Universidade do Minho

Licenciatura em Ciências da Computação

Computação Gráfica

Phase 2 - Geometric Transforms

Grupo Nº 1

Bruna Carvalho (a87982) | Carlos Beiramar (a84628) | Daniel Ferreira (a85670) | Ricardo Cruz (a86789)

31 de março de 2021

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

O objetivo proposto nesta segunda fase do trabalho prático é a criação de cenários hierárquicos usando transformações geométricas, isto é, um cenário definido como uma árvore onde cada nodo contém um conjunto de transformações geométricas (*translações, rotações e escalas*) e, opcionalmente um conjunto de modelos (esfera, cone,etc). Cada nodo pode ter vários "subnodos". As transformações geométricas só podem estar dentro do nodo/classe *group* e, nesta classe, tem, ainda, como subclasses *model* e subgrupos *group*.
Para isso, foram necessárias algumas alterações na aplicação ***engine***, de modo a que pudesse ler, de um ficheiro *XML*, o cenário pretendido com as suas subclasses.
Para além disso, era pretendido que nesta fase fosse criado um cenário de um Sistema Solar, num ficheiro *XML*, em que inclui o sol, os planetas e as luas, definida pela sua hierarquia.
A ordem das transformações geométricas é relevante, sendo que o nosso grupo optou pela seguinte ordem:

1. Translação
2. Rotação
3. Escala

Mais adiante, iremos apresentar, de forma pormenorizada, o processo de elaboração deste cenário hierárquico, tanto na escrita do ficheiro *XML* como na leitura deste mesmo ficheiro.

Capítulo 2

Alterações no *Engine*

2.1 Classe *Group*

Esta classe foi criada para representar cada elemento *group* no ficheiro *XML*. Nesta classe vão ser guardados os pontos necessários para realizar os desenhos e, também, as possíveis transformações geométricas presentes dentro de cada elemento *group*, no ficheiro *XML* correspondente ao cenário que vai ser desenhado.

```
class Group {

public:
        std::vector<float> translation;
        std::vector<float> rotation;
        std::vector<float> scale;
        std::vector<std::vector<float>> trianglesCoordinates;
```

Figura 2.1: Classe *Group*

2.2 Classe *Scene*

Esta classe, originalmente criada na **Fase 1** para representar o desenho completo do ficheiro *XML*, foi adaptada para esta fase. Agora, para além de conter um vetor com os nomes dos ficheiros ".3d"que vão ser desenhados, passou também a conter uma *hashtable*, de maneira a associar uma lista de *Group* a cada ficheiro ".3d".

```
class Scene {
public:
        std::vector<std::string> files; //nomes ficheiros 3d
        std::unordered_map<std::string, std::vector<Group*>> data;
```

Figura 2.2: Classe *Scene*

2.3 Leitura do novo formato *XML*

Dado que as transformações geométricas só podem existir dentro de elementos *group* e são aplicadas a todos os modelos e subgrupos desse elemento, a leitura ficheiro *XML* foi feita de forma recursiva (conceito de herança).

```
void readFromXmlRec(XMLElement* element, std::vector<float> trans, std::vector<float> rot, std::vector<float> scal) {
    for (XMLElement* next = element; next != NULL; next = next->NextSiblingElement()) {
        if (strcmp(next->Name(), "rotate") == 0) {
            if (next->FindAttribute("angle")) {
                rot[0] += atof(next->FindAttribute("angle")->Value());
            }
            if (next->FindAttribute("X")) {
                rot[0] += atof(next->FindAttribute("X")->Value());
            }
            if (next->FindAttribute("Y")) {
                rot[1] += atof(next->FindAttribute("Y")->Value());
            }
            if (next->FindAttribute("Z")) {
                rot[2] += atof(next->FindAttribute("Z")->Value());
            }
        }

        else if (strcmp(next->Name(), "translate") == 0) {

            if (next->FindAttribute("X")) {
                trans[0] += atof(next->FindAttribute("X")->Value());
            }
            if (next->FindAttribute("Y")) {
                trans[1] += atof(next->FindAttribute("Y")->Value());
            }
            if (next->FindAttribute("Z")) {
                trans[2] += atof(next->FindAttribute("Z")->Value());
            }
        }
```

Figura 2.3: *readXMLRec1*

```
            else if (strcmp(next->Name(), "scale") == 0) {
                    if (next->FindAttribute("X")) {
                            scal[0] *= atof(next->FindAttribute("X")->Value());
                    }
                    if (next->FindAttribute("Y")) {
                            scal[1] *= atof(next->FindAttribute("Y")->Value());
                    }
                    if (next->FindAttribute("Z")) {
                            scal[2] *= atof(next->FindAttribute("Z")->Value());
                    }
            }

            else if (strcmp(next->Name(), "models") == 0) {
                    for (XMLElement* model = next->FirstChildElement(); model != NULL; model = model->NextSiblingElement()) {
                            std::string filename = model->Attribute("file");
                            files.push_back(filename);
                            addFile(filename, trans, rot, scal);
                    }
            }

            else {
                    readFromXmlRec(next->FirstChildElement(), trans, rot, scal);
            }
    }
}
```

Figura 2.4: *readXMLRec2*

Capítulo 3

Sistema Solar em formato *XML*

Este capítulo irá conter os passos necessários para a criação de um Sistema Solar. Para isso, foi desenvolvido um cenário hierárquico, tendo como subclasses *group*, que contém as transformações geométricas e os *model* (esferas que representam, o sol, os planetas e as luas) e, também, poderá ter subclasses *group*. Inicialmente, os planetas foram desenhados a partir de uma escala real, todavia, para que fosse possível visualizar todos os planetas no mesmo plano a escala foi adaptada.

```
<scene>
    <group>
        <group id = "Sol">
            <translate X="0" Y="0" Z="0" />
            <scale X="20" Y="20" Z="20" />
            <models>
                <model file="sphere.3d" />
            </models>
        </group>

        <group id = "Mercurio">
            <translate X="28" Y="0" Z="0" />
            <scale X="0.4" Y="0.4" Z="0.4" />
            <models>
                <model file="sphere.3d" />
            </models>
        </group>

        <group id = "Venus">
            <translate X="35" Y="0" Z="0" />
            <scale X="1.74" Y="1.74" Z="1.74" />
            <models>
                <model file="sphere.3d" />
            </models>
        </group>

        <group id = "Terra">
            <translate X="41" Y="0" Z="0" />
            <scale X="1.83" Y="1.83" Z="1.83" />
            <models>
                <model file="sphere.3d" />
            </models>
```

Figura 3.1: *Cenário do Sistema Solar em XML*

Capítulo 4

Resultados: Sistema Solar

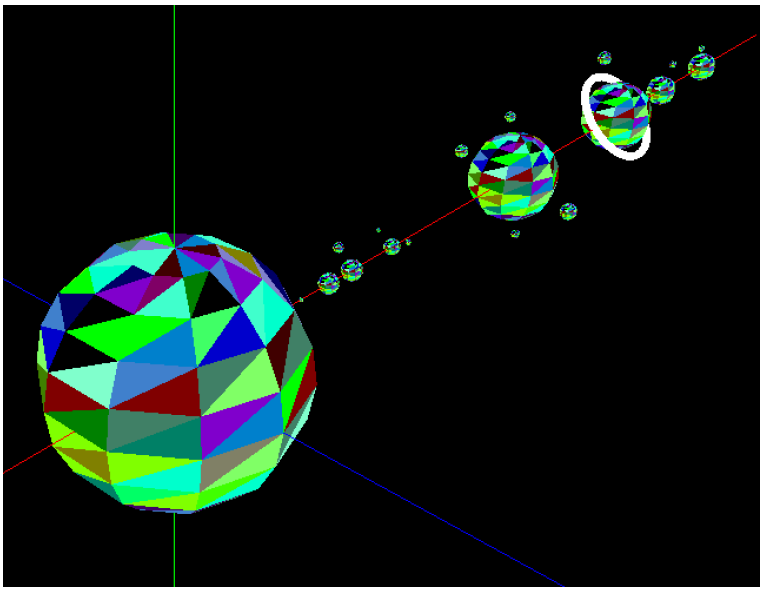

Figura 4.1: *Sistema Solar*

Capítulo 5

Conclusão

Após a realização da segunda fase, obtivemos uma melhor compreensão sobre como criar cenários complexos através de transformações individuais ou combinadas.
No entanto, sentimos dificuldades em determinar as distâncias entre cada planeta e os seus respetivos tamanhos pois, o objetivo era desenhar o mais próximo com a realidade.
Concluindo, esta fase correu de forma esperada e conseguimos cumprir os requisitos necessários para completar a mesma.