Universidade do Minho

Escola de Engenharia

Mestrado Integrado em Engenharia Informática

Computação Gráfica

Ano Letivo de 2015/2016

Relatório da 1ª Fase

João Pedro Monteiro Miranda	**67714**
João Tiago Cruz Cunha	**67741**
Rui Manuel Abreu Pereira	**67747**
João Carlos Silva Monteiro	**67740**

Índice

Índice de Figuras

1. Introdução

Esta unidade curricular, tem como objectivo introduzir uma nova perspectiva de programação. Esta permite-nos ter uma manipulação gráfica conseguindo através do OpenGL, uma API muito versátil utilizada no desenvolvimento de aplicativos gráficos, ambiente 3D, jogos,etc.

Para consolidar os conhecimentos adquiridos nas aulas, foi-nos solicitado a resolução de um projecto prático. O principal objectivo desta fase é criar duas aplicações para o desenvolvimento de uma mini cena baseado num motor 3D. A primeira aplicação denominada por “Gerador” irá ser responsável pela criação de ficheiro .3D de acordo com o input e também conterá toda a informação de cada modelo. Quanto a segunda aplicação denominada por “Engine”, esta irá interpretar um ficheiro .XML que conterá o nome dos modelos a criar e de seguida mostrará os modelos pedidos.

As ferramentas utilizadas para o desenvolvimento do software foi o *Visual Studio 2015* e, alem disso, a biblioteca “*tinyxml2*”.

2. Gerador

O gerador será a aplicação responsável pela informação sobre a construção dos modelos, esta aplicação será escrita em XML e terá de ser executada através da linha de comandos como exemplificado no enunciado. Esta aplicação recebe como argumentos o tipo de primitiva e as especificações necessárias a sua definição e tem como objectivo criar um ficheiro .3D onde estarão armazenadas os vértices usados no desenho da primitiva.

O ficheiro XML deve ter o seguinte formato, por exemplo:

```
<Figura>
   <Triangle>
      <Vertex X="-2.5" Y="0" Z="2.5"/>
      <Vertex X="2.5" Y="0" Z="2.5"/>
      <Vertex X="2.5" Y="0" Z="-2.5"/>
   </Triangle>
   <Triangle>
      <Vertex X="2.5" Y="0" Z="-2.5"/>
      <Vertex X="-2.5" Y="0" Z="-2.5"/>
      <Vertex X="-2.5" Y="0" Z="2.5"/>
   </Triangle>
</Figura>
```

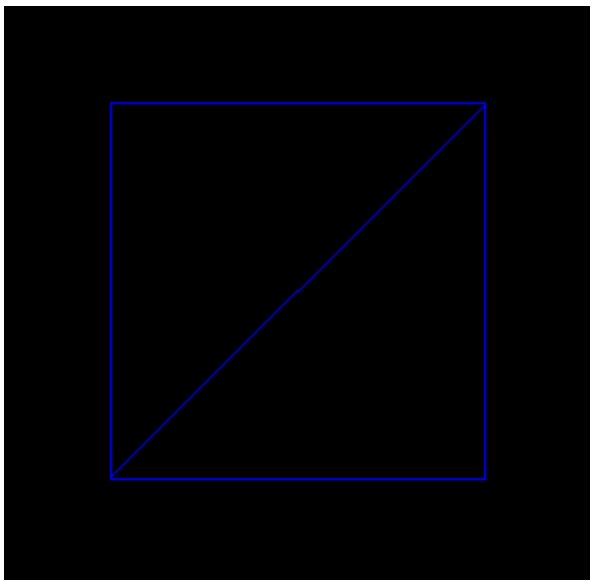


Figura 1 - Resultado do modelo

Onde "Figura" é o nome da figura do modelo desejado, neste caso, um plano. Todos os modelos são constituídos por vários grupos de triângulos. Triângulos com menos de 3 vértices são inválidos e não poderão se usados pelo motor.

Todos os modelos serão desenhados na origem, tal como as funções do *OpenGL* e do *GLUT*. O nosso plano é uma superfície desenhada no plano XZ. Os argumentos que deverão ser usados juntamente com a invocação da aplicação são os seguintes:

- `sphere <raio> <fatias> <camadas> <ficheiro>`
- `box <comprimento> <largura> <altura> <ficheiro>`
- `cone <raio> <altura> <fatias> <camadas> <ficheiro>`
- `plane <comprimento> <largura> <ficheiro>`

2.1. Código Fonte

A aplicação que gera os modelos está presente no ficheiro Gerador.cpp que inclui a biblioteca *tinyxml2*, como recomendada pelo professor.

2.2. Algoritmos de Desenho

2.2.1 Plano

O plano é constituído por quatro vértices que definem dois triângulos no plano XZ. Os vértices são construídos de maneira a que o centro coincida com a origem e com coordenadas em Y são nulas.

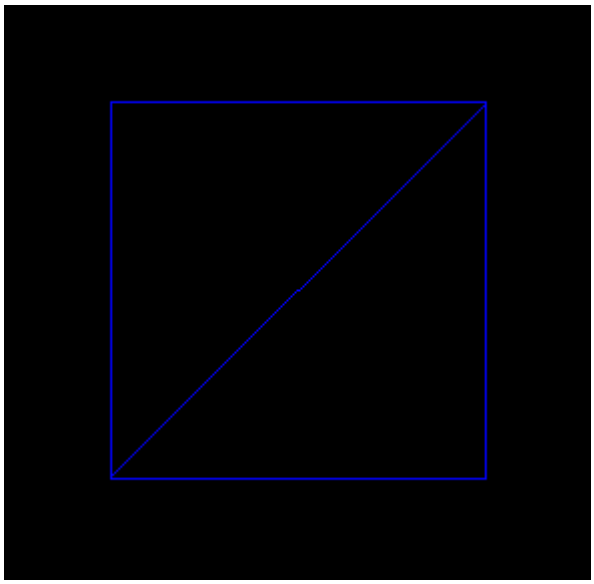


Figura 2 - Plano

2.2.2 Caixa

A caixa é formada 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.

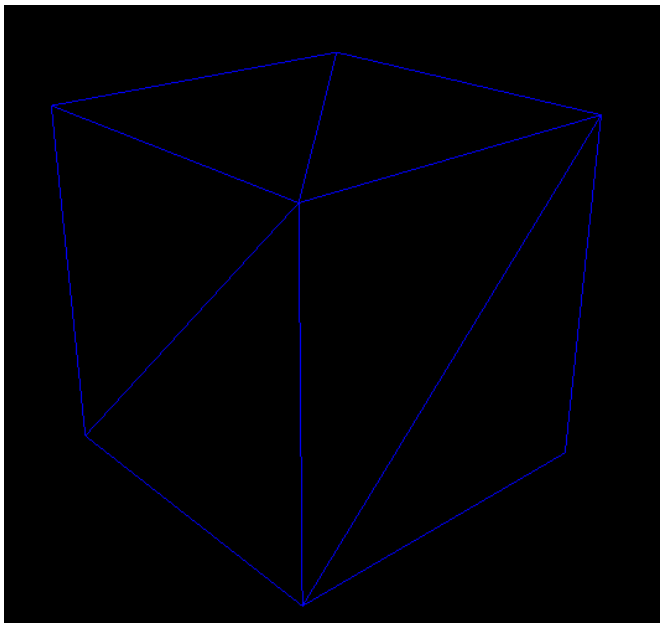


Figura 3 - Caixa

2.2.3 Esfera

A esfera é constituída por diversas subdivisões longitudinais e latitudinais, fatias e camadas respetivamente. Para a criação da esfera usamos coordenadas esféricas, são gerados pontos a uma distância r, que será o raio, do centro da esfera. 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.

$$x = r.sen(\theta).\text{sen}(\varphi)$$
$$y = r.\cos(\theta)$$
$$z = r.sen(\theta).\cos(\varphi)$$

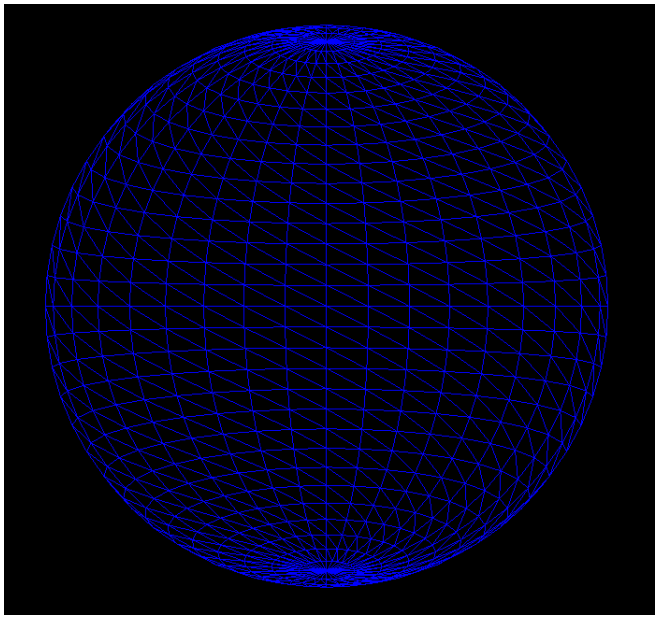


Figura 4 - Esfera

2.2.4 Cone

O cone é definido por uma circunferência na base e á medida que se acrescenta uma nova camada calcula se o novo raio sendo múltiplo de raio/camadas, que será obrigatoriamente menor que o raio da base, e que vai ser usado na construção das várias fatias laterais.

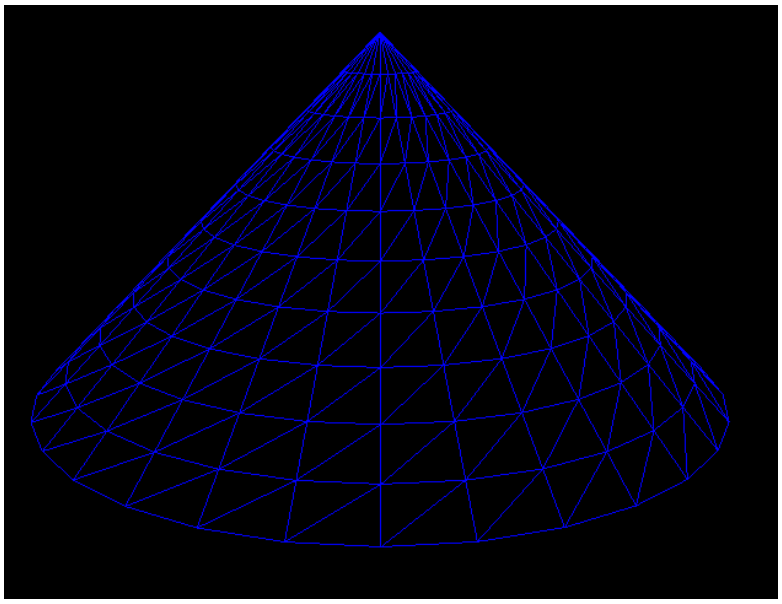


Figura 5 - Cone

3. Motor

O nosso motor gráfico, desenvolvido a partir de um dos esqueletos fornecidos nas aulas práticas, tem como função a leitura de um ficheiro .*XML* que definem os vértices de uma cena a desenhar. Para tal funcionalidade, optamos por dividir o Motor em duas classes: "Engine" e "SceneReader".

Na classe "SceneReader" onde procedemos a leitura do ficheiro .XML ,onde estarão os nomes dos ficheiros .3d a carregar, usando as funções das bibliotecas já mencionadas e uma vez encontrado o nome do ficheiro .3d procede-se ao desenho dos vértices.

Na classe "Engine" o seu esqueleto é idêntico ao que foi usado durante as aulas práticas contendo também a definição das funcionalidades extras.

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.

Inicialmente, cada modelo será carregado para um vetor, ou seja, todos os modelos irão estar num vetor de vetores. Depois, na função de desenho *drawScene*, 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 do teclado mais especificamente as teclas 'O' e 'L', que permite especificar o modo de desenho (preencher ou linhas respectivamente).

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.

3.1. Ficheiro XML

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

- `<Scene>` - indica o inicio e fim da definição da cena;
- `<model file= "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:

```
<Scene>
      <model file="ficheiro1"/>
      <model file="ficheiro2"/>
      .
      .
      .
</Scene>
```

4. Estruturação do Projeto

Todo o projeto foi desenvolvido utilizando o *Visual Studio 2015*, criando uma *solution* que contém os dois projetos. O projeto *Engine* 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.

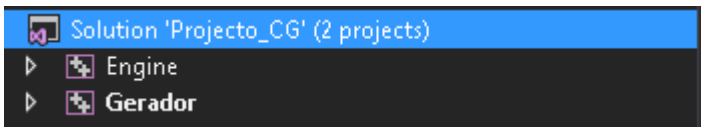


Figura 6 - Ilustração da *Solution* no *Visual Studio 2015*

5. 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.

5.1. Plano

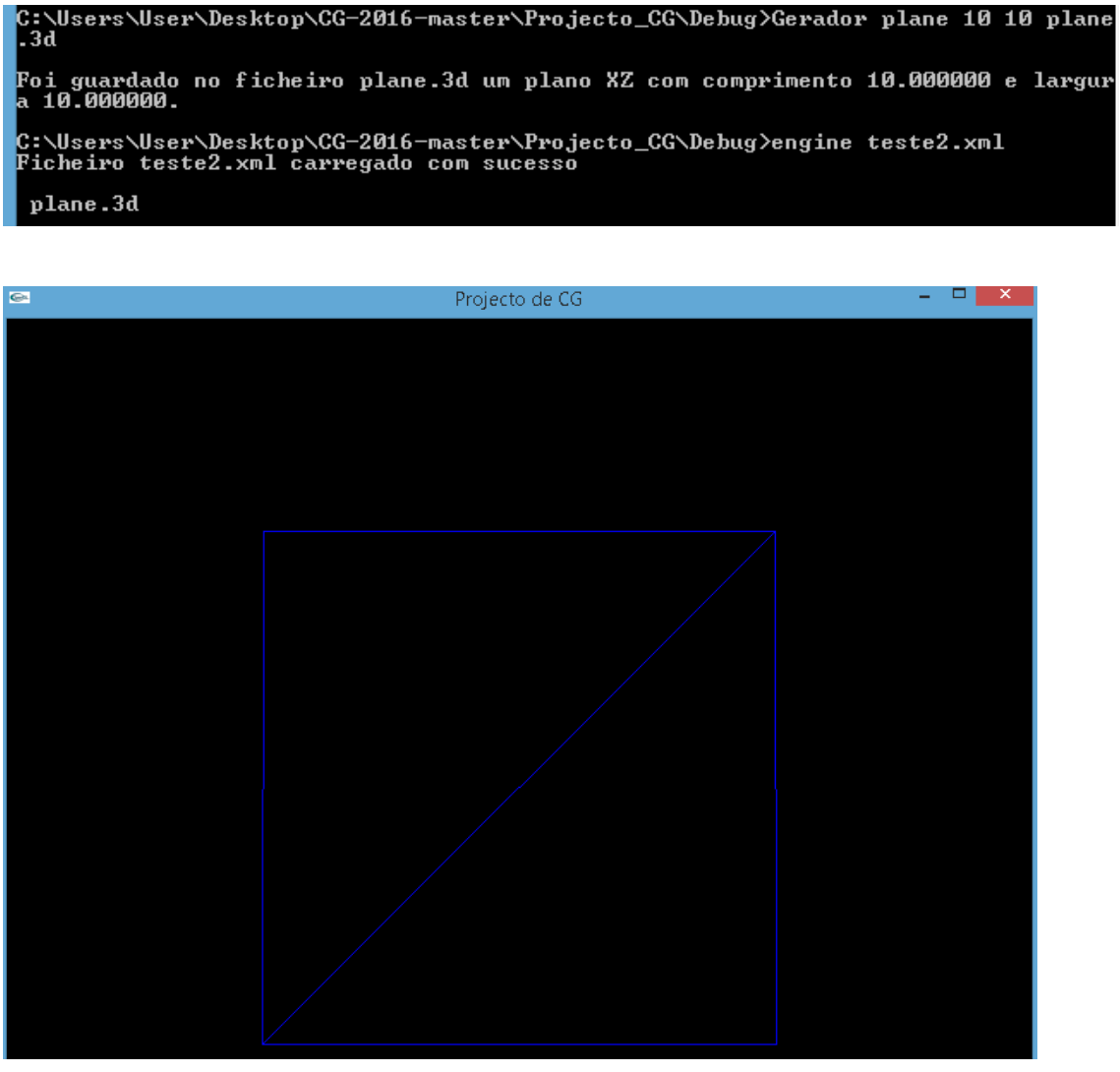


Figura 7 - Plano

5.2. Caixa

```
C:\Users\User\Desktop\CG-2016-master\Projecto_CG\Debug>Gerador box 5 5 5 box.3d

Foi guardado no ficheiro box.3d uma caixa com comprimento 5.000000, largura 5.00
0000 e altura 5.000000.

C:\Users\User\Desktop\CG-2016-master\Projecto_CG\Debug>Engine teste.xml
Ficheiro teste.xml carregado com sucesso

 box.3d
```

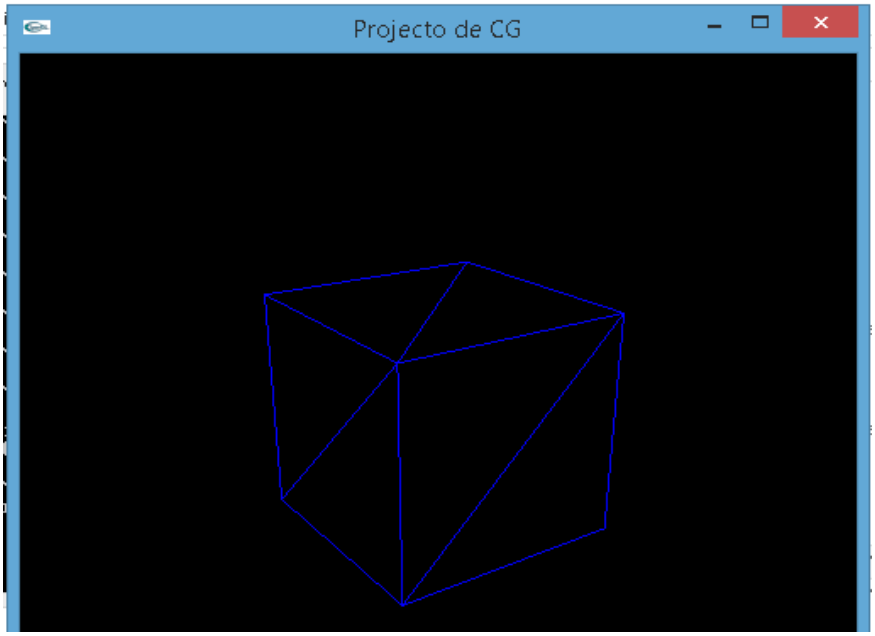


Figura 8 - Caixa

5.3. Esfera

```
C:\Users\User\Desktop\CG-2016-master\Projecto_CG\Debug>Gerador sphere 5 30 30 sp
here.3d

Foi guardado no ficheiro sphere.3d uma esfera com raio 5.000000, 30.000000 fatia
s e  30.000000 camadas.

C:\Users\User\Desktop\CG-2016-master\Projecto_CG\Debug>Engine teste.xml
Ficheiro teste.xml carregado com sucesso

 sphere.3d
```

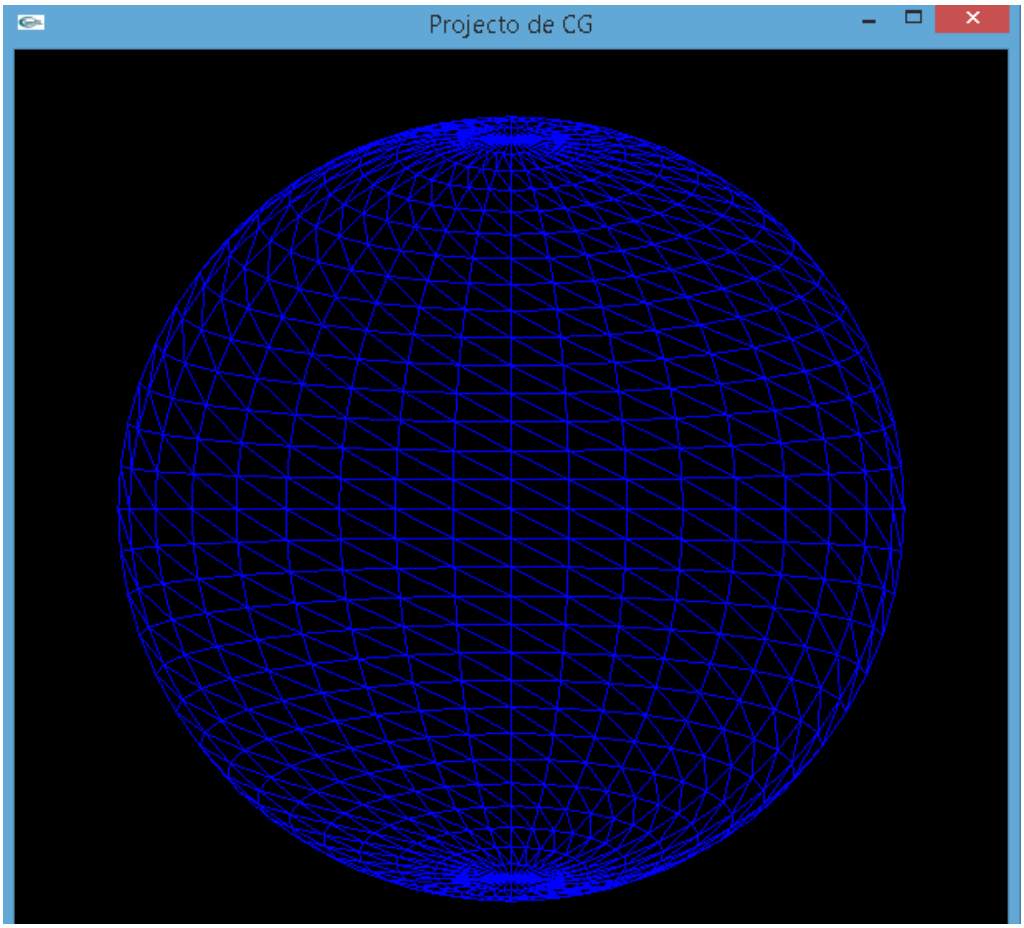
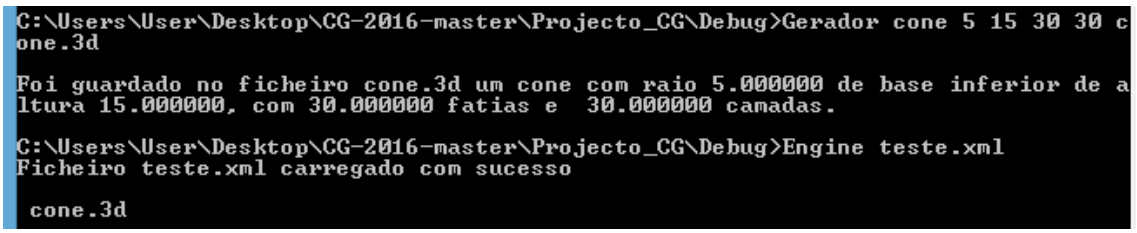


Figura 9 - Esfera

5.4. Cone



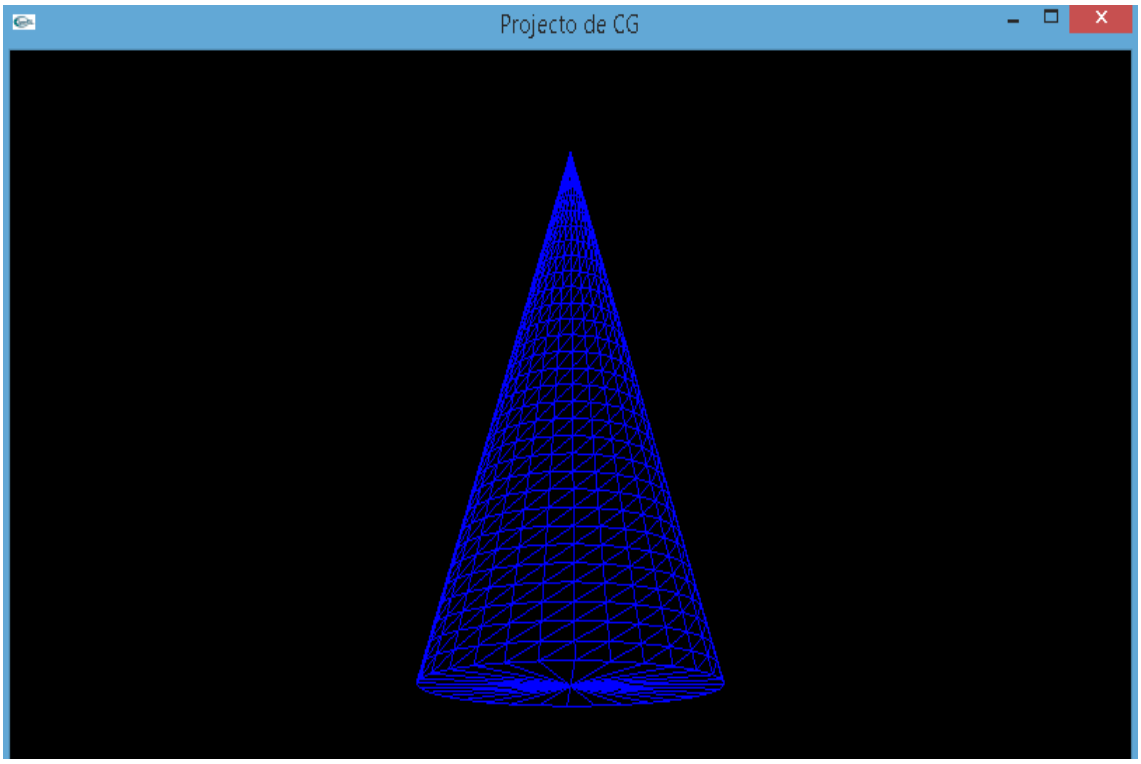


Figura 10 - Cone

5.4.1 XML com quatro figuras

Ficheiro: *teste2.xml*

Código em XML:

```
<Scene>
 <model file="sphere.3d"/>
 <model file="plane.3d"/>
 <model file="box.3d"/>
 <model file="cone.3d"/>
</Scene>
```

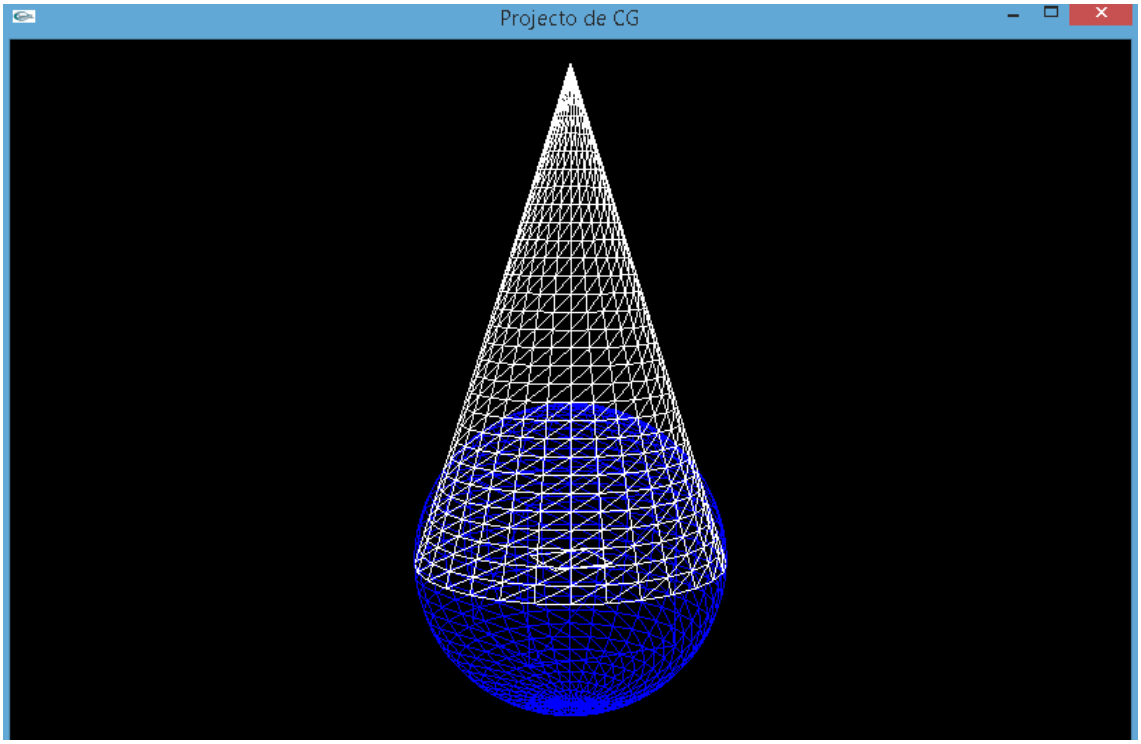


Figura 11 - Quatro figuras

6. Conclusão

Com a entrega desta primeira fase, podemos afirmar que responde de uma forma geral aos requisitos propostos pela equipa docente da disciplina; uma aplicação que cria grande parte dos modelos mais básicos, e, outra responsável pelo processamento de ficheiros *XML* com dados de cenas a desenhar.

Sentimos que este trabalho alargou a nossa familiarização perante interface gráfica e melhorou a nossa percepção de como as primitivas geométricas são desenhadas, sendo possível desenhar qualquer objecto a partir de triângulos.