Licenciatura em Ciências da Computação

Computação Gráfica

Geometric Transforms

Filipe Barbosa A77252

Luís Bigas A76964

Hugo Ferreira A78555

17 de Março de 2019

Conteúdo

Capítulo 1

Introdução

1.1 Introdução

No âmbito da unidade curricular Computação Gráfica, foi proposta a extensão da primeira fase deste trabalho.

Nesta segunda fase foi alterada a sua funcionalidade para possibilitar a leitura de um ficheiro XML mais complexo que descreve uma cena tridimensional com a finalidade de ser impresso numa janela gráfica.
As ferramentas utilizadas na primeira fase do trabalho foram também usadas nesta segunda fase, C++, FreeGlut e TinyXML2.

Capítulo 2

Ficheiro XML

2.1 Estrutura

Como pedido no enunciado do trabalho, é utilizado um ficheiro XML com diferentes elementos numa certa relação hierática.
Mais especificamente na forma de uma árvore em que cada nodo representa uma transformação geométrica ou modelos geométrico que se pretende desenhar.

Os modelos geométricos que se pretende desenhar deverão ser gerados previamente pelo **generator** criado para a primeira fase do trabalho.

Nodos Possíveis no ficheiro XML:

- <scene>
- <group>
- <models>
- <model file="name_of_model_file" />
- <translate X="x_cords" Y="y_cords" Z="z_cords" />
- <scale X="x_scale" Y="y_scale" Z="z_scale" />
- <rotate angle="r_angle" X="x_value" Y="y_value"Z="z_value" />

2.1.1 Exemplo

Exemplo 1:

```
<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>
        <models>
            <model file="box.3d" />
        </models>
    </group>

</scene>
```

Exemplo 2:

```
<scene>
    <group>

        <group>
            <scale X="20" Y="20" Z="20" />
            <models>
                <model file="sphere.3d" />
            </models>
        </group>

        <group>
            <translate X="30" Y="0" Z="0" />
            <models>
                <model file="sphere.3d" />
            </models>
            <translate X="10" Y="5" Z="0" />
            <scale X="3" Y="3" Z="3" />
            <models>
                <model file="sphere.3d" />
            </models>
        </group>

    </group>
</sceme>
```

Capítulo 3

Engine

O programa engine criado para a primeira fase do trabalho foi Substancialmente alterado para realizar a leitura dos ficheiros XML com o novo formato.

3.0.1 Utilização

Compilação e Execução

Para compilar o ficheiro engine.cpp é usado o comando **make eng**
De seguida para executar o programa, tendo o ficheiro .xml no formato adequado, e os ficheiros de coordenadas dos objetos chamados no mesmo ficheiro XML, é executado o comando **./engine "nome_ficheiro".xml**

Controlo

A camera está focada na raiz das coordenadas.
Esta pode ser controlada da seguinte forma:

- J ->Reduzir o nível de zoom (distância do ponto (0,0,0))
- K ->Aumentar o nível de zoom
- Setas ->cada uma das setas direcionais do teclado altera o ângulo de visualização da cena, na direção correspondente à seta utilizada.

3.0.2 Funcionamento

O programa **engine** está dividido em duas partes, na primeira parte é feito o processamento (parsing) do ficheiro XML dado como input, e a impressão do pretendido no ecrã.

Processamento Ficheiro XML

Para realizar este processamento, utilizando a biblioteca TinyXML2, verificamos primeiro se existe o nodo **<scene>**. Sendo este encontrado é chamada a função **ParserRoot()** que irá percorrer os nodos **<group>** exteriores.
Após é chamada a função **ParserGroup()** que processará o conteúdo de cada um dos groups, assim como a possível chamada recursiva para estes.
Caso seja lido do o elemento **<models>** é então utilizada a função **ParserModels()** que percorre esta Tag e envia cada um dos seus elementos à próxima função **ParseObject()**.

Para guardar o conteúdo lido com estas funções foi utilizado um vector de estruturas que guarda em ordem que comandos funções do FreeGlut deverão ser chamadas e em que ordem.

Impressão no Ecrã

Para imprimir no ecrã os objetos indicados no ficheiro XML, é utilizada a função **newrenderScene()** que percorre o vector criado durante a Processamento do XML.
Para cada um dos elementos do vector é interpretado a função que é necessária ser chamada. Para a impressão de um objeto, o Programa lê as coordenadas do ficheiro que representa a forma a desenhar com a função **lerCoords()**.

3.1 Sistema Solar

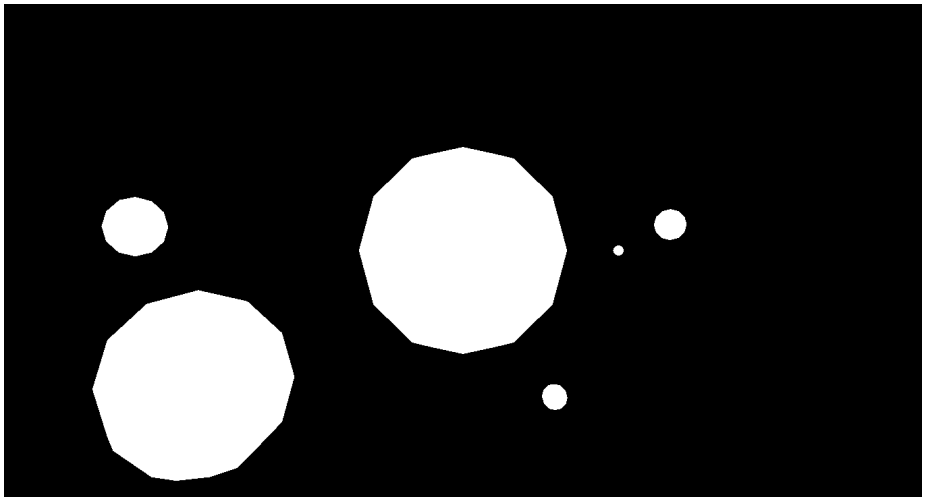
Como pedido no enunciado mostramos o funcionamento do programa com uma simples representação de um sistema solar.

XLM Utilizado:

```
<scene>
    <group>

        <group>
            <scale X="20" Y="20" Z="20" />
            <models>
                <model file="sphere.3d" />
            </models>
        </group>

        <group>
            <translate X="30" Y="0" Z="0" />
            <models>
                <model file="sphere.3d" />
            </models>
            <translate X="10" Y="5" Z="0" />
            <scale X="3" Y="3" Z="3" />
            <models>
                <model file="sphere.3d" />
            </models>
        </group>

        <group>
            <translate X="-55" Y="4" Z="15" />
            <scale X="5" Y="5" Z="5" />
            <models>
                <model file="sphere.3d" />
            </models>
        </group>

    </group>

    <group>
        <translate X="-20" Y="-10" Z="70" />
        <scale X="7" Y="7" Z="7" />
        <models>
            <model file="sphere.3d" />
        </models>
```

```
        <translate X="10" Y="-10" Z="-40" />
        <models>
            <model file="sphere.3d" />
        </models>
    </group>
</scene>
```

Resultado de execução:



./generator config.xml

Capítulo 4

Conclusão

Após a realização desta segunda fase do trabalho começamos por perceber o potencial da utilização das transformações geométricas aprendidas nas aulas nos gráficos computacionais.

Conseguimos realizar os objetivos pretendidos. Apesar de não ter acabado por adicionar todas as funcionalidades que desejaríamos, como a possibilidade de escolher a cor de cada um dos objetos tridimensionais.

Esperamos conseguir completar o trabalho com as funcionalidades que pretendemos no futuro.