Índice

1. Introdução

Dando seguimento ao trabalho desenvolvido na primeira fase introduzimos as transformações geométricas na definição da cena para que seja criado um modelo hierárquico. Vamos considerar que a cena é definida como uma árvore, em que cada nodo tem uma conjunto de transformações geométricas (translação, rotação e escala) e um conjunto de modelos associados.

```
<scene>
	<group>
		<translate X=5 Y=0 Z=2 />
		<rotate angle=45 axisX=0 axisY=1 axisZ=0 />
		<models>
			<model file="sphere.3d" />
		</models>
	</group>
</scene>
```

Figura 1 - Exemplo de ficheiro XML com um grupo

```
<scene>
	<group>
		<translate X=1 />
		<models>
			<model file="sphere.3d" />
		</models>
		<group>
			<translate Y=1 />
			<models>
				<model file="cone.3d" />
			</models>
		</group>
	</group>
</scene>
```

Figura 2 – Exemplo de ficheiro XML com uma hierarquia de grupos

2. Transformações geométricas

3.1 Translação

Translação é o movimento que um objeto realiza de um ponto para outro.[1]

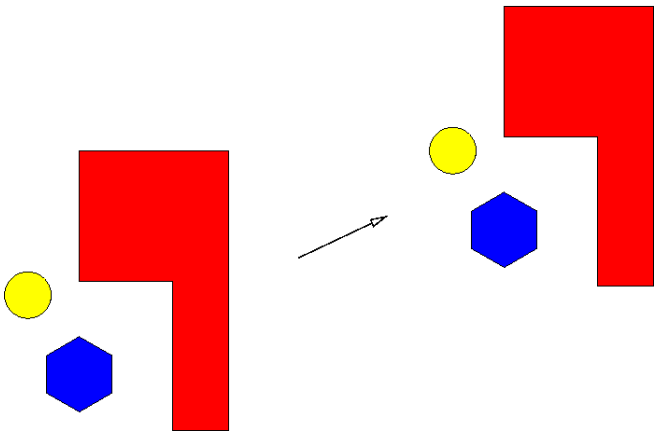

Figura 3 - Translação de três objetos

Pode ser visto num sistema de equações:

$$\begin{cases} x' = x + \Delta x \\ y' = y + \Delta y \\ z' = z + \Delta z \end{cases}$$

Em termos matriciais é visto como:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

[1] Fonte: https://pt.wikipedia.org/wiki/Translação

3.2 Rotação

Rotação é a figura que, sem sair da origem, vai rodando em diferentes graus definindo a sua posição final. 2

Dado um ponto (x,y,z) e um angulo θ a sua rotação sobre o eixo Y pode ser visto da seguinte forma:

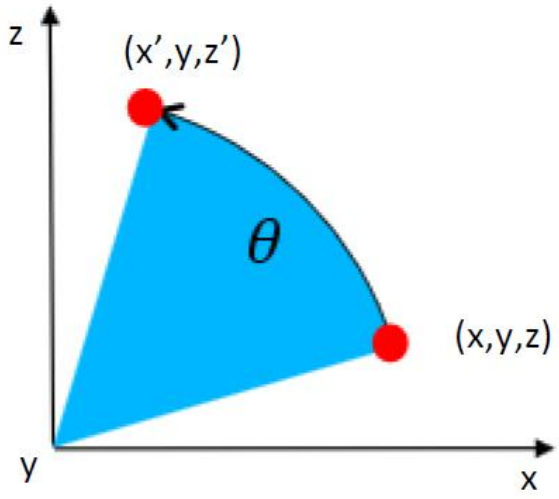

Figura 4 - Rotação de um ponto

Sistema de equações:

$$\begin{cases} x' = x\cos\theta + y\sin\theta \\ y' = y \\ z' = -x\sin\theta + y\cos\theta \end{cases}$$

Na sua forma matricial temos:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix} + \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

[2] https://pt.wikipedia.org/wiki/Rotação_(matemática)

3.3 Escala

As escalas permitem variar o tamanho de um objeto multiplicando cada ponto por um escalar.

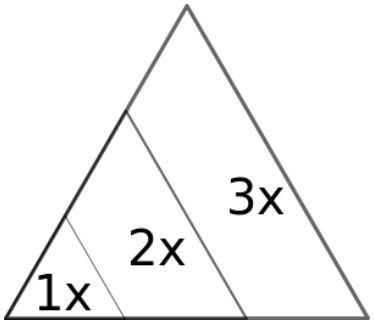

Figura 5 – Escala de uma imagem

Matematicamente:

$$\begin{cases} x' = \lambda_x x \\ y' = \lambda_y y \\ z' = \lambda_z z \end{cases}$$

Matricialmente:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \lambda_x & 0 & 0 & 0 \\ 0 & \lambda_y & 0 & 0 \\ 0 & 0 & \lambda_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

3. Motor 3D

O motor 3d é agora capaz de criar primitivas geométricas simples e aplicar transformações, tais como, translações, rotações e escalas às respetivas primitivas.

3.1 Classes

3.1.1 Point

A classe point guarda as coordenadas de um ponto. Tem três métodos "get" para cada uma das coordenadas e um construtor para criar objetos desta classe.

```
class Point {
    double x;
    double y;
    double z;
public:
    Point(double, double, double);
    double getX() { return x; }
    double getY() { return y; }
    double getZ() { return z; }
    virtual ~Point(void);
};
```

Figura 6 – Classe Point

3.1.2 Type

A classe type guarda o tipo das transformações de acordo com o numero de parâmetros. O construtor vazio coloca todas as variáveis a zero. O construtor com três parâmetros funciona para translações e escalas e o com quatro parâmetros para rotações.

```
class Type {
    float x;
    float y;
    float z;
    float ang;
public:
    Type();
    Type(float, float, float);
    Type(float, float, float, float);
    float getTX() { return x; }
    float getTY() { return y; }
    float getTZ() { return z; }
    float getTAng() { return ang; }
    bool tipoVazio();
    ~Type(void);
};
```

Figura 7 – Classe Type

3.1.3 Transformation

A classe transformation é composta pelas três transformações possíveis mais um parâmetro para a cor.

```
class Transformation {
    Type transl;
    Type rot;
    Type scale;
    Type color;
public:
    Transformation();
    Transformation(Type, Type, Type,Type);
    Type getTranslacao() { return transl; }
    Type getRotacao() { return rot; }
    Type getEscala() { return scale; }
    Type getCor() { return color; }
    void setTranslacao(Type t) { transl = t; }
    void setCor(Type t) { color = t; }
    void setEscala(Type t) { scale = t; }
    void setRotacao(Type t) { rot = t; }

    bool trasnformacaoVazia();
    ~Transformation(void);
};
```

Figura 8 – Classe Transformation

3.1.4 Model

Nesta classe é armazenado o nome do modelo e as respetivas transformações.

```
class Model {
    string nome;
    Transformation transf;
public:
    Model(string);
    vector<Point> pontos;
    string getNomeModelo() { return nome; }
    vector<Point> getPontos() { return pontos; }
    Transformation getTransformacao() { return transf; }
    void setTransformacao(Transformation t) { transf = t; }
    void adicionaPonto(Point p);
    virtual ~Model(void);
};
```

Figura 9 – Classe Model

4. Resultado

Foi criado um modelo do sistema solar para demonstração do motor em funcionamento. Todos os planetas têm a sua posição e escala relativa ao sol, e todas as luas tem a sua posição e escala relativa ao seu planeta.

```
<scene name="Sistema Solar">
        <group name="Sun">
                <scale X="1" Y="1" Z="1" />
                <color R="1" G="0" B="1"/>
                <models>
                        <model file="sphere.3d" />
                </models>

                <group name= "Mercurio">
                        <scale X="0.035" Y="0.035" Z="0.035" />
                        <translate X="4"  />
                        <rotate angle="45" X="1" Y="1" Z="1"/>
                        <color R="1" G="0" B="0.3"/>
                        <models>
                                <model file="sphere.3d" />
                        </models>
                </group>

                <group name= "Venus">
                        <scale X="0.086" Y="0.086" Z="0.086" />
                        <translate X="7"  />
                        <rotate angle="45" X="1" Y="1" Z="1"/>
                        <color R="0.6" G="0.4" B="0.3"/>
                        <models>
                                <model file="sphere.3d" />
                        </models>
                </group>

<group  name= "Terra">
        <scale X="0.09" Y="0.09" Z="0.09" />
        <translate X="12.16"  />
        <color R="0" G="1" B="0"/>
        <models>
                <model file="sphere.3d" />
        </models>

        <group name="Lua">
                <scale X="0.272" Y="0.272" Z="0.272" />
                <translate X="0.7" Y="0.5" Z="0.7"  />
                <color R="1" G="2" B="24"/>
                <models>
                        <model file="sphere.3d" />
                </models>
        </group>
</group>
```

Figura 10 – Ficheiro XML do sistema solar

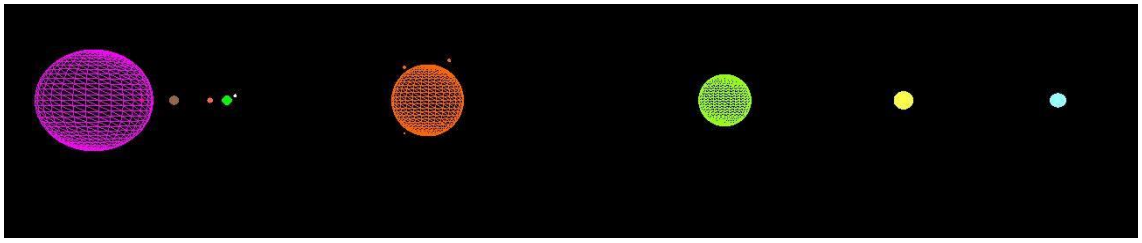

Figura 11 – Demonstração do sistema solar

5. Conclusão e trabalho futuro

No final da segunda fase do trabalho prático consideramos que os objetivos propostos foram atingidos com sucesso. O modelo do sistema solar prova que o motor 3D faz o que é pedido.

O sistema solar é ainda muito simples e há vários pormenores que devem ser melhorados no futuro, tais como rotação e translação dos planetas, acrescentar os anéis a Saturno, entre outros.

Em suma, achamos que o trabalho foi concebido com satisfação tendo em conta que foram atingidas as metas impostas.