Universidade do Minho
Escola de Engenharia
Licenciatura em Engenharia informática
Mestrado Integrado em Engenharia Informática

Unidade Curricular de Computação Gráfica

Ano Letivo de 2023/2024

Computação Gráfica – Checkpoint 3

João Pedro Mota Baptista a100705
João Pedro da Rocha Rodrigues a100896
João Manuel Machado Lopes a100594
Tiago Nuno Magalhães Teixeira a100665

Computação Gráfica

João Pedro Mota Baptista a100705
João Pedro da Rocha Rodrigues a100896
João Manuel Machado Lopes a100594
Tiago Nuno Magalhães Teixeira a100665

26 abril 2024

Resumo

Nesta fase do projeto foi-nos proposto atualizar tanto o Generator, como o Engine.

Para o generator, era necessário criar um novo modelo baseado em patches Bezier, recebendo como argumentos o nome do ficheiro que contém os pontos de controlo, e o nível de tesselação.

Para o Engine, tivemos de extender os elementos de Rotação e Translação. Na translação, serão dados os pontos para definir a curva Catmull-Rom, assim como o tempo em segundos para fazer a tal curva, e um booleano que define se o elemento está alinhado ou não com a curva. Em relação à rotação, em vez de ângulo podemos ter agora o tempo, que define a duração em segundos para rodar 360 graus no eixo especificado.

Área de Aplicação: Desenvolvimento de Software Gráfico
Palavras-Chave: Computação Gráfica

Contents

Engine

Em relação ao Engine, podemos constatar que manteve a mesma estrutura geral, sofrendo apenas ligeiras alterações e algumas adições.

O parser foi dividido em várias partes, com o intuito de melhorar a organização. Foram também adicionadas secções para lidar com as mudanças requisitada na “Rotação” e na “Translação” (Catmull-Rom). Para esta última, o parser lê os pontos do xml e armazena-os na struct “Translate” do “Group” em questão.

Foram também adicionadas as funções “getCatmullRomPoint” e “getGlobalCatmullRomPoint” para a funcionalidade da “Catmull-Rom curve”. Estas, tal como nas aulas, lidam com o processamento e cálculos dos pontos fornecidos para a execução da curva e da translação do objeto sobre a mesma.

Quanto ao desenho da curva em si, achamos adequado implementar a função “drawCatmullRomCurve” com um extra. O desenho da curva é escolha do utilizador. Este pode pressionar a tecla “space” para evidenciar ou não o desenho da curva.

É também essencial referir a existência da “alignObject”, que permite alinhar o objeto à curva Catmull-Rom caso a opção “align” do objeto no xml tenha o valor “true”.

Para além de todos estes elementos requisitados, nesta fase melhoramos também a rotação da câmara, tornando-a mais correta e “suave”.

Em relação aos VBOs, criamos uma struct Model com os atributos GLuint vboID, e um int numVertices. Foi criado também uma hashmap em que a chave é o nome do ficheiro, e o valor é uma struct Model. Esse hashmap permite, na função drawModel, verificar se o ficheiro já foi lido. Caso não tenha sido lido, esse mesmo ficheiro será lido e adicionado ao buffer, guardando no hashmap o seu ID e o número de vértices. Caso já esteja guardado na cache, apenas o é necessário desenhar.

Generator

No generator, foi adicionado uma nova condição ao argumento desse programa, o “patch”. Esse modo pede como argumentos: o ficheiro .patch, o nível de tesselação e o nome do ficheiro .3d resultante, contendo os vértices dos triângulos.

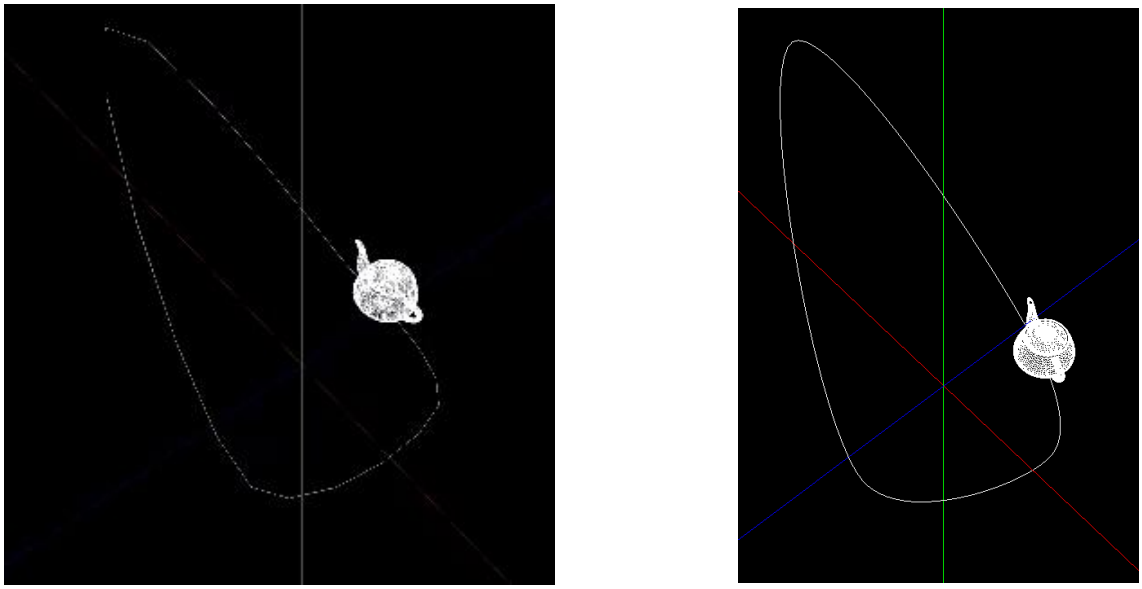
Inicialmente, é feito o parsing do ficheiro .patch e armazenado em estruturas de dados (patches e os pontos de controlo).

Seguidamente, são criados os pontos com o polinómio de Bernstein.

Finalmente, para cada patch, a função writeBezierPatch calcula pontos de acordo com o nível de tesselação, escrevendo os triângulos pela devida ordem no ficheiro .3d.

Testes

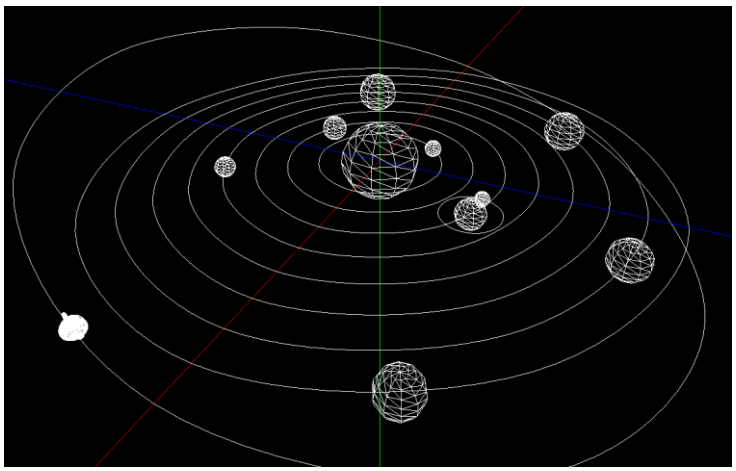
De forma a comprovar o correto funcionamento dos nossos programas, pusemos à prova os mesmos através da realização dos testes disponibilizados. Apresentamos de seguida os resultados do parsing de cada xml seguido da representação da figura respetiva.



Teste 1. Resultado Pretendido / Resultado Obtido

Assim, como podemos observar pelas figuras e ao comparar o nosso resultado com o esperado, concluímos que os outputs e o funcionamento tanto do Generator como do Engine estão corretos.

Em relação ao caso para demonstração, produzimos um esquema do sistema solar com o sol, os planetas, a lua e um “teapot” a orbitar à volta do sistema solar como representação de um cometa.



Conclusão

Acreditamos que atingimos os resultados pretendidos para esta fase, tendo o nosso programa cumprido os testes fornecidos.