Universidade do Minho
Escola de Ciências

Computação Gráfica (3ª ano de LCC)

Trabalho Prático

2ª Fase

Relatório

Grupo 15

Pedro Manuel Pereira dos Santos	(A100110)
João Manuel Franqueira da Silva	(A91638)
David Alberto Agra	(A95726)
João Pedro da Silva Faria	(A100062)

4 de abril de 2024

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

No âmbito da unidade curricular de Computação Gráfica da Licenciatura em Ciências da Computação, foi proposto o desenvolvimento de duas aplicações utilizando a linguagem C++, recorrendo à ferramenta *OpenGl*. Esta segunda etapa do trabalho consiste na continuação em relação à primeira, na qual o objetivo é realizar uma cena hierárquica usando transformações geométricas a partir de um arquivo XML, onde apenas seria necessário atualizar a engine.
O modelo dos ficheiros XML a serem analisados é semelhante ao da fase 1, contudo, nesta segunda fase, teremos que ter em conta a existência de novos elementos referentes a transformações geométricas como translações, rotações e escalas, que deverão ser aplicadas antes de desenhar os objetos em questão, de forma correta e hierárquica.
Por último, também nos é solicitado um arquivo XML, com o objetivo, de quando este ser lido pela nossa aplicação, ser gerado uma imagem referente ao nosso sistema solar, com o sol, os planetas, e também algumas das suas luas. Tal será realizado com a introdução das transformações geométricas ao nosso meio.
Será fornecida uma análise detalhada das decisões e abordagens adotadas para viabilizar a implementação dessas aplicações propostas neste relatório.

Capítulo 2

Generator

Esta fase visa apenas alterar a aplicação engine, no entanto, para aumentar a complexidade do nosso Sistema Solar, decidimos também modificar a aplicação generator, de forma a introduzir uma nova primitiva gráfica que servirá para representar o anel do planeta Saturno.

2.1 Torus

É um toro, centrado no plano XZ, dividido em secções (slices) sendo estas divididas em subsecções (stacks), construido à base de duas circunferências, uma com raio maior, que representa a parte exterior da figura, e outra com um raio menor, referente à parte interior do objeto. Os ângulos **arch_alfa** e **arch_beta** representam o ângulo da stack e da slice que num dado momento nos encontramos a iterar. Para cada stack e para cada slice calculamos primeiro o vértice para a stack e slice atual, seguido do vértice referente à próxima stack na mesma slice, seguido depois do vértice nas seguintes stacks e slices, e, por último, o vértice para a atual stack e próxima slice. Todas estes pontos são obtidos através de funções trigonométricas.

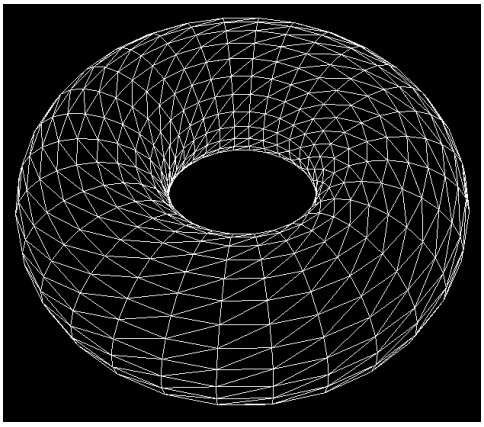

Figura 2.1: Torus: max_radius=6, min_radius=2, stacks=30, slices=25

Capítulo 3

Engine

Conforme mencionado anteriormente, a vasta maioria das modificações relativamente à fase 1 serão efetuadas na aplicação engine, sendo esta que lida com o parsing do ficheiro XML, e o desenho final, que incluí as possíveis transformações geométricas.

3.1 Parsing do XML e Armazenamento de Informação

Para extrair toda a informação necessária, optámos por percorrer cada arquivo XML como se tratasse de um grafo, para tal ser possível, atribuímos a cada elemento, um número identificador, referente à ordem pelo qual este surge no ficheiro.
Iniciada a travessia na função **get_transformacoes** para cada elemento, guardá-mos num vetor de pares **info** a informação relativa ao elemento bem como o seu identificador, e, de forma a obter todas as transformações geométricas a serem efetuadas antes de desenhar os vértices de um ficheiro, guardamos também num vetor **pais**, o identificador do seu pai, sendo este elemento o que chamou recursivamente a função de travessia para o elemento atual.
Adicionalmente, caso o elemento se trate de uma ficheiro, guardamos o seu valor e identificador num vetor adicional **ficheiros**, onde numa função separada **get_numbers**, serão obtidos os vértices a serem desenhados, e, caso se trate de uma transformação, num vetor **transf**. Tal facilitará a reorganização de toda a informação de seguida.
Depois de efetuada a travessia pelo arquivo, na função **organiza_vetores**, procuramos organizar toda a informação obtida, num formato mais pequeno e compacto. Começamos por percorrer o vetor dos ficheiros, e para cada elemento, através do seu identificador, percorrer todos os seus pais, e para cada pai percorrer todos os seus irmãos. Caso algum destes seja um elemento referente a uma transformação geométrica, guardamos noutro vetor de pares **final**, a transformação em causa, o valor das variáveis referentes à transformação, e também o identificador do ficheiro em questão.
Desta forma, para aplicar as transformações geométricas corretas antes de desenhar os pontos obtidos de um ficheiro, não será necessário percorrer os vetores referentes aos pais e as informações, tal seria custoso. Precisamos apenas de percorrer o nosso vetor final de uma forma sequencial, o que torna o nosso programa bastante mais eficiente.

3.2 Desenho

Após completado o processo mencionado anteriormente, para desenhar precisamos apenas de percorrer os vetores com os pontos retirados dos ficheiros .3d, primeiramente, empilhando na stack a matriz identidade, e, ao mesmo tempo, percorrer o vetor final das transformações, se o número associado à transformação corresponder ao número do vetor com os pontos, então, antes de desenhar os pontos, a transformação geométrica em causa é efetuada, sendo no fim apenas necessário, efetuar um pop na stack das matrizes, para recuperar a matriz identidade, garantindo que o próximo vetor de pontos, não tenha nenhuma aplicação geométrica previamente efetuada.
Assim asseguramos que, para cada ficheiro .3d, as transformações de todos os seus pais são aplicadas sequencialmente, de forma a desenhar corretamente o objeto.

Capítulo 4

Sistema Solar

Com o propósito de ilustrar as funcionalidades desenvolvidas nesta segunda fase do projeto, foi solicitada uma demonstração do Sistema Solar. Para isso, foi elaborado um arquivo XML que detalha todas as transformações e modelos necessários para representar todos os corpos celestes do Sistema Solar, o Sol, os planetas, desde Mercúrio até Neptuno, assim como a Lua em relação à Terra, e também o anel de Saturno.

4.1 Modelos Utilizados

As únicas primitivas a serem usadas para desenhar o Sistemas Solar serão a esfera, no caso dos planetas, luas e sol, e o torus, para o anel de Saturno.

4.2 XML Utilizado

O modelo para o nosso ficheiro XML é bastante simples, como a posição de cada planeta é apenas dependente do Sol e como todos os planetas se encontram relativamente no mesmo plano, podemos primeiro desenhar o Sol no centro, com uma escala, e para todos os planetas, aplicar uma rotação em torno do eixo y (para espalhar os planetas pela tela), seguida de uma translação (que respeite a distância real do próprio ao Sol) e uma escala (que respeite o seu tamanho relativo para com todos os outros corpos celestes), em relação ao posicionamento do Sol, não herdando assim nenhuma transformação prévia. Deparamos-nos com uma situação de hierarquia de transformações no caso das luas, em que iremos herdar as transformações realizadas para o desenho do respetivo planeta. Optamos então por realizar outra translação, desta vez em torno de vários eixos (para que a lua em questão seja desenhada num plano diferente do dos planetas) e outra escala (tendo em conta a escala herdada). No caso do anel de Saturno, apenas efetuamos uma rotação, para inclinar minimamente o próprio, de resto, basta apenas garantir que após a escala realizada para desenhar Saturno, o raio da menor circunferência pertencente ao anel seja superior ao raio do planeta.

```
<group> <!-- SATURNO -->
    <transform>
        <rotate angle="295" x="0" y="1" z="0" />
        <translate x="150" y="0" z="150" />
        <scale x="0.88" y="0.88" z="0.88" />
    </transform>
    <models>
        <model file="sphere_10_20_20.3d" /> <!-- ./generator sphere 10 20 20 sphere_10_20_20.3d -->
    </models>
    <group> <!-- SATURNO ANEL -->
        <transform>
            <rotate angle="-15" x="1" y="0" z="1" />
        </transform>
            <models>
                <model file="torus_15_14_20_20.3d" /> <!-- ./generator torus 15 14 20 20 torus_15_14_20_20.3d -->
            </models>
    </group>
    <group> <!-- SATURNO LUA -->
        <transform>
            <translate x="-13" y="13" z="-13" />
            <scale x="0.1" y="0.1" z="0.1" />
        </transform>
            <models>
                <model file="sphere_10_20_20.3d" /> <!-- ./generator sphere 10 20 20 sphere_10_20_20.3d -->
            </models>
    </group>
</group>
<group> <!-- URANO -->
    <transform>
        <rotate angle="148" x="0" y="1" z="0" />
        <translate x="200" y="0" z="200" />
        <scale x="0.75" y="0.75" z="0.75" />
    </transform>
    <models>
        <model file="sphere_10_20_20.3d" /> <!-- ./generator sphere 10 20 20 sphere_10_20_20.3d -->
    </models>
    <group> <!-- URANO LUA -->
        <transform>
            <translate x="15" y="15" z="0" />
            <scale x="0.11" y="0.11" z="0.11" />
        </transform>
            <models>
                <model file="sphere_10_20_20.3d" /> <!-- ./generator sphere 10 20 20 sphere_10_20_20.3d -->
            </models>
    </group>
</group>
<group> <!-- NEPTUNO -->
    <transform>
        <rotate angle="100" x="0" y="1" z="0" />
        <translate x="250" y="0" z="250" />
        <scale x="0.79" y="0.79" z="0.79" />
    </transform>
```

Figura 4.1: Parte do ficheiro XML para o Sistema Solar

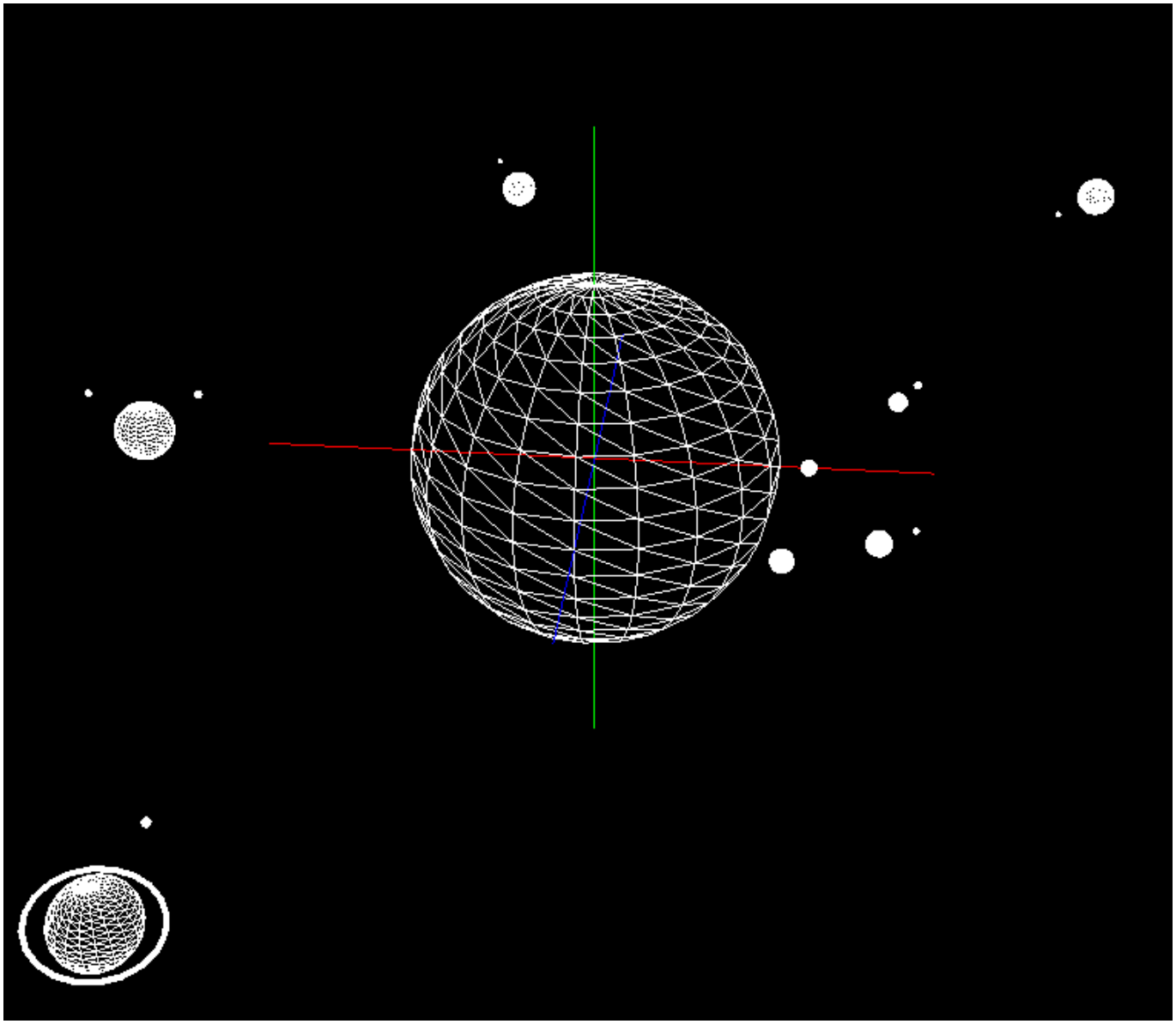

Figura 4.2: Desenho do Sistema Solar

Capítulo 5

Conclusão

Concluindo então esta segunda fase do projeto, destaca-mos a implementação bem-sucedida das funcionalidades propostas, com ênfase na capacidade do sistema de criar representações hierárquicas e realizar as devidas transformações geométricas, de forma correta, a partir do arquivo XML. A demonstração do Sistema Solar ilustra efetivamente a aplicação dessas funcionalidades, evidenciando a precisão e a flexibilidade do sistema em lidar com diferentes elementos e as suas relações espaciais.
Além disso, ao longo do desenvolvimento, foram encontrados desafios técnicos, que requeriam soluções criativas e eficientes, principalmente, na forma em como se iriam guardar as informações, de modo a mais tarde conseguir obter o relacionamento hierárquico das transformações para cada ficheiro, demonstrando assim, a capacidade do grupo em superar obstáculos e encontrar soluções adequadas para alcançar os devidos objetivos.
No entanto, há espaço para melhorias contínuas, como a otimização do processo de obtenção e reorganização das informações presentes nos arquivos XML, que resultaria numa otimização do nosso programa.
Resumindo, o grupo atribuí um balanço positivo ao trabalho realizado nesta fase, tendo cumprido todos os requisitos pedidos, e tendo em conta todas as bases adquiridas até agora, sentimos-nos também, bem preparados para a realização das futuras etapas deste projeto.