Computação Gráfica

2ª Fase – Transformações Geométricas

André Geraldes 67673

Patrícia Barros 67665

Sandra Ferreira 67709

Conteúdo

[Conteúdo 2](#_Toc416450947)

[Índice de Figuras 3](#_Toc416450948)

[Introdução 4](#_Toc416450949)

[Desenvolvimento 5](#_Toc416450950)

[Classes 5](#_Toc416450951)

[Leitura e Análise do XML 6](#_Toc416450952)

[Função de Desenho 7](#_Toc416450953)

[Análise de Resultados 8](#_Toc416450954)

[Conclusão 9](#_Toc416450955)

Índice de Figuras

**Não foi encontrada nenhuma entrada do índice de ilustrações.**

Introdução

Este trabalho foi desenvolvido no âmbito da Unidade Curricular de Computação Gráfica, pertencente ao plano de estudos do 3º ano da licenciatura em Engenharia Informática.

Este projeto será constituído por 4 fases distintas com o objetivo final de criar um *motor 3D*. Nesta segunda fase foi-nos proposto fazer algumas alterações no trabalho desenvolvido na primeira fase, nomeadamente a implementação de transformações geométricas (rotações, translações e escalas).

Desenvolvimento

Após a análise do enunciado respetiva a esta fase do projeto, foi tomada a decisão de criar classes de modo a facilitar a compreensão e o manuseamento da informação presente no XML.

Classes

Foram construídas as classes **Ponto**, **Rotação**, **Translação**, **Escala**, **Transformação** e **Primitiva**. Para todas estas classes foram definidas funções que retornam o valor de cada variável bem como funções para definir o valor de cada uma delas (*gets* e *sets*).

A classe **Ponto** guarda apenas a informação relativa ao valor das variáveis do ponto, isto é, o valor de X, Y e Z.

A classe **Rotação** guarda a informação relativa ao ângulo da rotação e têm ainda três variáveis relativas aos eixos de rotação (eixo X, eixo Y e eixo Z). Por exemplo, se o valor da variável relativa ao eixo X for 0 quer dizer que a rotação não se efetua em torno desse eixo, pelo contrário, se for 1 quer dizer que sim.

A classe **Translação** guarda os valores das coordenadas do ponto para o qual o objeto deverá ser movido.

A classe **Escala** guarda três *floats* correspondentes aos valores da escala em relação a cada eixo.

A classe **Transformação** guarda um objeto do tipo Translação, um do tipo Rotação e um do tipo Escala. Em cada uma destas variáveis este guarda um objeto do tipo que lhe corresponde.

A classe **Primitiva** guarda toda a informação necessária para o desenho de uma determinada primitiva, ou seja, o seu nome, a transformação que ela irá sofrer, os seus filhos (cada um deles é por si uma primitiva) e os pontos necessários ao seu desenho.

Leitura e Análise do XML

Após a criação das classes o próximo passo foi adaptar o *parsing* do XML que tínhamos da fase anterior para funcionar com as novas condições. Para isso, para cada grupo, com o auxílio dos métodos **FirstChildElement()** que nos dá a primeira *tag* no nível imediatamente abaixo da anterior e **NextSiblingElement()** que nos dá aproxima *tag* com o mesmo nível da anterior, começamos por percorrer as suas transformações, verificar quais são os seus tipos e consoante isso guardar os valores das mesmas nas classes criadas para esse propósito. O próximo passo foi aplicar as transformações do nodo pai ao nodo filho. Para isso, à translação criada anteriormente somamos os valores das translações novas nas várias variáveis correspondentes e fazemos o mesmo para a rotação. Quanto à escala os valores anteriores são multiplicados pelos novos.

Com o auxílios dos mesmos métodos mencionados em cima é possível também saber quais os modelos que sofrem quais transformações, guardando essa informação num objeto da classe **Primitiva**, para que depois cada primitiva possa ser corretamente desenhada com as devidas transformações.

Função de Desenho

Para o desenho das primitivas iteramos o vetor das mesmas e para cada uma fazemos o **getTransformacao()**. Depois, utilizando a transformação obtida para ir buscar os valores de cada tipo de transformação e utilizá-los nas primitivas do *glut* **glRotatef(), glScalef()** e **glTranslated()**. Após efetuar as transformações procede-se ao desenho dos triângulos procedendo da mesma forma utilizada na primeira fase do projeto.

Análise de Resultados

Passamos então a mostrar os resultados que obtivemos com o nosso Motor3D para alguns ficheiros XML de teste criados:

PÔR AQUI OS PRINTS QUANDO ESTIVER TUDO A FUNCIONAR DIREITINHO

Criamos também um ficheiro XML com um esboço do Sistema Solar Estático. Não usamos valores reais para as escalas e distâncias entre os planetas mas sim valores arbitrários na ordem de grandezas que desejávamos visualizar. Segue-se o resultado:

PÔR AQUI O PRINT DO SISTEMA SOLAR

Conclusão

Nesta fase do projeto houve uma mudança de maneira de lidar com o *C++* em relação à fase anterior: começamos a explorar melhor as suas funcionalidades de linguagem orientada aos objetos através da criação das classes e utilização de objetos. Foi portanto útil e interessante para nós esta nova aprendizagem.

Para além disso foi muito útil para melhorar a nossa perceção do funcionamento das transformações geométricas.