Universidade do Minho

LEI 3ºAno 2ºSemestre

Computação Gráfica

Mini Motor 3D

Fase 2 - Transformações Geométricas

Daniel Caldas a67691
José Cortez a67716
Marcelo Gonçalves a67736
Ricardo Silva a67728

10 de Abril de 2015

Conteúdo

1 Introdução

1.1 Contexto

No âmbito da UC de Computação Gráfica, surge um projeto prático para consolidar os conceitos adquiridos ao longo do semestre: a criação de um mini motor 3D.

Usando as tecnologias C++ e OpenGL, será então desenvolvido o motor 3D ao longo de quatro fases, sendo que esta segunda está resumida em tópicos na subsecção seguinte.

1.2 Resumo

Nesta etapa foi efetuada:

- Alteração do motor 3D previamente desenvolvido, para que fosse possível efetuar transformações geométricas para a definição da cena por forma a criar um modelo hierarquico.
- Criação de um modelo do sistema solar estático.

2 Estrutura de dados

De maneira a que as formas geométricas a desenhar numa dada cena, podessem sofrer transformações geométricas pré-definidas, foi criada uma estrutura de dados que representará **opcionalmente** umas das seguintes três transformações:

- **Escala** - em *Ponto3D escala* são guardadas escalas a aplicar a uma dada forma;
- **Translação** - em *Ponto3D translacao* são guardadas as translações;
- **Rotações** - em *Rotacao3D rotacoes* são guardadas as rotações;

```
// Um objecto TransformsWrapper representa opcionalmente
class TransformsWrapper {
 public:
    string nome;
    Ponto3D escala;
    Ponto3D translacao;
    Rotacao3D rotacao;
    void printT() const {
        if (nome=="SCALE"){
            escala.printP();
        }
        else if (nome=="ROTATE"){
            rotacao.printR();
        }
        else if (nome=="TRANSLATE"){
            translacao.printP();
        }
    }
};
```

Figura 1: **Classe TransformsWrapper**.

2.1 Rotacao3D

A partir da classe *Ponto3D* **estendemos** um novo tipo de dados, **Rotacao3D** que ao seu super tipo acrescenta uma variável para guradar o valor do **ângulo** associado à rotação.

```
// Uma rotação 3D é uma extensão de um ponto 3D com o acréscimo de um ângulo
class Rotacao3D : public Ponto3D {
  public:
    Rotacao3D();

    float ang;
    void printR() const; // debug
};
```

Figura 2: **Classe Rotacao3D**.

2.2 Ficheiro XML

As transformações serão processadas pela motor fazendo-se a leitura de um ficheiro fonte XML com o formato sugerido pelo docente da UC, mas com as seguintes modificações que de certa forma **facilitaram a codificação do motor**:

- **Translações** - optamos por colocar **sempre** todos os valores das coordenadas x, y e z;
- **Cores** - acrescentamos ao **modelo** a cor com que se pretende pintar a respetiva forma geométrica;
- **Profundidade** - acrescentamos um **atributo** profundidade para facilitar o processo de *parsing* de um ficheiro input, este atributo permite a um dado grupo de modelos herdar ou transformações geométricas consoante as relações com grupos anteriores;

```xml
<?xml version="1.0" ?>
<imagem>
        <grupo prof=1>
            <translacao X=10 Y=0 Z=0 />
            <rotacao angulo=45 eixoX=1 eixoY=0 eixoZ=0 />
            <modelos>
                <modelo ficheiro="cone.3d" colorX=1 colorY=0 colorZ=1 />
            </modelos>
            <grupo prof=2>
                <translacao X=-5 Y=0 Z=10 />
                <modelos>
                    <modelo ficheiro="cone2" colorX=0 colorY=1 colorZ=1 />
                    <modelo ficheiro="esfera.3d" colorX=1 colorY=1 colorZ=0 />
                </modelos>
                <grupo prof=2>
                    <translacao X=-5 Y=0 Z=-10 />
                    <modelos>
                        <modelo ficheiro="paralel.3d" colorX=0 colorY=0 colorZ=1 />
                    </modelos>
                </grupo>
                <grupo prof=-1>
                </grupo>
            </grupo>
        </grupo>
</imagem>
```

Figura 3: **Exemplo de ficheiro XML**.

Para armazenar os grupos de modelos e povoamento das estruturas de dados foi criada uma classe **Grupo** que agrega informação acerca das transformações geométricas do referido grupo.

```
class Grupo {
 public:
     Grupo(){};
     Grupo(vector<TransformsWrapper> fromfather){ transfs = fromfather; }
     int id; // noção de profundidade no ficheiro
     vector<TransformsWrapper> transfs;
};
```

Figura 4: **Classe Grupo**.

2.3 Ordem das transformações

Durante a leitura do ficheiro é garantida a **ordem das transformações geométricas** para que posteriormente seja garantida a sequencialidade da aplicação das diversas transformações geométricas.

2.4 Aplicação das transformações

Para aplicarmos as transformações a cada forma foi definido um método comum que poupa espaço e torna mais claro o métod *draw()* de cada uma das formas.

```
void Forma::applyTransforms()
{
    int i, n;

    n = transforms.size();
    glPushMatrix();
    for (i = 0; i < n; i++){
        TransformsWrapper tw = transforms.at(i);
        const char* op = tw.nome.c_str();

        if (strcmp(op, ROTATE) == 0){
            glRotatef(tw.rotacao.ang, tw.rotacao.x, tw.rotacao.y, tw.rotacao.z);
        }
        else if (strcmp(op, TRANSLATE) == 0){
            glTranslatef(tw.translacao.x, tw.translacao.y, tw.translacao.z);
        }
        else if (strcmp(op, SCALE) == 0){
            glScalef(tw.escala.x, tw.escala.y, tw.escala.z);
        }
    }
}
```

Figura 5: **applyTransforms()** método que aplica as tranformações geométricas.

Na figura 5 podemos observar como o ciclo for percorre a estrutura de transformações aplicando-as sequencialemente, **pois as mesmas foram armazenadas pela ordem pré-definida no ficheiro de entrada.**

2.5 Exemplo

2.5.1 Input

```
<?xml version="1.0" ?>
<imagem>
        <grupo prof=1>
            <translacao X=10 Y=0 Z=0 />
            <rotacao angulo=45 eixoX=1 eixoY=0 eixoZ=0 />
            <modelos>
                <modelo ficheiro="cone.3d" colorX=1 colorY=0 colorZ=1 />
            </modelos>
            <grupo prof=2>
                <translacao X=-10 Y=0 Z=0 />
                <modelos>
                    <modelo ficheiro="esfera.3d" colorX=1 colorY=1 colorZ=0 />
                </modelos>
            </grupo>
            <grupo prof=-1>
            </grupo>
        </grupo>
</imagem>
```

Figura 6: Ficheiro XML input.

2.5.2 Output

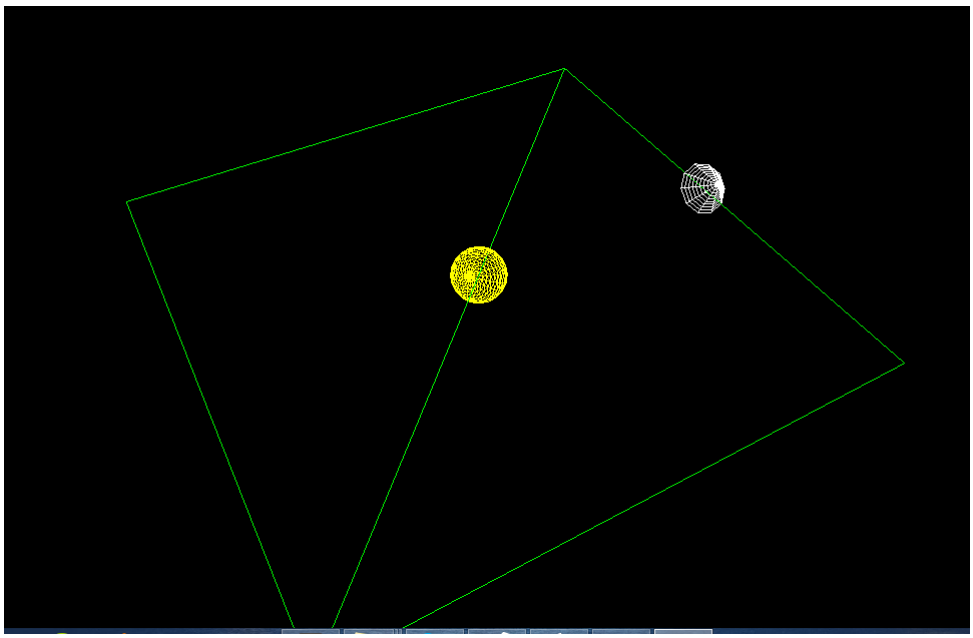

Figura 7: Cena resultante.

3 Sistema solar estático

Como nos é pedido, são entregues ficheiros com modelos diversos.

Nesta secção, iremos explicar o desenvolvimento seguido para implementar um sistema solar estático, os testes efectuados com os outros modelos, são entregues em anexo.

3.1 Escalas

Para o desenvolvimento deste modelo foi necessário usar uma escala apropriada para que se conseguisse visualizar todos os planetas.

Inicialmente, a dimensão dos corpos celestiais foi calculada através do logaritmo de base 10 dos respectivos raios. No entanto, os raios obtidos não eram do nosso agrado pois, por exemplo, Júpiter era quase do tamanho do sol.

Foram, então, ajustados os raios dos planetas de modo a que se pudesse visualizar facilmente algumas relações básicas do Sistema Solar, por exemplo, Mercúrio é o planeta mais pequeno, Júpiter é o maior, o Sol é bastante maior do que todos os planetas representados, etc.

No que às distâncias entre planetas diz respeito, foi utilizada, inicialmente, uma escala linear. No entanto, tal era impossível pois, dado a escala das dimensões, Mercúrio ficaria sempre dentro do Sol.

Tal como no ponto anterior, foi, então, ajustada essa escala de modo a que algumas relações mais óbvias batessem certo, por exemplo, planetas rochosos estão bastante próximos uns dos outros, maior espaçamento entre planetas gasosos, etc.

Finalmente, para a representação da Lua, foi utilizada uma escala linear (raio da lua = 25% do raio da terra).

3.2 Ficheiro XML

O ficheiro XML que nos permite implementar um sistema solar estático segue as regras já previamente definidas, como podemos observar pela figura 8.

Só está representado parte do ficheiro, mas para a representação dos outros planetas usou-se sempre a mesma técnica.

```
<?xml version="1.0" ?>
<imagem>
    <grupo prof=1>
        <modelos>
            <modelo ficheiro="plano.3d" colorX="1" colorY="1" colorZ="1" />
        </modelos>
        <grupo prof=2>
            <translacao X="0" Y="17" Z="0" />
            <modelos>
                <modelo ficheiro="sol.3d" colorX="1" colorY="0.5" colorZ="0" />
            </modelos>
            <grupo prof=3>
                <translacao X="0" Y="0" Z="23"  />
                <modelos>
                    <modelo ficheiro="mercurio.3d" colorX="0.75" colorY="0.75" colorZ="0.75" />
                </modelos>
            </grupo>
            <grupo prof=3>
                <translacao X="0" Y="0" Z="30" />
                <modelos>
                    <modelo ficheiro="venus.3d" colorX="0.73" colorY="0.73" colorZ="0" />
                </modelos>
            </grupo>
            <grupo prof=3>
                <translacao X="0" Y="0" Z="38"  />
                <modelos>
                    <modelo ficheiro="terra.3d" colorX="0" colorY="0.5" colorZ="1" />
                </modelos>
                <grupo prof=4>
                    <translacao X="3" Y="0" Z="0" />
                    <modelos>
                        <modelo ficheiro="lua.3d" colorX="0.75" colorY="0.75" colorZ="0.75" />
                    </modelos>
                </grupo>
            </grupo>
```

Figura 8: Ficheiro .xml, Sistema Solar.

3.3 Output

São agora apresentadas as imagens correspondentes ao sistema solar gerado, a figura 9, onde é apresentada a vista globar do Sistema Solar, e a figura 10, onde podemos ver de perto a Lua.

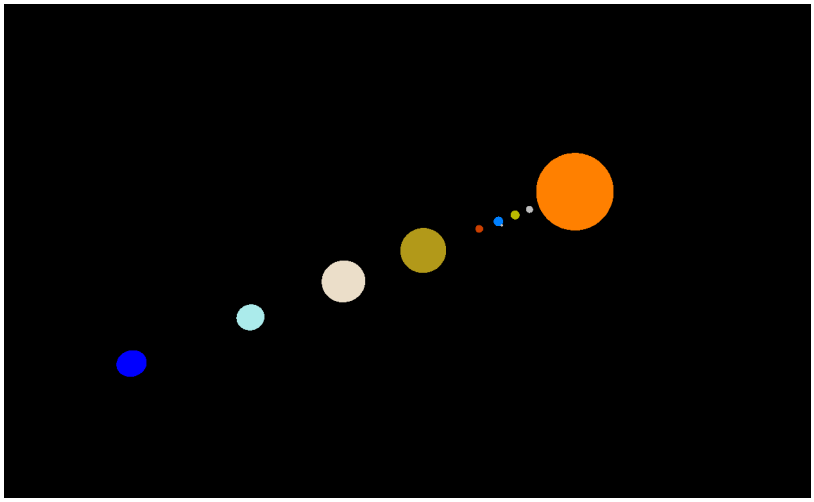

Figura 9: Vista global do Sistema Solar.

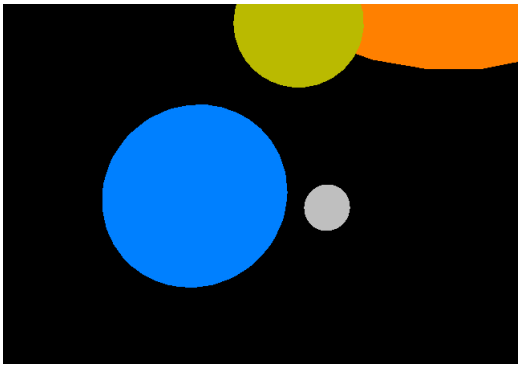

Figura 10: Vista da representação da Lua.

4 Conclusão

Nesta segunda fase treinamos, através das primitivas gráficas previamente desenvolvidas, as transformações geométricas, ou seja, nesta fase foi consulidado os conhecimentos adquiridos para a definição de cenas através de várias transformações geometricas.

Completamos um pouco mais o nosso mini motor 3D, de maneira a que fossem suportadas as transformações geometricas nesseçárias. Estamos agora familiarizados com a definição de cenas estáticas.