Universidade do Minho

Mestrado Integrado em Engenharia Informática

Computação Gráfica

Trabalho Prático - Fase II

Grupo 21

Ana Pereira
A81712

Francisco Freitas
A81580

Maria Dias
A81611

Pedro Freitas
A80975

March 25, 2019

Índice

1 Introdução

Nesta segunda fase do projeto o objetivo é, através do engine implementado na fase anterior, ler e processar informação de novos ficheiros XML que além de receber as figuras recebe também transformações geométricas.

Assim a estrutura dos ficheiros XML sofreram alterações de forma a conseguirem garantir as rotações, translações e também a mudança de escala e cor dos modelos.

Para satisfazer estes novos requisitos foram implementadas novas classes e alteradas algumas outras, permitindo-nos assim criar um primeiro modelo estático do sistema solar.

2 Arquitetura de Código

2.1 Novas classes

Além das classes definidas na fase anterior, *Point* e *Shape*, foi necessário criar mais duas classes de modo a armazenar a informação extraída dos ficheiros XML, estas denominadas de *Action* e *Group*.

2.1.1 Action.cpp

Esta classe representa uma ação (translate, rotate, scale ou colour), sendo identificada pela respetiva tag que indica o tipo de transformação, contendo os três valores x, y e z designados no ficheiro e ainda o modo de desenho(linha ou ponto). De notar, que foi necessário implementar a subclasse *Rotate* que acrescenta o ângulo da rotação.

2.1.2 Group.cpp

Classe que guarda num vetor todas as transformações presentes num determinado grupo do ficheiro XML e noutro vetor o conjunto das figuras necessárias para o desenho desse mesmo grupo.

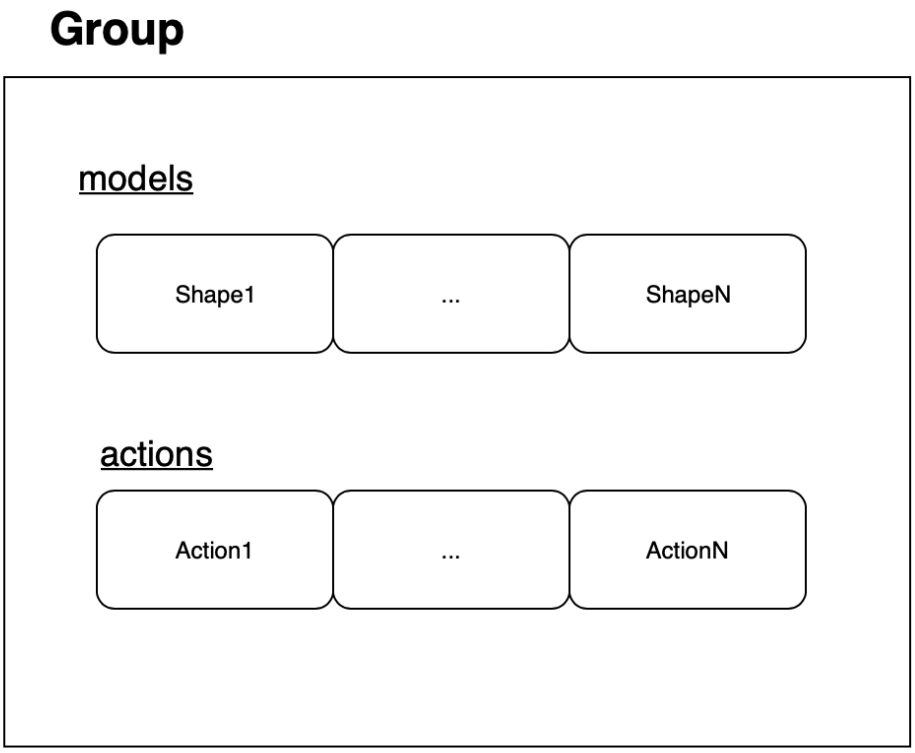

Figura 1: Estrutura da classe *group*

2.2 Classes inalteradas

Nesta segunda fase foi necessário a criação das classes referidas anteriormente, porém as classes já definidas na primeira fase continuam igualmente indispensáveis.

2.2.1 Point.cpp

Classe que guarda as coordenadas x, y e z de um ponto, necessário para o desenho de cada vértice de um triângulo, uma vez que a base de todas as figuras desenhadas é o triângulo.

2.2.2 Shape.cpp

Classe que armazena num vector o conjunto de todos os pontos pertences a um determinado modelo.

2.3 Engine

De modo a aplicar as transformações geométricas, criou se a variável global **scene** que consiste num vetor que armazena a informação dos grupos encontrados no ficheiro.

A função **renderGroup** é aplicada para percorrer a lista de ações de um grupo, efetuando as respetivas tranformações. São ainda desenhadas as figuras guardadas nesse grupo. Sendo que a nossa **scene** consiste num vetor de **groups**, a função que acabamos de mencionar é iterada pelo mesmo.

3 Parsing dos Ficheiros XML

O ficheiro XML fornecido passa por um processo de leitura e interpretação, que começa por ler, um a um, os diferentes elementos presentes no mesmo, e definindo um comportamento para cada uma. Assim, são lidas as formas e ações presentes no ficheiro. Entenda-se por ação qualquer um dos seguintes: *translate*, *rotate*, *scale* e *colour*.

A função principal que trata da leitura dos ficheiros XML designa-se **parseGroup**.

Na primeira chamada da função **parseGroup**, é lhe dado como input o primeiro filho do elemento **scene** no ficheiro XML, visto que a informação que pretendemos aceder está dentro dos elementos que são filhos de *scene*.

Sempre que o elemento lido se tratar de uma ação, esta é armazenada na lista de ações do grupo atual. Caso seja um modelo, este aplica a função **readFile** que lê e insere os pontos das figuras presentes no ficheiro indicado numa instância da estrutura *Shape*. Posteriormente, esta será adicionada ao vetor de formas do grupo.

Dado que o ficheiro está organizado hierarquicamente, os grupos contidos dentro de outros grupos, isto é, os grupos filho, herdam as transformações do grupo pai. Assim, ao encontrar um grupo filho, é efetuado o clone do grupo corrente e aplicado recursivamente a função **parseGroup**.

3.1 Exemplos de novos ficheiros XML

Como foi referido anteriormente, para esta fase os ficheiros XML sofreram algumas alterações de forma a suportar os novos requisitos. Podemos ver um claro exemplo disso nos seguintes excertos:

```
<group>
    <tag type="L" />
    <colour R='255' G='197' B='0' />
    <scale X='4' Y='4' Z='4' />
    <models>
        <model file='../files3d/sun.3d' /> '<!--'#SOL '-->'
    </models>
</group>
```

Figura 2: Excerto de um ficheiro xml - Sol

Como podemos ver, a nova estrutura *group* é responsável pela criação da figura representante do Sol. Observando a figura 2 vemos duas das novas funcionalidades: *colour* e *scale* aplicadas a um modelo correspondente a uma esfera. Utilizamos a tag *colour* para dar uma cor personalizada aos modelos e a tag *scale* para redimensionar os modelos presentes nos ficheiros fornecidos na escala fornecida pelas coordenadas.

```
<group>
    <tag type="L" />
    <rotate angle='270' X='0' Y='1' Z='0' />
    <rotate angle='-25' X='1' Y='0' Z='0' />
    <translate X='23' Y='0' Z='0' />
    <colour R='230' G='212' B='168' />
    <scale X='0.670' Y='0.670' Z='0.670' />
    <models>
        <model file='../files3d/sphere.3d' /> '<!--' #SATURNO '-->'
    </models>
    <group>
        <translate X='0.4' Y='0.7' Z='2.5' />
        <colour R='255' G='255' B='255' />
        <scale X='0.15' Y='0.15' Z='0.15' />
        <models>
            <model file='../files3d/sphere.3d' /> '<!--' #Saturno LUA Titan'-->'
        </models>
    </group>
    <group>
        <colour R='230' G='212' B='168' />
        <scale X='0.6' Y='0.6' Z='0.6' />
        <models>
            <model file='../files3d/ring.3d' /> '<!--' #SATURNO ANEL '-->'
        </models>
    </group>
</group>
```

Figura 3: Excerto de um ficheiro xml - Saturno

Já neste pedaço do ficheiro xml, podemos ver que temos dois *groups* filhos dentro de um *group* pai. No *group* pai são feitos primeiro dois *rotate* para direcionar o planeta, sendo que o primeiro serve para posicionar o planeta relativamente ao sol, e o segundo para ilustrar a ligeira inclinação que saturno apresenta em relação aos outros planetas. Depois disso, é feito um *translate* para centrar o planeta na posição devida. Além disso, podemos ver também um *colour* e um *scale*. Estas ações são relativamente a uma esfera. No primeiro *group* filho estamos a criar um satélite natural do planeta(a Lua Titan) usando para isso uma translação que é relativa ao planeta Saturno. No segundo *group* filho já estamos a criar o anel do planeta. Assim, é necessário aplicar uma cor e uma escala diferentes da lua e do próprio planeta. Sendo este um anel também é necessário adaptar a figura em que vamos aplicar a transformação, tratando-se neste caso de um anel.

4 Sistema Solar

Depois de implementadas as funcionalidades foi hora de pormos em prática de forma a ir de encontro com o pedido: uma maqueta do sistema solar. Num ficheiro XML, foram descritas as figuras geométricas e as respetivas ações necessárias para obter um sistema solar.

Como resultado de todo o processo, ficamos com os seguintes modelos:

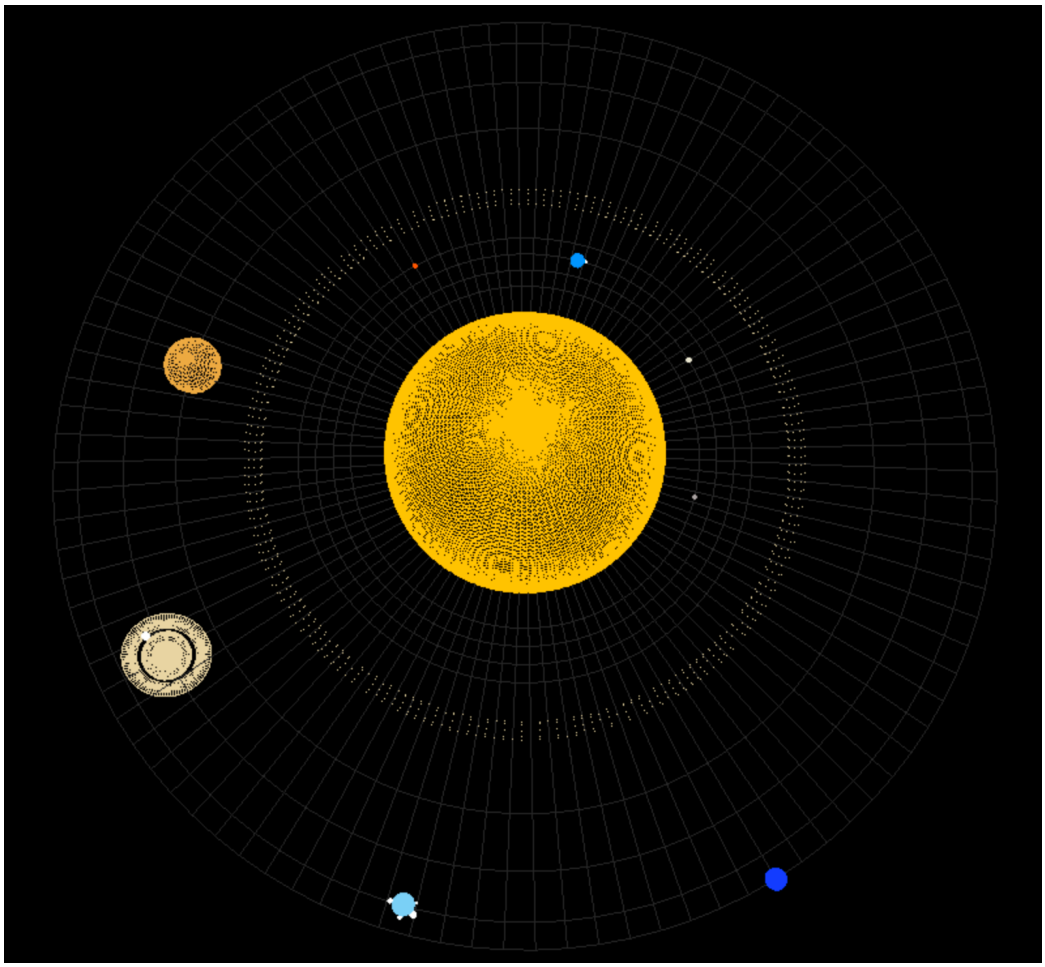

Figura 4: Sistema Solar visto de cima (1)

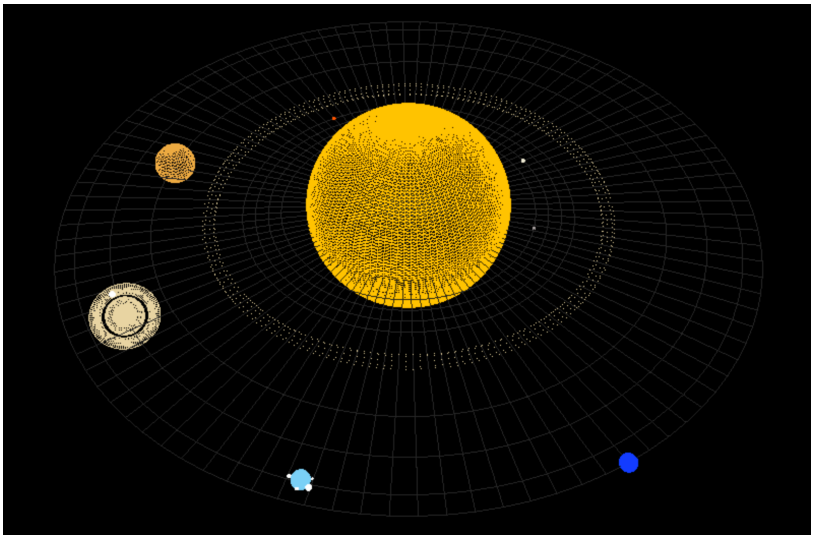

Figura 5: Sistema Solar visto de outra perspetiva (2)

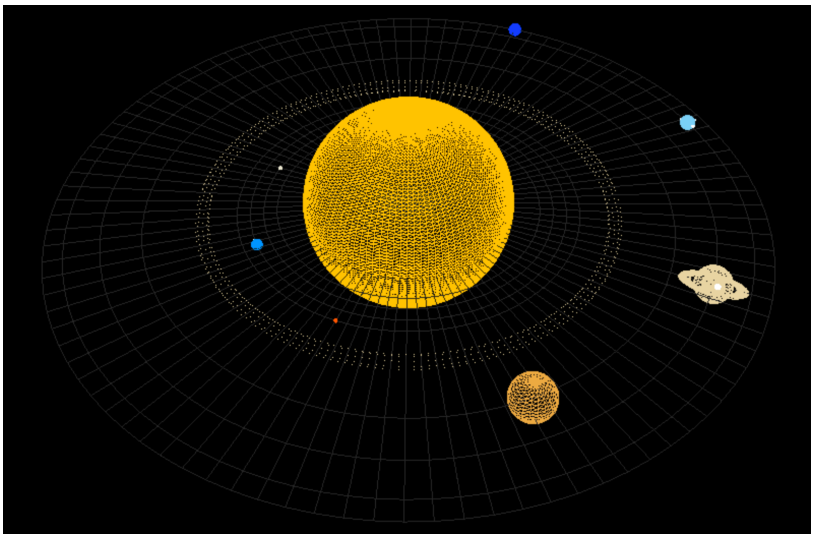

Figura 6: Sistema Solar visto da outra perspetiva (3)

5 Conclusão

Concluindo assim esta segunda fase do projeto podemos ver e perceber o quão útil são as transformações e rotações para trabalhos um pouco mais complexos. Assim, com as figuras geométricas previamente implementadas podemos montar todo um cenário com maior grau de dificuldade, porém visualmente mais agradável e aprazível.

Depois de obtermos o resultado final, achamos que temos um trabalho bem conseguido com um bom grau de fidelidade ao esperado.