Computação Gráfica

Phase 2

GRUPO 6
A95917 - Eduardo Diogo Costa Soares - LCC
A95609 - Duarte Alexandre Oliveira Faria - LEI
A97941 - Diogo Filipe Oliveira da Silva - LEI
A98979 - Pedro Domingues Viana - LCC

Introdução

Nesta segunda fase do projeto foram modificados:

- O generator para poder criar torus.
- O engine para ler ficheiros XML com grupos dentro de grupo, criando uma hierarquia.

Também criamos, por fim, um ficheiro xml com informação do sistema solar para ser gerado.

Mudanças no generator

Como dito antes, no generator adicionamos o torus.
O **torus** assume a forma de um donut e para isso precisa de precisa de:

- outer radius (raio do anel)
- rings (número de lados do anel)
- inner radius (raio do círculo em volta do anel)
- slices (número de lados do círculo)

O número de vértices necessários para gerar o torus é:

$$slices \ * \ rings \ * \ 6$$

Analisando a imagem, podemos ver que o modelo é constituído por sequências de quadrados que avançam pelo anel num ângulo $ringStep \ = \ 2 \ * \ PI/rings$ e consequentemente $sliceStep \ = \ 2 \ * \ PI/slices$.

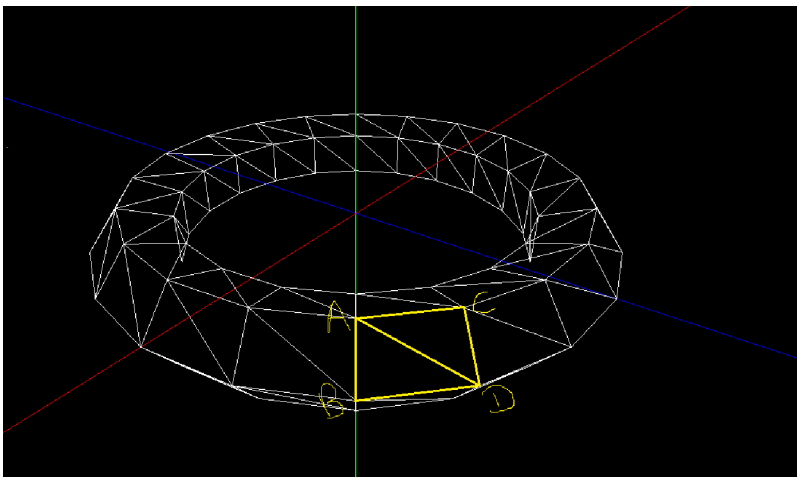
$A(x1, y1, z1)$
$x1 \ = \ (outer\_radius \ + \ inner\_radius \ * \ cos(slice\_angle)) \ * \ cos(ring\_angle)$
$y1 \ = \ inner\_radius \ * \ sin(slice\_angle);$
$z1 \ = \ (outer\_radius \ + \ inner\_radius \ * \ cos(slice\_angle)) \ * \ sin(ring\_angle)$

$B(x2, y2, z2)$
$x2 \ = \ (outer\_radius \ + \ inner\_radius \ * \ cos(slice\_angle)) \ * \ cos(next\_ring\_angle)$

$$y2 = inner\_radius * sin(slice\_angle)$$
$$z2 = (outer\_radius + inner\_radius * cos(slice\_angle)) * sin(next\_ring\_angle);$$

$$C(x3, y3, z3)$$
$$x3 = (outer\_radius + inner\_radius * cos(next\_slice\_angle)) * cos(next\_ring\_angle)$$
$$y3 = inner\_radius * sin(next\_slice\_angle);$$
$$z3 = (outer\_radius + inner\_radius * cos(next\_slice\_angle)) * sin(next\_ring\_angle)$$

$$D(x4, y4, z4)$$
$$x4 = (outer\_radius + inner\_radius * cos(next\_slice\_angle)) * cos(ring\_angle)$$
$$y4 = inner\_radius * sin(next\_slice\_angle)$$
$$z4 = (outer\_radius + inner\_radius * cos(next\_slice\_angle)) * sin(ring\_angle)$$



Como podemos ver na imagem estes quadrados são constituídos por dois triângulos,"A - D - B e A - C - D" garantindo uma ordem anti-horário para que a normal fique virada para o lado exterior.

Mudanças no engine

```
<group>
    <transform>
            <translate x="0" y="1" z="0" />
    </transform>
    <models>
        <model file="box.3d" /> <!-- generator box 2 3 box_2_3.3d -->
    </models>
                <group>
                        <transform>
                                <translate x="0" y="1" z="0" />
                        </transform>
                        <models>
                                <model file="cone.3d" /> <!-- generator cone 1 2 4 3 cone_1_2_4_3.3d -->
                        </models>
                        <group>
                                <transform>
                                        <translate x="0" y="3" z="0" />
                                </transform>
                                <models>
                                        <model file="sphere.3d" /> <!-- generator sphere 1 8 8 sphere_1_8_8.3d -->
                                </models>
                        </group>
                </group>
        </group>
```

O nosso engine precisa agora de conseguir fazer parse à informação deste tipo.

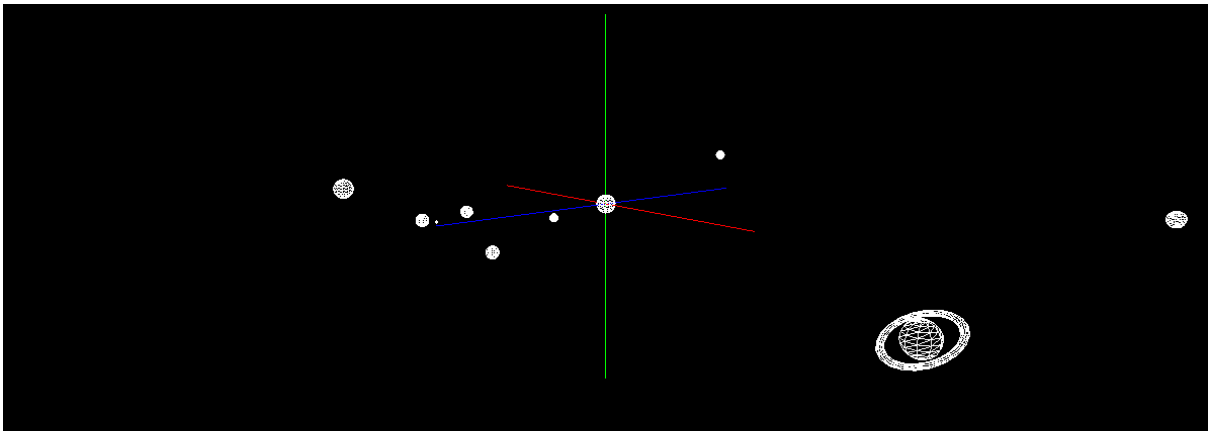
Agora para além de termos os modelos a serem carregados temos também as transformações a serem aplicados nesses modelos, este conjunto de informações “group” pode conter outro(s) group(s) dentro do mesmo que herdam as transformações dos seus antecessores.

Para isto nós achamos que a melhor maneira de guardar a informação dos modelos e das transformações em memória seria criando um objeto que poderia guardar o nome dos modelos, uma transformação de cada tipo (translação, rotação e escala) e uma lista do próprio objeto.

Por fim só teríamos que iterar pela(s) nossa(s) árvore(s) (dependendo de quantos groups sem antecessor é que temos) de groups em DFS dando push as transformações para a matriz e dando apenas pop quando chegássemos a uma das folhas ou quando os filhos do group atual já foram processados.

Sistema Solar

Para simular o sistema solar, foi feito um ficheiro xml com o sol como group com mais hierarquia, os planetas como seus filhos diretos, e as luas respectivas dos planetas como filhos dos mesmos.
Obviamente, dado que a distância relativa entre os planetas é demasiado grande, o sistema não está à escala.
Foram feitos dois exemplos do sistema solar.
Um em que não há rotação nenhuma dos planetas, e estão completamente alinhados, ou seja, não há rotações.
O outro tem as posições relativas dos planetas do dia 28 de março de 2025, já que o primeiro parecia um exemplo muito específico, o segundo também o é.



Testar

Para correr assumindo que está no diretório do projeto(ou seja no CG-main) basta fazer:

- make
- ./Generator/generator nome [argumentos do nome] nome.3d
- ./Engine/engine teste_desejado