Universidade do Minho

Departamento de Informática

Trabalho Prático

Ciências da Computação - LCC

Computação Gráfica

Grupo 25

Gonçalo Rodrigues A91641

Hugo Sousa A91654

Março de 2023

Conteúdo

Capítulo 1

Introdução

1.1 Contextualização

O presente relatório provém da segunda fase do projeto proposto no âmbito da Unidade Curricular de Computação Gráfica, dando continuidade à primeira fase, com o objetivo de criar um modelo estático do Sistema Solar a partir de um ficheiro XML previamente organizado e composto pelas transformações geométricas necessárias.

1.1.1 Estrutura do Relatório

Neste relatório será descrita a interpretação efetuada do enunciado do trabalho prático e a abordagem usada para resolver os problemas propostos.

Capítulo 2

Análise e Especificação

2.1 Descrição e Enunciado

2.1.1 Generator

Tendo em conta o enunciado e fase que nos encontramos, o *generator* não será modificado visto que não necessitamos de nenhuma figura diferente do que as que já foram construídas e geradas. Desta forma o *generator* desta fase é igual ao da primeira entrega.

2.1.2 Engine

No caso da *engine* são necessárias algumas alterações. A sua função continua a ser ler o ficheiro XML, sendo que numa primeira fase era apenas capaz de ler a configuração da janela, câmara e modelos (figuras geradas pelo *generator*). Para esta fase é necessário que seja capaz de reconhecer transformações como translações, rotações e escalas de modo que sejamos capazes de gerar o sistema solar a partir de transformações geométricas. Para além disso é necessário que exista uma estrutura de dados que permita a organização da informação das transformações que foram feitas, e que estabeleça uma hierarquia entre elas.

2.2 Requesitos

2.2.1 Engine

A *engine* deve então ser capaz de reconhecer alguns tipos de dados, tais como, configurações da window, camera e group. Este tipo, *group*, contém toda a informação referente à modelagem do Sistema Solar estático. Cada *group* pode ser estruturado por um *transform* (cada tipo transform deve conter apenas uma transformação de cada tipo), um *models* (tipo que contém as figuras a serem representadas) e por *subgrupos* (*group* dentro de outro *group*) que nos permitem estabelecer uma hierarquia entre transformações, ou seja, os subgrupos devem manter as transformações realizadas nos respectivos grupos pais.

Capítulo 3

Concepção e Desenho da Resolução

3.1 Organização do ficheiro config.xml

- **<world> e </world>**

  Reconhece a criação e o término de um mundo.

- **<window width= x height= y / >**

  Reconhece os inputs para o tamanho da janela.

- **<camera> e </camera>**

Reconhece as configurações iniciais para a position, lookAt, up e projection da câmera e o término deste tipo.

- **<group> e </group>**

  Reconhece a criação e o término de um grupo.

- **<models> e </models>**

  Reconhece a criação e término da secção onde devem ser desenhadas as figuras.

- **<model file="figura.3d">**

  Identifica o ficheiro com informação necessária para a "figura".

3.2 Transformações geométricas

- <**scale x=** ***x0*** **y=** ***y0*** **z=** ***z0*** **/ >**

  Reconhece uma escala de (x_0, y_0, z_0).

- <**translate x=** ***x1*** **y=** ***y1*** **z=** ***z1*** **/ >**

  Reconhece uma translação de (x_1, y_1, z_1).

- <**rotate angle=** ***a*** **x=** ***x2*** **y=** ***y2*** **z=** ***z2*** **/ >**

  Reconhece uma rotação de (a, x_2, y_2, z_2).

Capítulo 4

Codificação

4.1 Código e ferramentas utilizadas

Para a execução deste trabalho foi utilizada a linguagem C++ tanto na construção do *generator* como da *engine*. Para permitir a leitura da configuração do motor foi usada a biblioteca *tinyXML2*.

4.2 Alterações à engine

4.2.1 Leitura do ficheiro XML para uma árvore

Para esta fase foi alterado a estrutura do ficheiro com a configuração para poderem ser aplicadas transformações aos modelos, agora cada grupo contém dois valores, o **transform** onde são descritas as transformações e outro **models** onde são descritos os ficheiros a ser lidos.

De forma a ler estas transformações foi modificado o parser do ficheiro XML. A procura e a aplicação das definições foram mantidas, tanto da câmara como da janela, contudo agora são percorridos de forma recursivamente os grupos para serem guardadas as transformações e os modelos.

Foram criadas classes para guardar os grupos, as transformações e os modelos como objetos de forma a facilitar a construção da cena. Ainda foram criadas duas árvores, uma para guardar as definições de cada grupo e uma para guardar as informações dos ficheiros. Posteriormente estes serão guardados numa árvore com todos os grupos e semigrupos e num vetor de vbos.

4.2.2 Aplicação das Transformações

Para aplicar as transformações e desenhar a cena, é feita uma travessia **deph-first** pela árvore que contêm as informações dos grupos. Sempre que é encontrado um grupo é chamada a instrução **glPushMatrix**, no caso de ser encontrado uma translação, uma rotação ou uma escala são usadas as instruções **glTranslatef**, **glRotatef** e **glScalef** respetivamente. No caso de ser um modelo, o ficheiro é aberto e a sua informação é guardada num **vbo**. Por fim, de modo às transformações não passarem para grupos onde não é pretendido, é chamada a instrução **glPopMatrix** para restaurar a matriz.

Capítulo 5

Testes

5.1 Sistema Solar

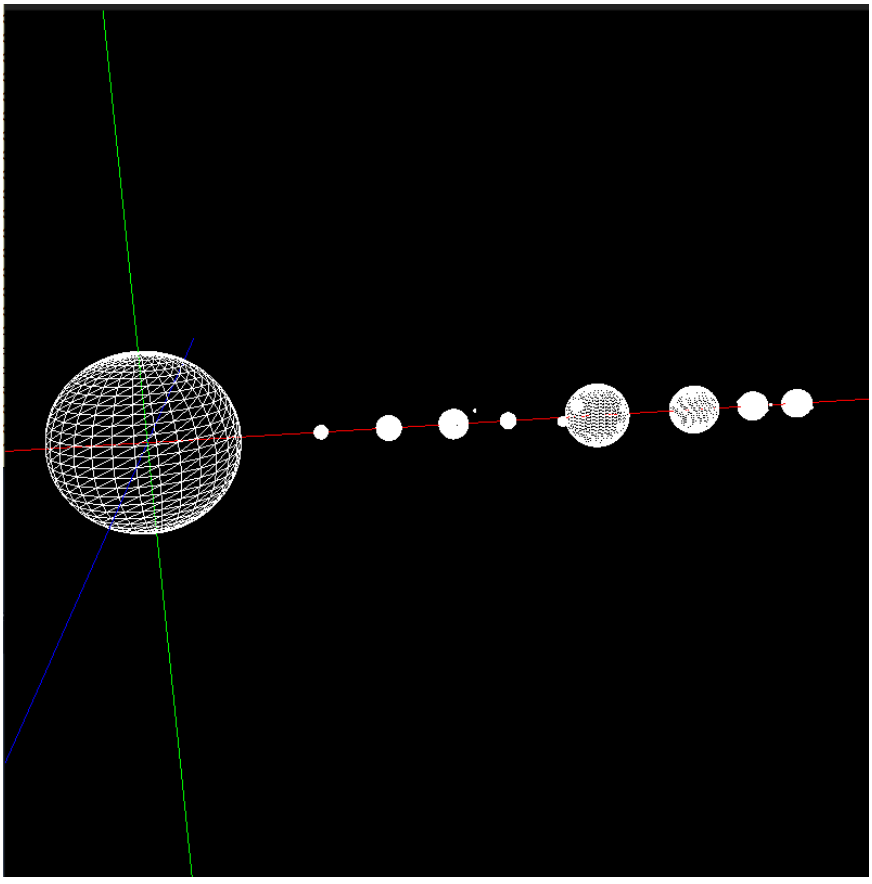

Figura 5.1: Representação do Sistema do Solar 1.

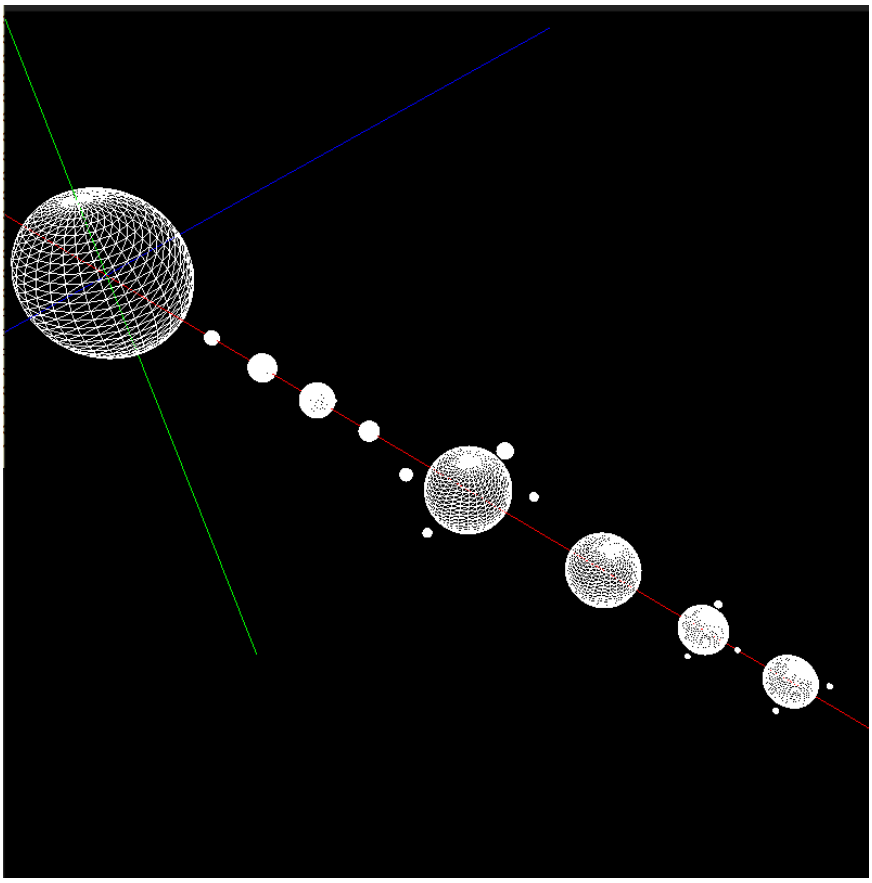

Figura 5.2: Representação do Sistema do Solar 1.

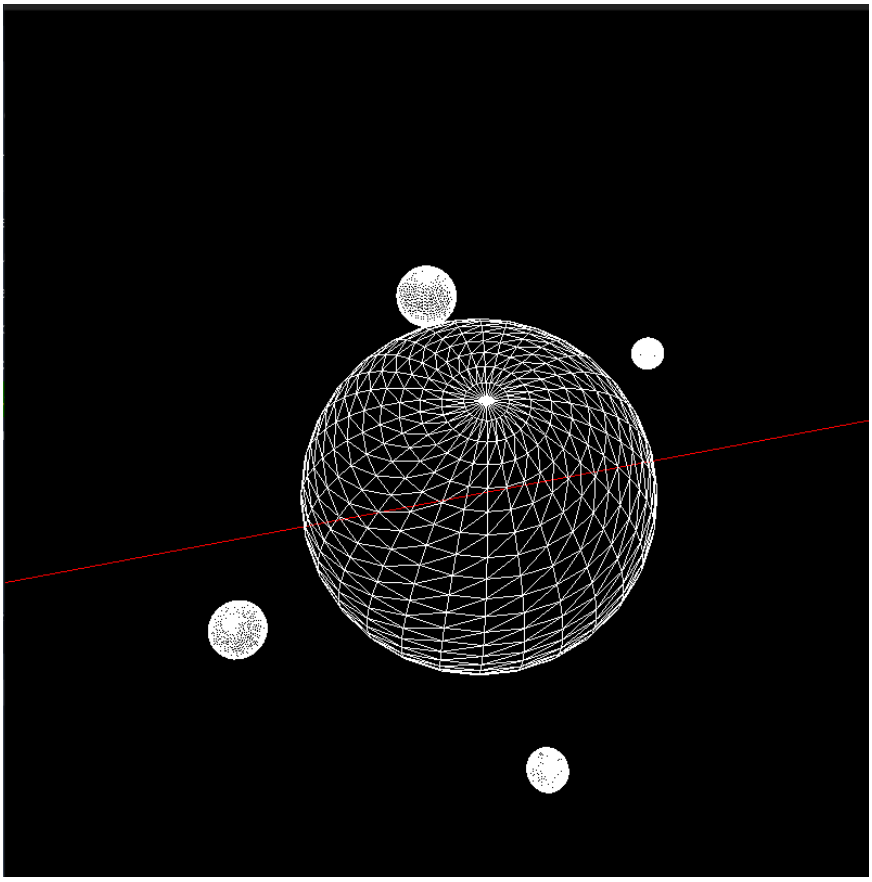

Figura 5.3: Representação de jupiter.

Capítulo 6

Conclusão

Nesta fase do trabalho foi proposto que o nosso programa fosse capaz de representar um sistema solar, para isso foi necessário implementar uma estrutura de dados capaz de organizar a informação de forma a que pudéssemos dar prioridades a certas transformaçs e hierarquizar as mesma. Desta maneira foi apenas preciso alterar a engine, implementando a estrutura de dados escolhida. Assim fomos capazes de realizar esta fase sem problemas alarmantes.