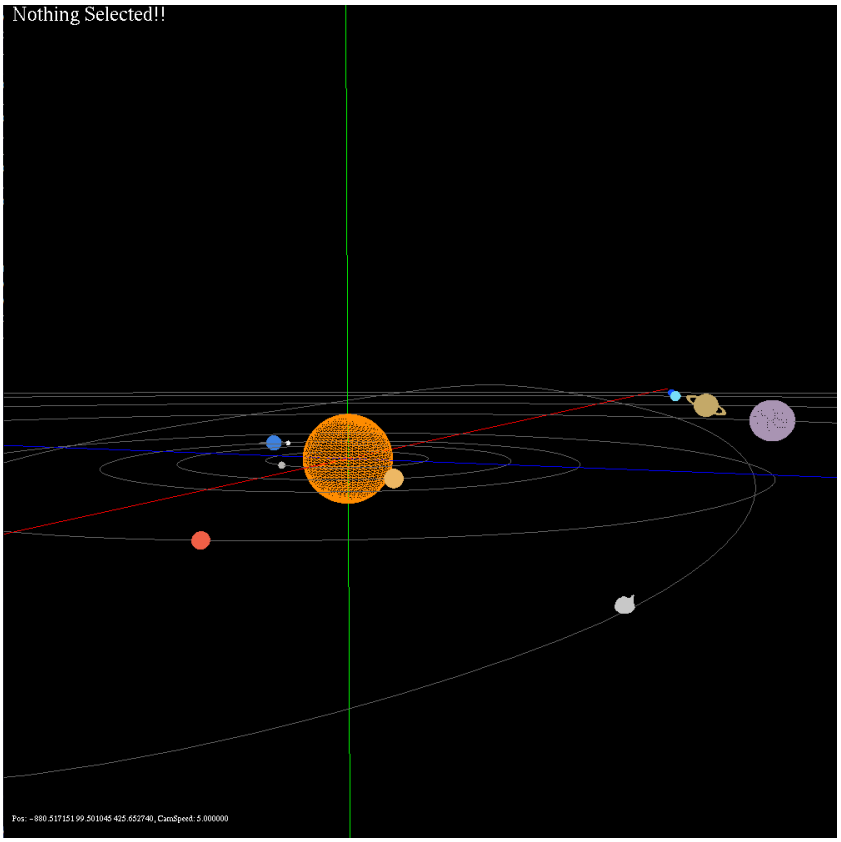


Figure 1: Cenário visto de longe

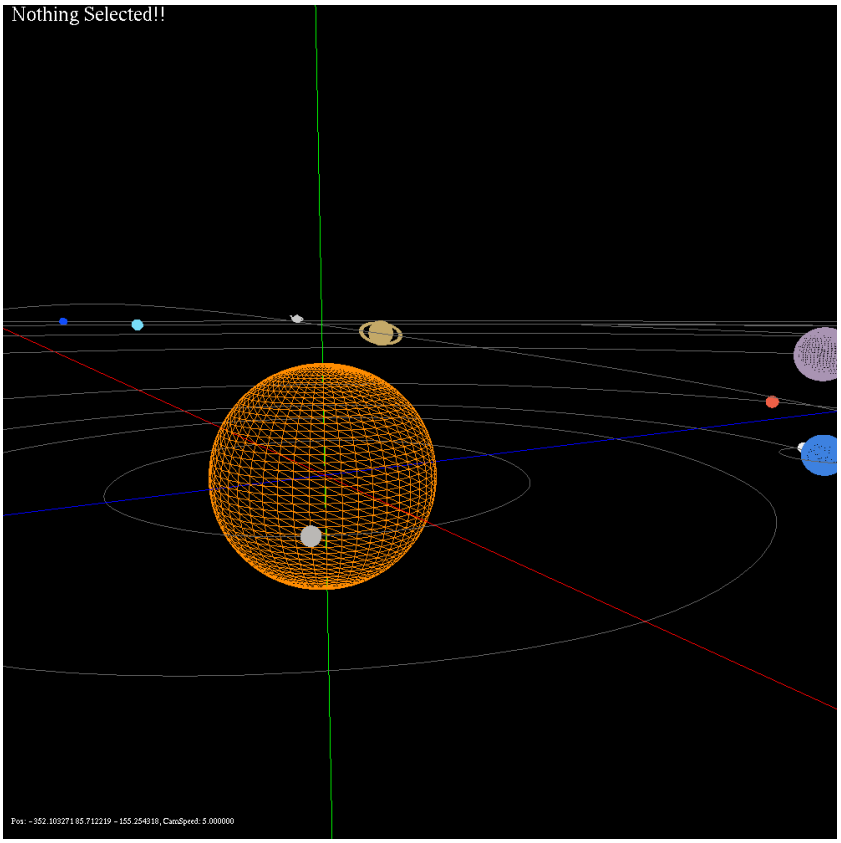


Figure 2: Cenário visto de perto

5 Conclusão

Com a terceira fase do projeto concluída, conseguimos aplicar os conceitos lecionados nas aulas sobre curvas e superfícies cúbicas. Ainda tivemos a oportunidade de aplicar no trabalho os VBO's, conceito que já tinha sido usado e lecionado em aulas práticas.

Esperamos que, o trabalho desenvolvido nesta fase, sirva como uma ótima base para a implementação da fase final do projeto de Computação Gráfica.