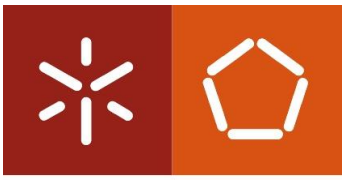

Universidade do Minho

Departamento de Informática

Computação Gráfica
Transformações Geométricas

Bárbara Cardoso (a80453) Márcio Sousa (a82400)
Pedro Mendes (a79003)

12 de março de 2022

Conteúdo

1 Introdução

Este trabalho foi proposto no âmbito da unidade curricular de Computação Gráfica, e tem como objetivo desenvolver um motor gráfico genérico para representar objetos a 3 dimensões.

Nesta segunda fase do projeto estendemos as capacidades do motor gráfico anteriormente definido com a adição de transformações geométricas, para aplicar as antes definidas primitivas.

Para alem disto, também foi adicionada a definição de um torus ao gerador, especificamnente para suportar o anel de Saturno.

Com o intuito de demonstrar as capacidades do motor gráfico desenvolvido até ao presente, procedemos à criação de um modelo do Sistema Solar.

2 Estrutura dos ficheiros usados pelo programa

Os ficheiros utilizados pelo programa estão definidos em formato `xml` com as seguintes tags:

- `<scene>` Define o início de uma cena. Este funciona como se fosse um super grupo e não tem nenhum atributo.
- `<group>` Define um grupo de transformações, modelos e subgrupos, e aceita como atributos as cores com que os modelos devem ser pintados. Por exemplo, `<group R='1'> ...</group>` pinta todos os modelos de vermelho. Para além disto, os subgrupos deste irão herdar também estas cores a não ser que definam as suas próprias cores. Os atributos que podem ser usados são R, G, B e A, que correspondem, respetivamente, a *Red*, *Green*, *Blue* e *Alpha*. Alternativamente, pode também ser passado o parâmetro *RAND* para que as cores sejam escolhidas aleatoriamente.
- `<models>` e `<model>` Definem os modelos a desenhar num grupo. O `<models>` contém uma lista do `<model>` e este tem o atributo `file` que indica qual o ficheiro `.3d` que vai ser usado.
- `<translate>`, `<rotate>` e `<scale>` Definem as transformações geométricas possíveis que podem ser usadas para transformar os modelos e os subgrupos. Os atributos de cada uma destas transformações são:
  - **Translate:** X, Y, Z.
  - **Rotate:** angle, X, Y, Z
  - **Scale:** X, Y, Z

Exemplo de uma ficheiro de input:

```
<scene>
    <group R="1">
        <translate X="1" />
        <models>
            <model file="sphere.3d" /> <!-- Este modelo será vermelho -->
        </models>
        <group>
            <translate axisY="1" />
            <models>
                <model file="cone.3d" /> <!-- Este modelo será vermelho -->
            </models>
        </group>
        <group G="1">
            <models>
                <model file="torus.3d" /> <!-- Este modelo será verde -->
            </models>
        </group>
    </group>
    <group RAND="true">
        <translate X="5" axisY="0" axisZ="2" />
        <rotate angle="45" axisX="0" axisY="1" axisZ="0" />
        <models>
            <model file="box.3d" /> <!-- Este modelo tera cores aleatorias -->
        </models>
    </group>
</scene>
```

3 Arquitetura do Projecto

3.1 Câmera

A câmera não sofreu grandes alterações nesta segunda fase, em relação à fase anterior. Esta, apenas, foi movida para um *namespace*, à parte.

3.2 Model

O model também não sofreu grandes alterações. Agora, para além de poder ser desenhado com cores aleatórias pode também ser desenhado sem cores ("herdando" a última cor que tenha sido seleccionada).

3.3 Render

O render foi criado para mover a lógica de desenho para um *namespace*, à parte, e alterado para acomodar o resto das modificações.

3.4 Group

O `Group` é a nossa classe principal desta fase do projeto. Esta foi desenhada para imitar o formato do xml, possuindo então uma sequência de transformações, um conjunto de modelos e por fim os seus subgrupos, ficando assim, com uma definição recursiva. Este é construído diretamente a partir do xml, e disponibiliza uma função de desenho que pede a profundidade da árvore de grupos e subgrupos que se pretende desenhar. Esta função funciona da seguinte forma:

Caso o número passado como argumento seja maior do que 0, então esta começa por realizar todas suas transformações, e de seguida desenha todos os modelos. Este passo tem a atenção de aplicar as cores que possui, por passagem direta, ou por herança, ou então desenha o modelo com cores aleatórias (ver 2). Por fim, chama recursivamente o método de desenho sobre os subgrupos passando como argumento a profundidade recebida menos um.

Esta funcionalidade permite que sejam desenhadas apenas partes de árvore durante a execução usando as teclas `[` e `]` para diminuir ou aumentar a profundidade da "árvore"de grupos que são desenhados, respetivamente. Por defeito, são desenhados todos os níveis.

3.5 Transformations

As transformações foram definidas como uma interface que obriga a implementação do método `transform`. Foram definidas também 3 implementações da mesma, `Rotate`, `Translate` e `Scale`, correspondentes às 3 transformações possíveis.

4 Novas Primitivas

4.1 Torus

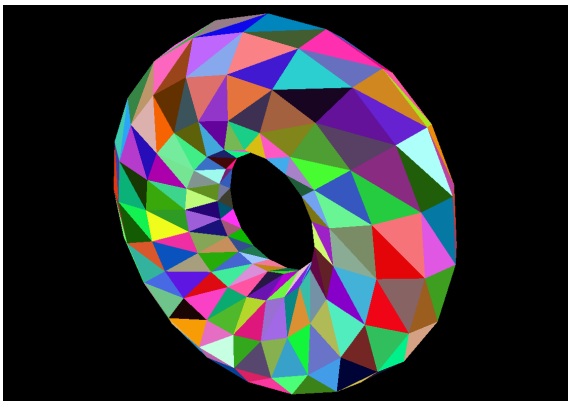

Figura 1: Torus: *innerRadius*: 1, *outerRadius*: 3, *sides*: 10, *rings*: 20

Um torus é definido por duas circunferências, uma interior e outra exterior, em que a circunferência exterior tem um raio de *outerRadius* e a circunferência interior um raio de $outerRadius - 2 \times innerRadius$. O *innerRadius* é o raio de cada um dos anéis, que estão situado entre as duas circunferências. Os centros destes anéis estão distanciados segundo um ângulo de α_{step}, usando coordenadas polares.

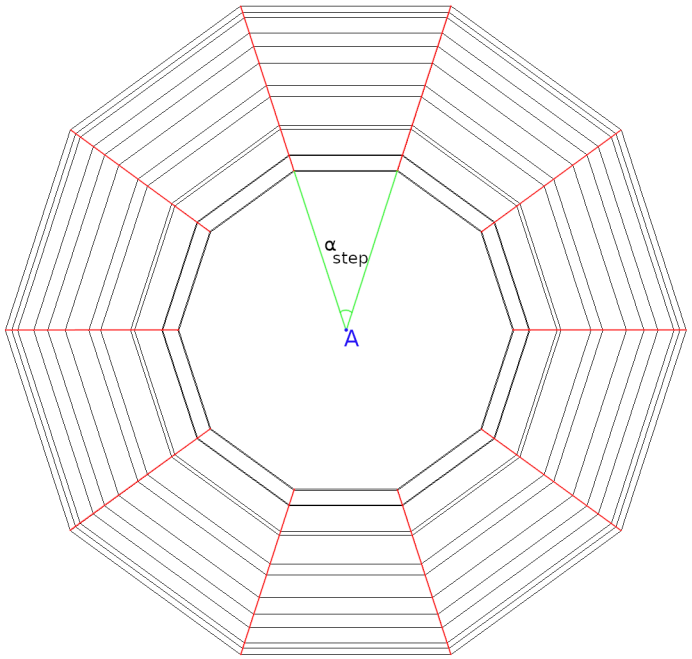

Figura 2: Torus completo

Para cada par de anéis, são desenhados uma série de retângulos, que os interligam ambos, para fazer a "parede" do torus. Estes retângulos têm como vértices os pontos p_0, p_1, p_2 e p_3 que são calculados por coordenadas polares, e têm como centro de aplicação o centro do anel a que pertencem (ponto B). O espaçamento destes pontos segue, por sua vez, um ângulo de β_{step}.

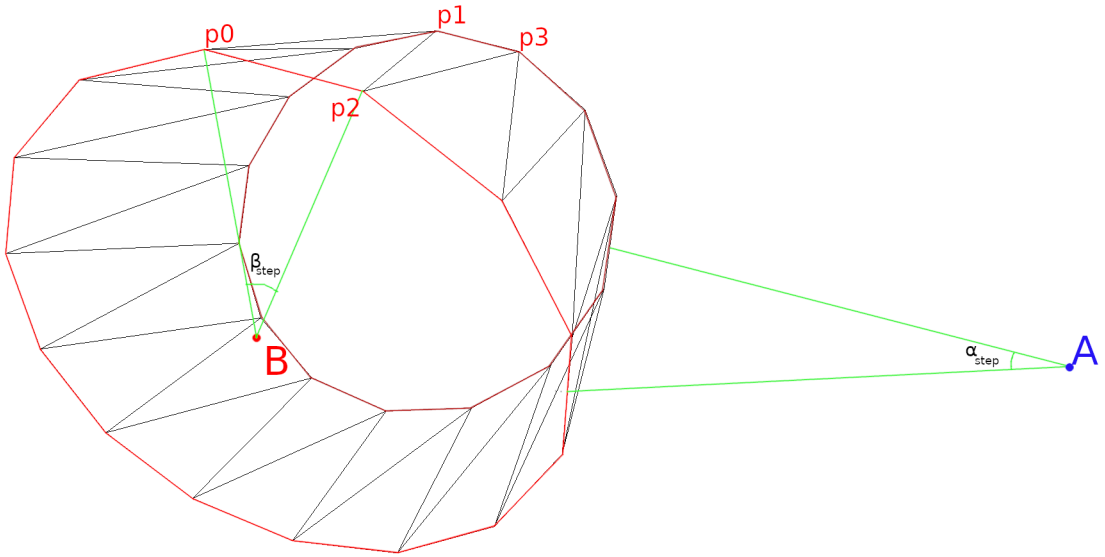

Figura 3: Corte do torus com um par de anéis

$$B = ((outerRadius - innerRadius) \times \sin\alpha, \quad (outerRadius - innerRadius) \times \cos\alpha, \quad 0)$$

$$B' = ((outerRadius - innerRadius) \times \sin(\alpha + \alpha_{step}), \quad (outerRadius - innerRadius) \times \cos(\alpha + \alpha_{step}), \quad 0)$$

$$p_0 = \begin{pmatrix} B_x + (innerRadius \times \cos\beta \times \sin\alpha) \\ B_y + (innerRadius \times \cos\beta \times \cos\alpha) \\ B_z + (innerRadius \times \sin\beta) \end{pmatrix}$$

$$p_1 = \begin{pmatrix} B'_x + (innerRadius \times \cos\beta \times \sin(\alpha + \alpha_{step})) \\ B'_y + (innerRadius \times \cos\beta \times \cos(\alpha + \alpha_{step})) \\ B'_z + (innerRadius \times \sin\beta) \end{pmatrix}$$

$$p_2 = \begin{pmatrix} B_x + (innerRadius \times \cos(\beta + \beta_{step}) \times \sin\alpha) \\ B_y + (innerRadius \times \cos(\beta + \beta_{step}) \times \cos\alpha) \\ B_z + (innerRadius \times \sin(\beta + \beta_{step})) \end{pmatrix}$$

$$p_3 = \begin{pmatrix} B'_x + (innerRadius \times \cos(\beta + \beta_{step}) \times \sin(\alpha + \alpha_{step})) \\ B'_y + (innerRadius \times \cos(\beta + \beta_{step}) \times \cos(\alpha + \alpha_{step})) \\ B'_z + (innerRadius \times \sin(\beta + \beta_{step})) \end{pmatrix}$$

Para cada retângulo desenhado, o β avança β_{step} e para cada anel terminado α avança α_{step}.

5 Sistema Solar

O sistema solar é constituído pelo Sol, que está no centro, 8 planetas e as respetivas luas.

Cada um destes astros é representado por uma esfera de raio 1 que depois é escalada para o tamanho pretendido. O sol é o astro que está situado no centro do sistema solar e por isto todas as transformações dependem deste.

Os planetas pertencem ao grupo do sol e que por sua vez têm o seu próprio subgrupo que contem as suas luas. Logo, os planetas herdam todas as caraterísticas do sol e as luas herdam as caraterísticas do seu respetivo planeta. O planeta Saturno tem um anel que foi representado por um torus com a escala Z = 0.

Para colocar os planetas nas suas coordenadas foi escrito um curto *script* em *python* de forma a situar os planetas em torno do sol. Este usa raios predefinidos para cada planeta, que são as distâncias destes ao sol, e coordenadas polares para escolher uma posição aleatória. Para as luas de cada planeta decidimos utilizar a mesma estratégia mas com a pequena diferença de que estas têm também uma componente vertical, ou seja, as suas posições podem ficar acima do equador.

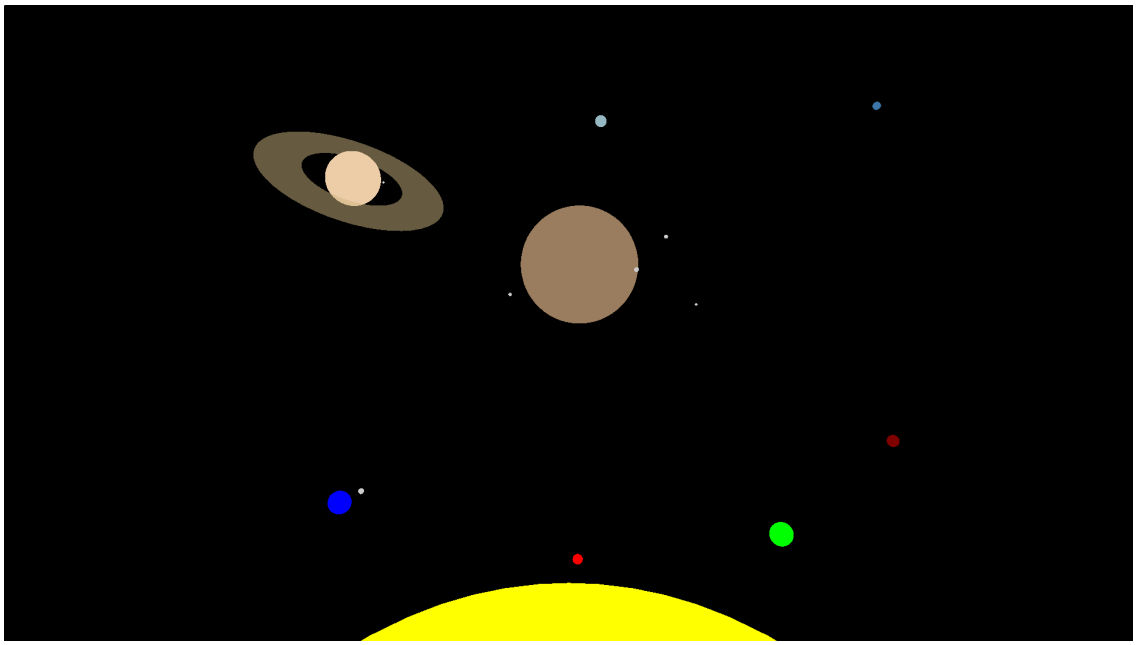

Figura 4: Um dos sistemas solares gerados

6 Conclusões e Trabalho Futuro

Esta fase do projeto possibilitou uma melhor compreensão do funcionamento das transformações geométricas do OpenGL e como estas podem ser usadas, sozinhas ou combinadas entre elas, de forma a criar cenários compostos por várias primitivas gráficas previamente definidas.

Fazendo uma análise geral ao trabalho desenvolvido ao longo desta fase do projeto, podemos concluir que foram cumpridos todos os objetivos que nos foram propostos.

Como trabalho futuro, serão animados todos os astros do modelo do sistema solar representado, com o objetivo de obter a representação mais fidedigna possível.