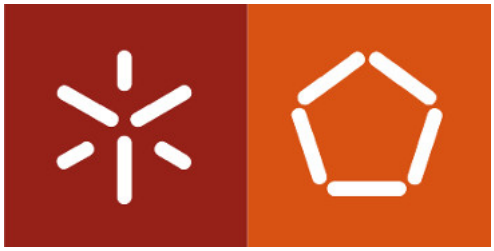


Universidade do Minho
Escola de Engenharia
Mestrado Integrado em Engenharia Informática

Unidade Curricular de Computação Gráfica

Ano Letivo de 2022/2023

Fase 3 – Curvas, Superfícies Cúbicas e VBOs

Grupo 6

A96434	Francisca Quintas Monteiro de Barros
A97588	Joana Isabel Freitas Pereira
A94870	Rafael Picão Ferreira Correia

5 de maio de 2023

Índice

Lista de Figuras

1 Introdução

O presente relatório é referente à terceira fase do trabalho prático desenvolvido pelo grupo 06 no âmbito da Unidade Curricular de Computação Gráfica, lecionada no curso de Licenciatura em Engenharia Informática no 2º Semestre do ano letivo 2022/2023.

Este projeto tem como objetivo o desenvolvimento de um mecanisno 3D baseado num cenário gráfico, através de ferramentas como o OpenGL e o C++, também usadas nas aulas práticas.

O projeto está dividido em 4 fases e para cada fase, serão fornecidos arquivos de configuração XML para testes e avaliação.

Nesta terceira fase o grupo fez algumas alterações tanto no *engine* como no *generator* de modo a suportar novas funcionalidades. No *engine*, foi substituído o método de desenho instantâneo por Vertex Buffer Objects (VBOs), o que resultou num melhor desempenho e maior eficiência na renderização dos modelos. Além disso, o grupo implementou funções de rotação e translação que decorrem ao longo do tempo, por sua vez criando animações. Foi ainda aprimorada a leitura dos arquivos de modelos, de forma a que eles sejam lidos apenas uma vez, o que reduz a carga no sistema e melhora ainda mais o desempenho.

No *generator*, foi adicionado o suporte para criar modelos com *Bezier patches*, permitindo a criação de superfícies curvas mais complexas e detalhadas.

Por fim, as melhorias foram implementadas no sistema solar, adicionando os novos recursos de rotação e translação aos planetas e criando um cometa que pode ser visto movendo-se pela sua órbita. Essas adições aumentaram o realismo e a interatividade da simulação.

2 Generator

Nesta 3ª fase, o programa do *generator* foi alterado de modo a suportar um novo tipo de modelos, modelos baseados em superfícies de *Bezier*. A criação desses modelos requer a definição de dois parâmetros: o nível de tesselação e o caminho para o ficheiro de configuração do modelo. Os ficheiros de configuração têm uma estrutura específica que deve ser seguida para gerar corretamente o modelo:

- A primeira linha do arquivo indica o número de patches (npatches) definidos no ficheiro;
- A npatches linhas seguintes têm 16 números cada uma e representam os índices dos pontos, mais a baixo definidos, que vão constituir aquele patch;
- A linha que se segue indica o número de pontos (npontos) definidos no ficheiro
- As últimas npontos linhas têm 3 floats e representam cada uma um ponto

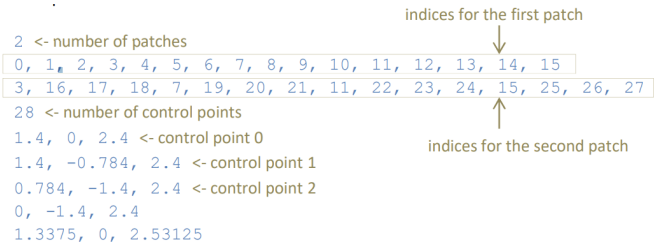


Figura 2.1: Exemplo da configuração de um ficheiro

A primeira fase para a construção deste tipo de modelos é fazer o parse do ficheiro de configuração. A primeira leitura feita do ficheiro de configuração é do número de patches. Com esta informação é criado uma matriz com 16 colunas e o número de linhas correspondente ao número de patches dado anteriormente, visto que se sabe que cada path é constituído por 16 pontos. O array é preenchido com os índices correspondentes, e, de seguida é seguida uma estratégia semelhante para os pontos.

Após terem sido obtidos todos os pontos e índices a partir do ficheiro de configuração, é iniciado o processo de cálculo dos pontos que compõem a superfície de *Bezier*. Para cada patch, são utilizados dois inteiros que variam de 0 até ao nível de tesselação e uma função que calcula o ponto com base nesses dois inteiros e nos pontos do patch. É desta forma que os triângulos da superfície são criados. Contudo, atualmente o código realiza cálculos repetidos, tanto na multiplicação da matriz M com a matriz dos pontos para cada patch, como no cálculo de pontos que poderiam ser guardados e reutilizados para criar triângulos.

3 Engine

Nesta terceira fase do projeto, o *Engine* sofreu algumas alterações de modo a suportar VBOs e novas configurações de XML. De seguida mostramos em quais classes fizemos alterações, explicando-as.

3.0.1 Translação

Com os aprimoramentos na funcionalidade de translação, o programa agora oferece duas opções de configuração. A primeira opção é a configuração antiga, em que apenas as coordenadas x, y e z eram especificadas. A segunda opção já tem mais parâmetros especificados de modo a permitir animações.

Com o uso do parâmetro *time*, a translação permite a definição do tempo que o(s) objeto(s) levarão para percorrer a trajetória. Além disso, foi introduzido o parâmetro *align*, que determina o alinhamento do objeto com a direção das derivadas da trajetória.

Após a definição desses parâmetros, são especificados múltiplos pontos aninhados (no mínimo 4), de modo a definir as curvas cúbicas *Catmull-Rom* que a trajetória irá seguir.

Essa nova configuração expande significativamente as possibilidades de animação e movimento dos objetos.

```
...
<translate time="10" align="True" >
    <point x="1" y="0" z="1" />
    <point x="0.707" y="0.707" <="1”"/>
    <point x="0" y="1" z="1" />
    ...
    <point x="-1" y="0" z="1" />
</translate>
...
```

Figura 3.1: Exemplo do novo translate no XML

3.0.2 Rotação

Na fase anterior do programa, a funcionalidade de rotação permitia guardar informações utilizando quatro variáveis: angle, x, y e z. As três últimas variáveis eram usadas para especificar o eixo em que a rotação seria realizada.

Nesta nova fase, a variável *angle* foi removida e substituída pela variável *time*, que representa o tempo, em segundos, que demora a fazer uma rotação de 360º sobre o eixo especificado.

```
...
<rotate time="10" x="0" y="1" z="0" />
...
```

Figura 3.2: Exemplo do novo rotate no XML

3.0.3 Outras mudanças

Outra mudança, foi a utilização de VBOs e, para isso, o grupo criou uma variável global que armazena o número total de pontos e um vetor global com a quantidade de pontos que cada model tem.

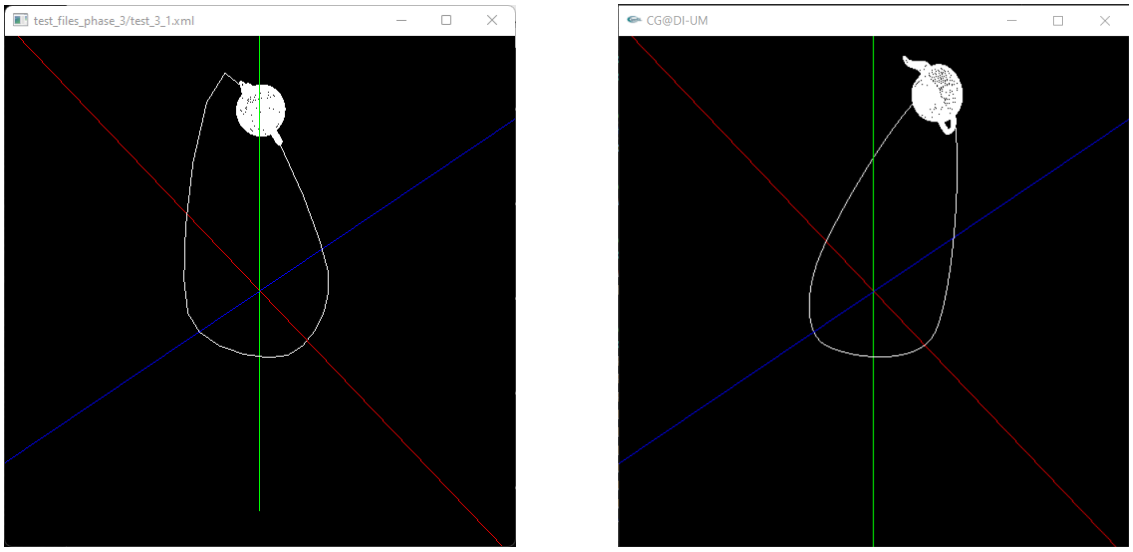
No processo de parse do ficheiro XML, o grupo utilizou, tal como nas fases anteriores, a biblioteca *RapidXML* para armazenar toda a informação em uma árvore. No entanto, diferentemente da fase anterior, os modelos são lidos antes de analisar as transformações. Isso permite ler os ficheiros dos modelos apenas uma vez e armazenar os seus pontos no vetor global, evitando a leitura repetitiva e custosa desses arquivos.

Dessa forma, conseguimos otimizar o processo de leitura do arquivo XML e melhorar a eficiência geral do programa.

4 Testes

Para testar a *Engine*, o grupo utilizou os ficheiros de teste fornecidos pelo professor e é possível afirmar que todos os modelos renderizados ficaram idênticos às imagens de teste. Isso significa que a *Engine* foi capaz de interpretar corretamente os dados dos ficheiros, realizar as transformações necessárias e desenhar os modelos de forma precisa e fiel às imagens fornecidas.

4.0.1 Teste 1



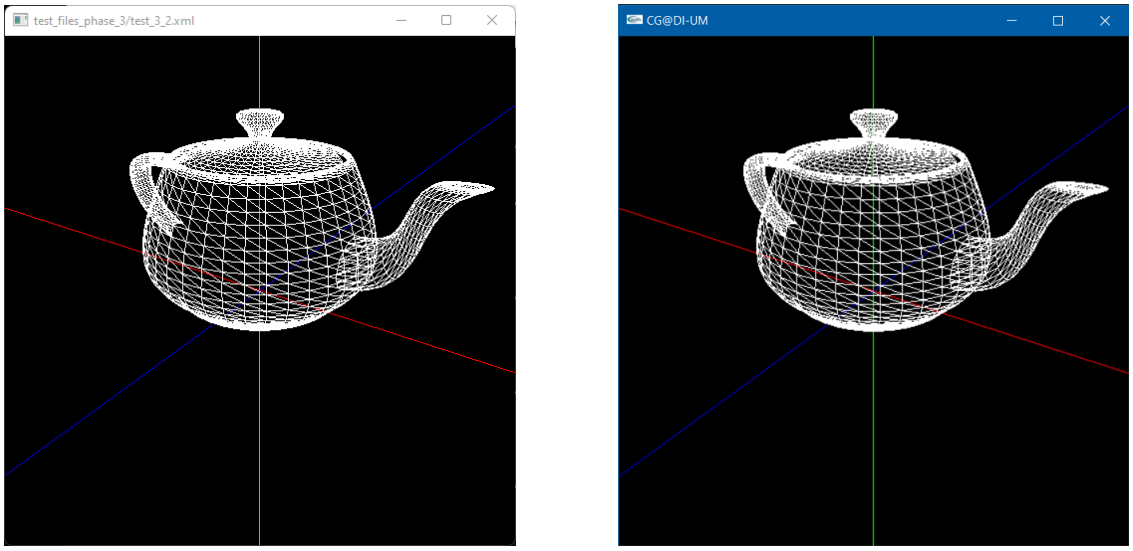
(a) Resultado esperado (b) Nosso resultado

Figura 4.1: Teste 1

O primeiro teste consistia em fazer o teapot seguir uma trajetória definida por uma curva cúbica *Catmull-Rom*. O teapot segue a trajetória e ajusta sua posição para seguir a direção das derivadas da curva. A trajetória em si é em torno do eixo do Y e demora 3 segundos para completar uma rotação completa de 360 graus.

Ao comparar o resultado obtido com o vídeo fornecido pelo professor, a principal diferença foi na quantidade de segmentos na trajetória. O grupo implementou a trajetória com 100 segmentos, enquanto que o resultado esperado parecia ter menos segmentos. É ainda de ressaltar que a comparação entre os resultados só foi possível através da visualização da trajetória em execução no nosso *Engine*, e não apenas por imagens comparativas.

4.0.2 Teste 2



(a) Resultado esperado

(b) Nosso resultado

Figura 4.2: Teste 2

4.0.3 Outros testes

Durante a implementação da nova funcionalidade de VBOs, foram encontrados alguns problemas que impediram que os testes das fases passadas executassem corretamente. Para garantir que tudo estivesse a funcionar corretamente, foram realizados os testes de todas as fases anteriores e corrigidos os erros encontrados. Após a correção desses erros, todos os testes tiveram os resultados esperados.

5 Sistema Solar

5.0.1 Criação do cometa

Para criar o modelo de cometa para o sistema solar desta fase, foi utilizado como base o arquivo *teapot.patch* fornecido pelo professor. Em seguida, foram removidas as características que se assemelhavam a um bule de chá (as pegas e o bico), e foram unidos os restantes pontos de modo a obter um modelo semi-esférico. O resultado pode ser visualizado na figura seguinte:

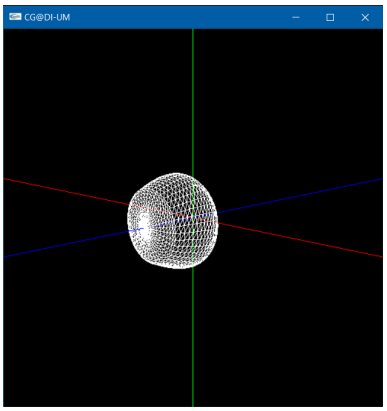


Figura 5.1: Teapot com as suas características retiradas

Para transformar esse modelo em um cometa, foi usado uma script em Python que aplicou deformações aleatórias no mesmo, gerando um formato mais semelhante a um cometa, que pode ser visto na figura seguinte:

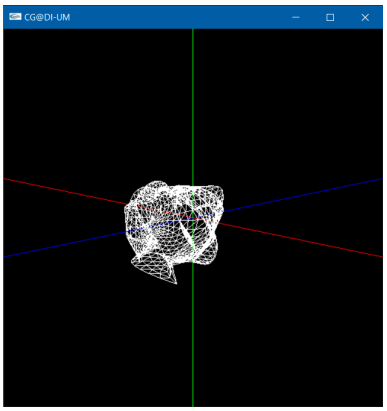


Figura 5.2: Cometa final

5.0.2 Alterações do sistema solar

Após a implementação das transformações na *Engine*, foram criadas as órbitas dos planetas utilizando a nova função de translação. Para isso, foi modificado o *script* de geração do sistema solar para criar pontos que formassem um círculo com curvas *Catmull-Rom*, utilizando um raio X. No entanto, o grupo optou por não utilizar essa implementação nas mais de 200 luas presentes no sistema solar, já que isso tornaria as trajetórias difíceis de serem visualizadas e afetaria o desempenho da demo. Além disso, também foram também implementadas as rotações em cada planeta para simular o ciclo diário.

Foram ainda considerados os tempos de rotação e translação de cada planeta, bem como as proporções entre eles. Por exemplo Vénus demora mais tempo a rodar sobre si mesmo do que demora a fazer uma rotação à volta do sol e o tempo que Neptuno demora a dar uma volta ao sol é maior que a Terra.

Por fim, o cometa foi adicionado com uma órbita que passa perto da câmera de modo a destacar as diferenças entre as esferas e o cometa O resultado final pode ser visualizado na figura abaixo.

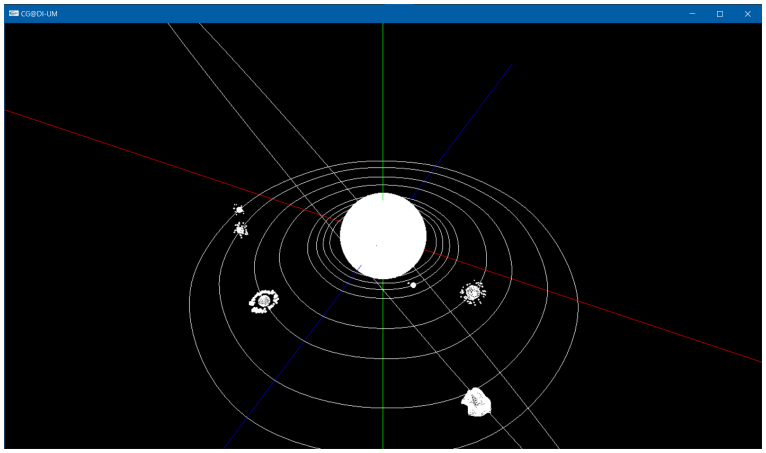


Figura 5.3: Sistema solar

Tal como na fase anterior, o ficheiro de configuração do sistema solar está localizado na pasta engine/config/config.xml, e os modelos estão armazenados na pasta engine/models. A *Engine* continua a utilizar o ficheiro config.xml da pasta config como ficheiro de configuração e qualquer modelo a ser utilizado deverá estar localizado na pasta models.

6 Conclusão

Concluímos que a terceira fase deste projeto foi um sucesso, uma vez que fomos capazes de cumprir todos os requisitos propostos no enunciado, melhorando o engine e o generator para ter capacidades de transformações geométricas, curvas, superfícies cúbicas, VBOs e melhorámos o modelo do sistema solar. Esses programas foram criados de forma extremamente satisfatória, demonstrando a nossa capacidade de aplicar os conhecimentos lecionados até o momento tanto nas aulas teóricas como nas práticas. Criámos novos modelos, como o teapot e o cometa e temos agora um sistema solar dinâmico graças aos translates e rotates desta fase.

Estamos confiantes que o sucesso desta terceira fase, nos ajudará na conclusão do projeto final.