

**67714**

**67741**

**67747**

**67740**

**João Pedro Monteiro Miranda**

**João Tiago Cruz Cunha**

**Rui Manuel Abreu Pereira**

**João Carlos Silva Monteiro**

**Universidade do Minho**

Escola de Engenharia

Mestrado Integrado em Engenharia Informática

**Computação Gráfica**

Ano Letivo de 2015/2016

**Relatório da 1ª Fase**

# Índice

[1. Introdução 1](#_Toc444881860)

[2. Gerador 2](#_Toc444881861)

[2.1. Estruturação do Código Fonte 3](#_Toc444881862)

[2.2. Algoritmos de Desenho 3](#_Toc444881863)

[2.2.1 Esfera 3](#_Toc444881864)

[2.2.2 Paralelepípedo 3](#_Toc444881865)

[2.2.3 Cone 4](#_Toc444881866)

[2.2.4 Plano 4](#_Toc444881867)

[3. Motor 5](#_Toc444881868)

[3.1. Ficheiro XML 6](#_Toc444881869)

[4. Estruturação do Projeto 7](#_Toc444881870)

[5. Exemplos 8](#_Toc444881871)

[5.1. Esfera 8](#_Toc444881872)

[5.2. Paralelepípedo 9](#_Toc444881873)

[5.3. Cone 9](#_Toc444881874)

[5.4. Plano 9](#_Toc444881875)

[5.4.1 XML com duas figuras 9](#_Toc444881876)

[6. Conclusão 11](#_Toc444881877)

# Índice de Figuras

[Figura 1 - Exemplo de um triângulo 2](file:///C:\Users\Rafael\Desktop\Relatório%20CG.docx#_Toc414637989)

[Figura 2 – Exemplo de um Plano 5](#_Toc414637990)

[Figura 3 - Ilustração da *Solution* no *Visual Studio 2013* 8](#_Toc414637991)

[Figura 4 . Exemplo de uma esfera 9](file:///C:\Users\Rafael\Desktop\Relatório%20CG.docx#_Toc414637992)

[Figura 5 - Exemplo de uma pirâmide quadrangular 10](file:///C:\Users\Rafael\Desktop\Relatório%20CG.docx#_Toc414637993)

[Figura 6 - Exemplo de um cilindro 10](file:///C:\Users\Rafael\Desktop\Relatório%20CG.docx#_Toc414637994)

[Figura 7 - Exemplo de um paralelepípedo 11](file:///C:\Users\Rafael\Desktop\Relatório%20CG.docx#_Toc414637995)

[Figura 8 - Exemplo de um cone 12](file:///C:\Users\Rafael\Desktop\Relatório%20CG.docx#_Toc414637996)

[Figura 9 - Exemplo de um plano 13](file:///C:\Users\Rafael\Desktop\Relatório%20CG.docx#_Toc414637997)

[Figura 10 - Duas figuras carregadas ao mesmo tempo 14](file:///C:\Users\Rafael\Desktop\Relatório%20CG.docx#_Toc414637998)

1. Introdução

Este trabalho prático está dividido em duas aplicações, tal como foi especificado no enunciado: Uma aplicação que cria os modelos 3D em ficheiros XML, e outra, que desenha cenas definidas em ficheiros XML, que referenciam os modelos 3D. Nesta 1ª fase, apenas é necessário criar os modelos com o gerador e desenhar múltiplos modelos com o motor. As aplicações foram desenvolvidas em C++, utilizando os recursos da biblioteca *OpenGL* e do *GLUT*

1. Gerador

O gerador é a aplicação responsável pela construção dos modelos em XML que irão ser desenhados pelo nosso motor gráfico. Estes modelos consistem num conjunto de triângulos que irão ser desenhados posteriormente. A aplicação é capaz de gerar planos, pirâmides, cones, paralelepípedos, esferas e cilindros. Para gerar modelos é preciso executar a aplicação na linha de comandos, especificando os argumentos, como é exemplificado no enunciado do trabalho prático. Todos os ficheiros produzidos serão guardados na diretoria do executável.

O ficheiro XML deve ter o seguinte formato:

<exemplo>

<triangle>

<vertex>X=0.5 Y=1 Z=2.5</vertex>

<vertex> X=0.5 Y=-1 Z=2.5</vertex>

<vertex> X=-0.5 Y=1 Z=2.5</vertex>

</triangle>

</exemplo>

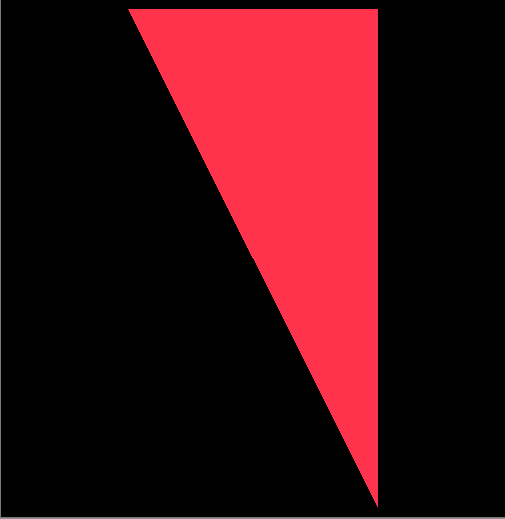


Figura 1 - Exemplo de um triângulo

Onde “exemplo” pode ser substituído por qualquer coisa, por exemplo, o nome da figura como nós fazemos nos nossos modelos. Normalmente os modelos são constituídos por vários grupos de triângulos. Triângulos com menos de 3 vértices são, obviamente, inválidos e irão ser rejeitados pelo motor. Este pequeno ficheiro XML produz a seguinte figura:

Todos os modelos serão desenhados na origem, tal como as funções do *OpenGL* e do *GLUT*. O nosso plano é uma superfície desenhada perpendicular ao eixo do Z. Os argumentos que deverão ser usados juntamente com a invocação da aplicação são os seguintes:

* esfera <raio> <camadas> <fatias> <ficheiro\_alvo>
* paralelepipedo <largura> <altura> <comprimento> <ficheiro\_alvo>
* cone <altura> <raio> <camadas> <fatias> <ficheiro\_alvo>
* plano <altura> <largura>
  1. Estruturação do Código Fonte

Toda a aplicação está presente no mesmo ficheiro fonte, “Gerador.cpp”, e é utilizado a biblioteca recomendada pelo docente, *tinyxml2*. Os códigos fonte da biblioteca é incluído diretamente no projeto. Pode ser encontrada documentação detalhada no próprio código, que trata de detalhes mais técnicos.

* 1. Algoritmos de Desenho
     1. Esfera

A esfera é constituída por diversas subdivisões longitudinais e latitudinais, fatias e camadas respetivamente. Utilizando coordenadas esféricas, são gerados pontos a uma distância *r* do centro da esfera que corresponde ao raio. As camadas correspondem ao número de subdivisões ao longo de *theta* e as fatias ao longo de *phi*. Antes de guardar cada vértice em ficheiro, estes são convertidos para coordenadas cartesianas usando as fórmulas abaixo representadas.

* + 1. Paralelepípedo

O paralelepípedo é formado por doze triângulos distribuídos em pares pelas seis faces do sólido geométrico. Estes triângulos são definidos apenas por oito pontos que correspondem as coordenadas dos oito vértices do paralelepípedo. Após os oito pontos estarem definidos, definem-se as faces.

* + 1. Cone

O processo de formação do cone é muito similar ao do cilindro, mas sendo necessário apenas definir a base inferior, da mesma forma que o cilindro, e a face lateral, que apesar de similar à do cilindro em cada camada os pontos tem de ser recalculados com um novo raio, com o valor do múltiplo de correspondente.

* + 1. Plano

O plano é constituído por quatro vértices que definem dois triângulos. Os vértices são definidos de maneira a que o centro do plano coincida com a origem do referencial e todas as coordenas em z são nulas.

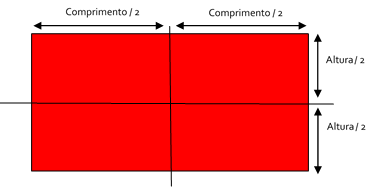


Figura 2 – Exemplo de um Plano

1. Motor

O nosso motor gráfico, desenvolvido a partir de um dos esqueletos fornecidos nas aulas práticas, é capaz de carregar ficheiros *XML* que definem dados de uma cena a desenhar. Estes ficheiros, nesta primeira fase, possuem referências aos modelos 3D feitos no gerador. Utilizando a biblioteca “*tinyxml2*”, processamos os ficheiros *XML* e desenhamos os modelos referenciados por estes.

Quando os ficheiros *XML* estão mal formados, a aplicação irá notificar o utilizador do erro respetivo. Estes erros podem ser:

* Erro de *parsing* do *XML* (formato *XML* erróneo);
* Erro de referência, que ocorre quando os ficheiros dos modelos 3D referenciados estão em falta.

Inicialmente, cada modelo será carregado para um vetor, ou seja, todos os modelos irão estar num vetor de vetores. Depois, na função de desenho *renderScene*, os modelos serão desenhados através do modo imediato (*glBegin* … *glEnd*). Desta forma, só se lê os modelos dos ficheiros uma vez, ficando depois em memória para serem redesenhados a qualquer momento.

De notar que estamos a utilizar *back face culling*, ou seja, os modelos lidos deverão ter os triângulos com a ordem dos vértices contra os ponteiros do relógio. O gerador define por defeito os vértices desta maneira.

Esta aplicação ainda possui algumas funcionalidades extras:

* *Zoom* com as teclas + e –;
* Navegação da câmara com as setas do teclado;
* Menu acessível através dum clique com o botão direito do rato, que permite especificar o modo de desenho (linha, preencher ou ponto).

A navegação com a câmara, incluindo o *zoom*, é feita usando coordenadas esféricas, onde as setas para cima e para baixo alteram o valor de *phi* e as setas para os lados afetam o valor de *theta*. A função de *zoom* é alcançada mudando o valor de *r*, ou seja, o raio.

* 1. Ficheiro XML

Para gerar uma cena, o ficheiro *XML* possui duas *tags*:

* <cena> - indica o inicio e fim da definição da cena;
* <modelo ficheiro= “ficheiro.3d”/> - indica um modelo a desenhar em que *ficheiro.3d* é o nome do ficheiro que contem o modelo *XML.*

O ficheiro deve seguir a seguinte sintaxe:

<cena>

<modelo ficheiro=”*ficheiro1”/>*

*<modelo* ficheiro=”*ficheiro2”/>*

*.*

*.*

*.*

*</cena>*

1. Estruturação do Projeto

Todo o projeto foi desenvolvido utilizando o *Visual* *Studio* *2013*, criando uma *solution* que contém os dois projetos. O projeto *Motor* utiliza, como é esperado, as bibliotecas do *GLUT* e do *OpenGL*. Fora isto, os projetos não possuem outras dependências, uma vez que a biblioteca do *parser* de XML vem incluída por defeito em ambos os projetos.



Figura 3 - Ilustração da *Solution* no *Visual Studio 2013*

1. Exemplos

De seguida iremos demonstrar alguns exemplos dos modelos que são gerados pelo nosso gerador. Para efeitos de demonstração irão ser desenhadas cenas só com o modelo em causa. As imagens poderão ser redimensionadas para efeitos de formatação deste relatório.

* 1. esfera1Esfera

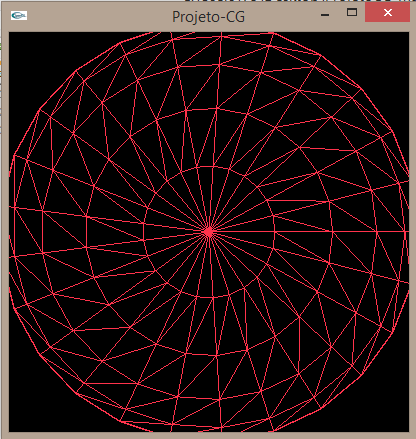


Figura 4 - Exemplo de uma esfera

* 1. Paralelepípedo
  2. Cone

Figura 8 - Exemplo de um cone

* 1. Plano

Figura 9 - Exemplo de um plano

* + 1. XML com duas figuras

Ficheiro: *cena.xml*

Código em XML:

<cena>

<modelo ficheiro="cone.3d"/>

<modelo ficheiro="cilindro.3d"/>

</cena>

1. Conclusão

Com a entrega desta primeira fase, podemos afirmar que cumprimos os objetivos por ela requisitados; uma aplicação que cria grande parte dos modelos mais básicos, e, outra responsável pelo processamento de ficheiros *XML* com dados de cenas a desenhar. Em relação aos aspetos que podiam ter sido melhorados é de relevo mencionar que os modelos desenhados poderiam ter sido criados utilizando *VBO's* ao contrário do modo imediato, que prevaleceu no nosso trabalho. De resto, não encontramos mais nenhuma falha de relevo ou outro qualquer comentário a acrescentar, sendo que podemos afirmar que realizamos um trabalho completo.