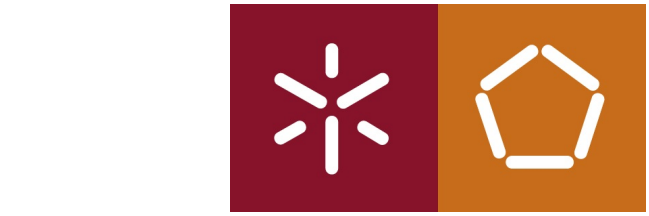
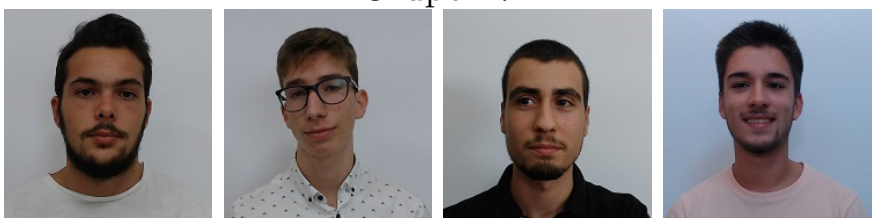


Universidade do Minho
Mestrado Integrado em Engenharia Informática
3ºano - 2º Semestre

Computação Gráfica

Relatório sobre a Fase 1

Grupo 47



a83732 – Gonçalo Rodrigues Pinto
a84197 – João Pedro Araújo Parente
a84829 – José Nuno Martins da Costa
a85059 – Diogo Paulo Lopes de Vasconcelos

6 de Março de 2020

Conteúdo

Lista de Figuras

1 Introdução

No 2º semestre do 3º ano do Curso de Engenharia Informática da Universidade do Minho, existe uma unidade curricular denominada por Computação Gráfica, que tem como objectivo ajudar os estudantes caracterizar as transformações geométricas e os referencias utilizados na computação gráfica, aplicar transformações para construção de modelos geométricos complexos e posicionamento da câmara, dar a conhecer algoritmos de iluminação local e global tais como Gouraud, Phong, Ray-tracing, Radiosity and Virtual Point Lights, têm também como meta que os estudantes apliquem texturas e definam coordenadas de textura, além de permitir uma análise de soluções do ponto de vista do desempenho recorrendo a profilers, utilizando apropriadamente soluções de eliminação de geometria, recorrendo a partição espacial e por fim aplicação de análise de algoritmos para geração de sombras

O presente trabalho pretendeu desenvolver um mecanismo 3D baseado em mini gráficos de cenas e fornecendo exemplos de uso que mostrem o seu potencial.

2 Fase 1 - Primitivas Gráficas

Nesta fase, foi nos requerido a criação de duas aplicações: uma para gerar documentos com as informações do modelo (nesta fase, apenas gerou-se os vértices do modelo), designada por ***generator***, e o próprio mecanismo que lê um documento de configuração, escrito em XML, e que exiba os modelos, designada por ***engine***.

2.1 Abordagem

A comunicação entre as duas aplicações, como foi solicitada, esta realizou-se a partir de um ficheiro que a aplicação *generator* gera e posteriormente a aplicação *engine* efectua a leitura, esta segunda aplicação como foi referido acima é quem realiza o desenho dos sólidos. Para atingir este objectivo é necessário possuir o conhecimento acerca de duas informações os pontos que tem de desenhar e como eles estão unidos. Uma abordagem mais ingénua/directa/fácil, a nossa abordagem inicial, foi escrever os pontos aos triplos, o que significava que estes estavam todos ligados, por exemplo: "ABC CDB"(os pontos no exemplo apresentado encontram-se como letras numa tentativa de simplificar na realidade, é claro que cada ponto possui coordenadas em x, y e z), corresponde à existência dos pontos A,B,C,D onde os pontos A,B,C encontram-se interligados através de um triângulo, ou seja, as ligações do primeiro triplo são algo do género:

$$\begin{bmatrix} A & B \end{bmatrix}$$

$$\begin{bmatrix} A \\ C \end{bmatrix}$$

e haveria a outra representação C,D,B através de outro triângulo;

Com este pequeno exemplo, conseguimos identificar algumas falhas nesta abordagem inicial tal como o facto de existir pontos, como é o caso dos pontos B e C, que fazem parte de múltiplos triângulos e por este motivo escreveu-se no ficheiro as suas coordenadas várias vezes, o que levou pensar numa nova abordagem, que evitasse a existência de informação repetida.

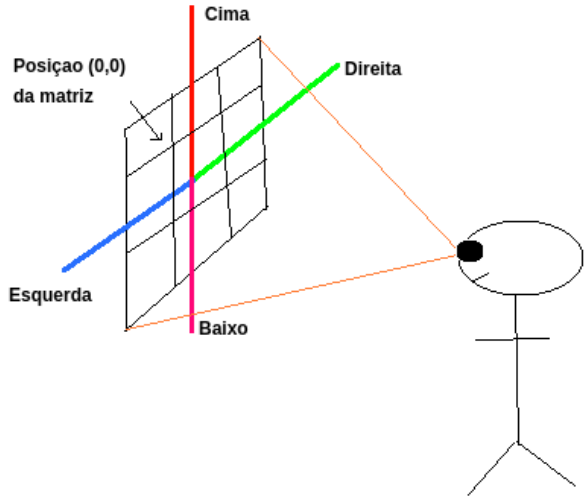
Considerando que todos os modelos que foram pedidos partilham de uma propriedade que é a seguinte, se identificarmos num ponto em qualquer lado do modelo, esse ponto apenas têm ligação com outro ponto à sua esquerda ou à sua direita ou abaixo de si ou até com outro ponto que se encontra-se acima, o que levou à reflexão de quais as estruturas que representam este conceito, o que levou à conclusão de representar os modelos como matrizes dado que implementam este conceito.

Utilizado o exemplo acima representado numa matriz ficaria da seguinte forma:

$$Matriz = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

Interpretando a matriz apresentada consegue-se conhecer que o ponto A possui apenas duas ligações, à sua direita com B e abaixo de si com C; o ponto B por sua vez tem também duas ligações à sua esquerda com A e abaixo de si com D; etc.

Esta matriz pode simplesmente representar o modelo mais simples, neste presente trabalho é o plano, considerando que as coordenadas de todos os pontos respeitem as regras necessárias para constituir um plano, até ao modelo mais abstracto necessitam apenas de uma matriz com tamanho superior dado a existência de mais pontos na sua representação.



É de realçar que alguns termos referidos acima como o caso de esquerda, de direita, de baixo e de cima são termos que dependem exclusivamente da perspectiva, por convenção decidiu-se que quando é referido, por exemplo, o termo de direita é relativamente à perspectiva do observador, isto é, para o lado do triângulo que é desenhado (nada relacionada com a posição dos eixos) para na sua representação fiquem orientados para a posição pretendida.

2.2 Generator

Para criar os documentos do modelo, a aplicação (independente do mecanismo) recebe como parâmetros o tipo de primitiva gráfica, outros parâmetros necessários para a criação do modelo e o arquivo de destino onde os vértices serão armazenados. O formato do documento como foi deixado à consideração dos alunos, o nosso grupo decidiu guardar a informação dos documentos em bytes de forma a reduzir o espaço ocupado por dados num determinado ficheiro e ao mesmo tempo obter mais desempenho na leitura e escrita dos dados nos ficheiros.

Desta forma criou-se uma estrutura de dados designada de *ponto* que possui nos seus campos três floats (x,y,z).

Consequentemente, seguindo a abordagem acima apresentada em todas as primitivas gráficas que se construiu preencheu-se uma posição da matriz com a estrutura de dados acima referida, *ponto*.

Por fim, criou-se uma função *escreveFicheiro* que recebe como parâmetros o número de linhas, o número de colunas, a matriz de pontos e o nome do ficheiro e assim abre o ficheiro designado em modo "append", ou seja, abre o documento para saída no final de um documento. As operações de saída gravam sempre os dados no final do ficheiro, expandindo-o. As operações de reposicionamento são ignoradas. O ficheiro é criado se não existir. Caso a abertura do ficheiro ocorra com sucesso escreve o número de linhas e colunas no ficheiro de modo a facilitar a possível leitura e consequentemente escreve a matriz que recebeu no ficheiro, ponto a ponto. No fim, o ficheiro é fechado.

2.2.1 Plane

Nesta primeira primitiva gráfica é descrita sendo um quadrado no plano XZ, centrado na origem, feito com 4 pontos diferentes, tendo como parâmetro o comprimento de lado (lado).

Para calcular estes pontos foi apenas necessário realizar as seguintes operações:

$$d = lado/2$$

todos os pontos possuem $y = 0$, dado que o plano encontra-se no plano XZ.

Ponto A
xa = -d;
za = -d;

Ponto C
xa = -d;
za = d;

Ponto B
xa = d;
za = -d;

Ponto D
xa = d;
za = d;

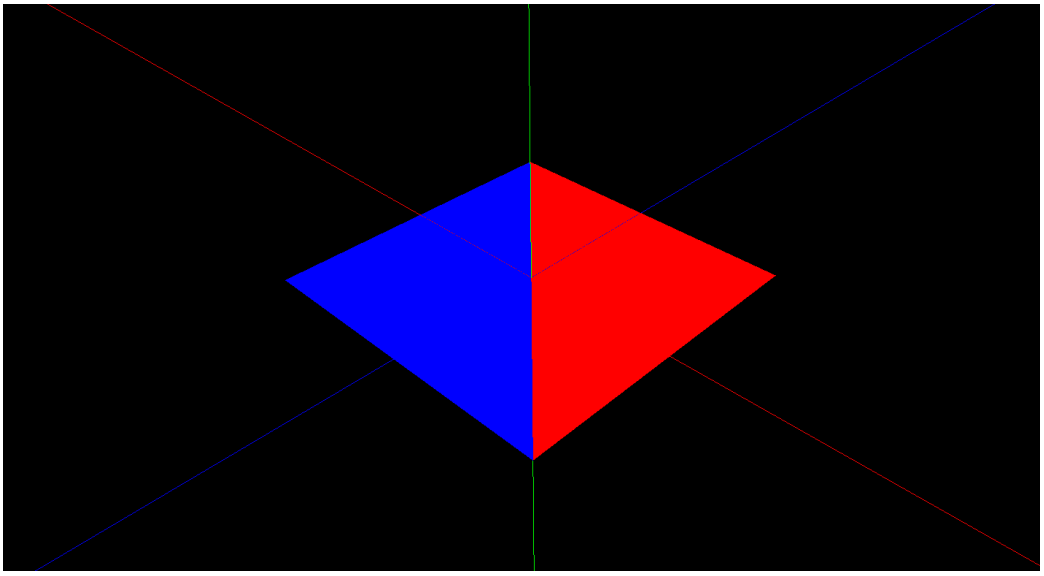


Figura 1: Representação do plano com lado 20.

2.2.2 Box

Nesta primitiva gráfica requer dimensões X, Y e Z e, opcionalmente, o número de divisões. Uma caixa consiste numa prisma de seis faces, onde teve-se em consideração a existência de mais uma dimensão, sendo as três que a constituem o comprimento(X), largura (Z) e altura(Y). O desenvolvimento das faces da caixa seguiu a mesma teoria do plano, no entanto, como foram implementadas mais camadas, precisou-se de um algoritmo mais complexo.

Consequentemente considerou-se o número de camadas logo tornou-se então necessário conhecer o espaçamento entre cada divisão. Os espaçamentos de uma divisão são calculados da seguinte forma:

espacoC = (float) comprimento / (divisoes + 1);
espacoL = (float) largura / (divisoes + 1);
espacoA = (float) altura / (divisoes + 1);

De modo a que a caixa fique centrada na origem, as coordenadas x,y,z do seu centro foram calculadas através das seguintes expressões.

x = (float) comprimento / 2;
z = (float) largura / 2;
y = (float) altura / 2;

O algoritmo para o cálculo de todos os vértices que constituem uma caixa foi dividido em três fases: cálculo das faces XY, cálculo das faces YZ e por fim, o cálculo das faces XZ. Todas estas três faces seguem o mesmo conjunto de procedimentos.

Considerando a face voltada para a frente, onde o valor de z é constante em todos os pontos, neste caso as faces XY a inserção dos pontos fez-se da esquerda para a direita, de baixo para cima, sendo as coordenadas dos vértices calculadas pelos espaçamento do comprimento, chegando ao fim da linha incrementou-se a altura e de seguida calculou-se o espaçamento da altura. Assim, foi possível desenvolver uma fórmula para a aplicação do algoritmo nas restantes faces.

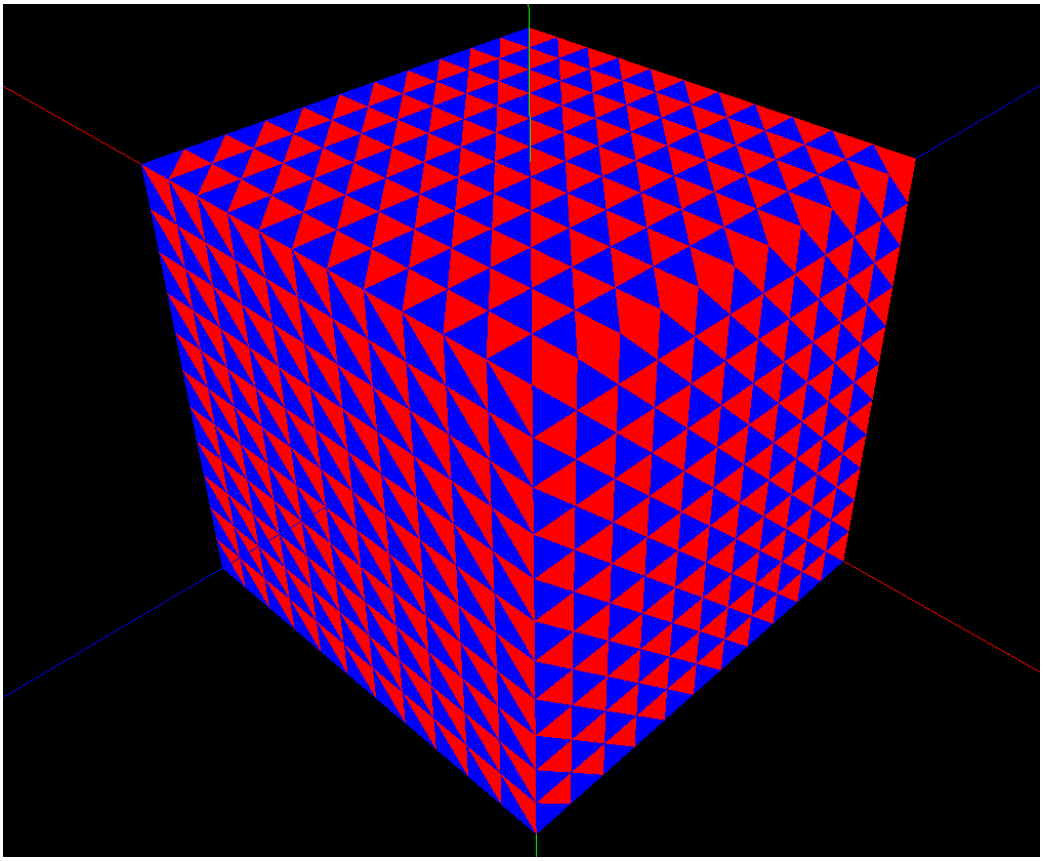


Figura 2: Representação de uma caixa com 20 de largura, profundidade, altura e 10 divisões.

2.2.3 Sphere

Nesta primitiva gráfica é necessário o raio, o número de fatias e pilhas.

Para calcularmos as coordenadas de todos os pontos primeiro efectuou-se o cálculo da altura de cada camada, para tal bastou calcular o y máximo da esfera, que é um ponto com $y = 2 * raio$, e subtrair-lhe a largura entre camadas que é igual a $(2 * raio)/pilhas$ este y é a altura da primeira camada para as próximas camadas bastou subtrair a largura entre camadas a esta primeira camada.

Nesta mesma camada a altura (coordenada y) dos pontos é sempre igual variando apenas as coordenadas x e z. Aplicando a fórmula do circulo, $x^2 + z^2 = raio^2$.

Sabe-se que o raio de cada camada, começando por exemplo, com x=0 conseguiu-se conhecer a coordenada z e como tal é possível saber o x e z e desta forma conhecemos todas as coordenadas, posteriormente aplicando

trigonometria, obtivemos todos os pontos dessa camada, efectuando $alfa+= alfa$, por exemplo:

Ponto2 = $(r*cos(alfa+angulo), alturadacamada, sqrt(raio-x^2))$, sendo $alfa = 2*pi/numerodefatias$.

Por fim, a única propriedade que ainda não foi referida foi o raio de cada camada, recorrendo à fórmula dos pontos da esfera: $x^2 + y^2 + z^2 = RaioDaEsfera^2$, tendo em conta que para cada camada consegue-se saber a altura, como o raio da esfera é fornecido pelo utilizador, quando coloca-se x=0 o z toma o valor do raio dessa camada.

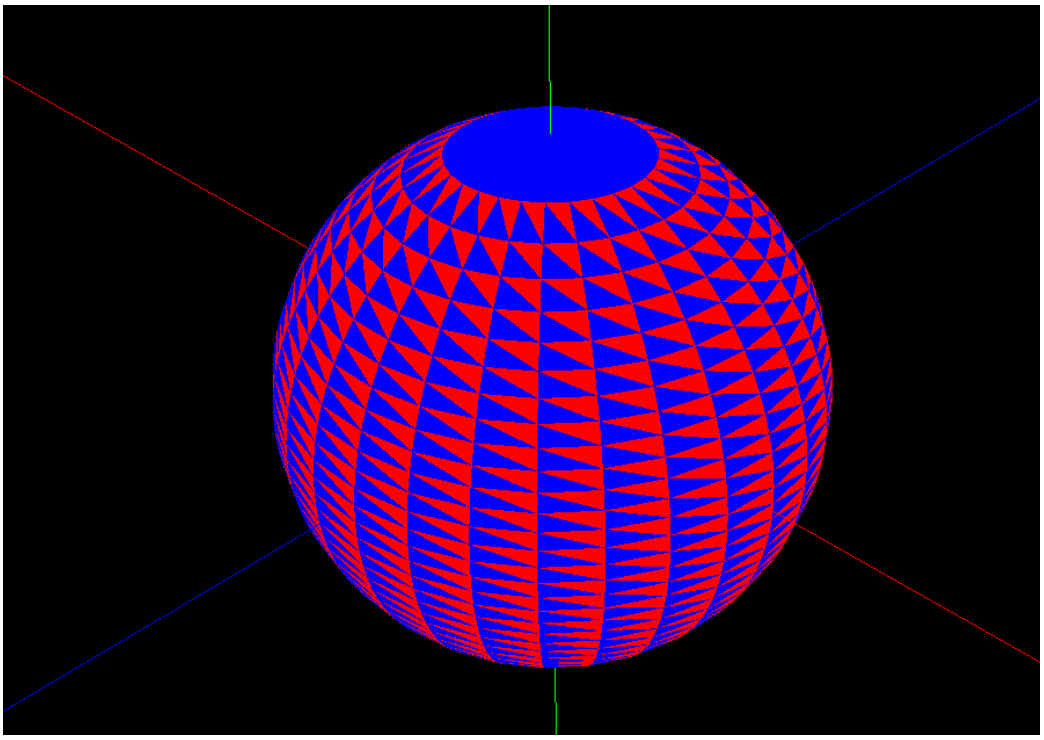


Figura 3: Representação de uma esfera com 10 de raio, 30 fatias e pilhas.

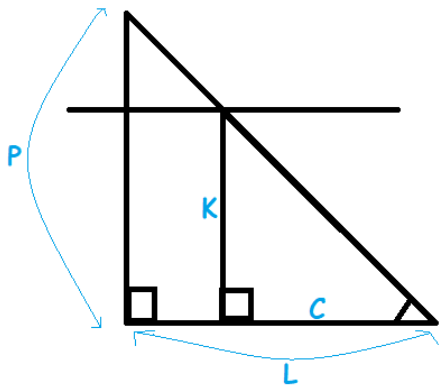
2.2.4 Cone

Nesta última primitiva gráfica requer raio inferior, altura, fatias e pilhas. Tal como no caso da esfera, as fatias e pilhas são, respectivamente, camadas na vertical e na horizontal ao longo da superfície do cone. Quanto maior forem em número, maior será o número de pontos e maior será a precisão no desenho da primitiva.

Tal como foi idealizado na construção dos pontos da esfera começou-se por calcular de igual modo o ponto de maior coordenada y, calculando depois

a largura entre cada camada, subtraindo a coordenada y desse ponto obteve-se a coordenada y da primeira camada repetindo este processo obtivemos a coordenada y de cada camada.

Na imagem abaixo apresentamos um cone visto de frente "assente no chão", ou seja, é um triângulo equilátero, dividindo este triângulo na vertical a meio, obtemos dois triângulos rectângulos. Por conseguinte, se efectuar um corte de um plano paralelo "ao chão"ao cone na horizontal (o leva a formação de uma camada), considerando o triângulo rectângulo da direita colocando dentro deste outro triângulo rectângulo como está representado na figura, aplica-se o conceito de semelhança de triângulos dado que estes dois triângulos possuem dois ângulos iguais, o que leva a determinação do raio de cada camada.



O raio de cada camada é dado por $= L - C = L - LK/P$, sendo que L o raio da base do cone, P a altura do cone e K a altura do triângulo pequeno.

Ficando a conhecer o raio de cada camada usando a fórmula do circulo tal como no modelo da esfera obteve-se os pontos de cada camada e consequentemente de todo o cone.

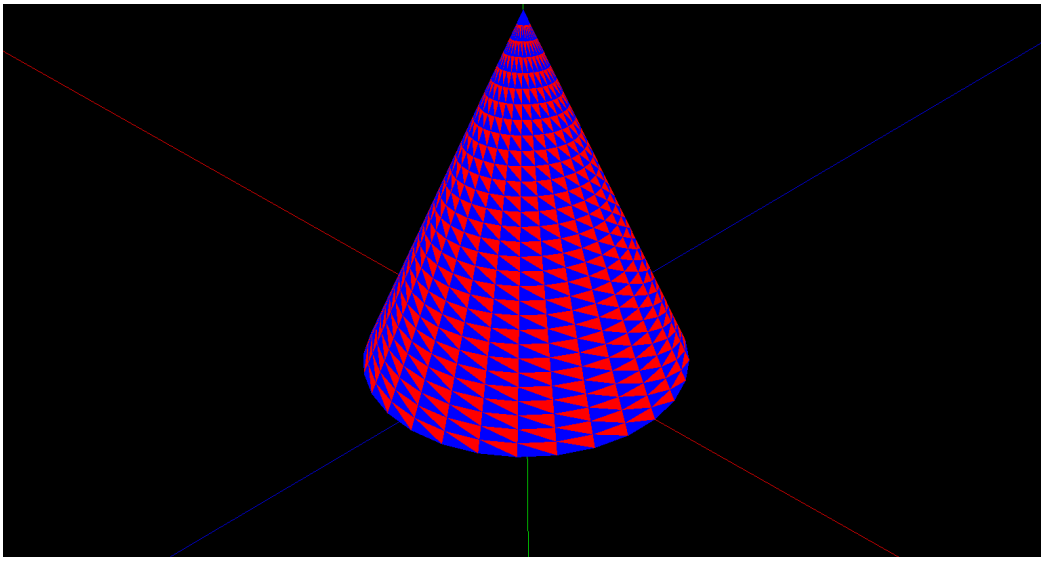


Figura 4: Representação de um cone com 10 de raio, 20 de altura, 30 fatias e pilhas.

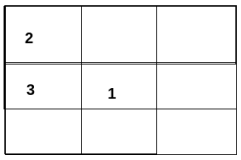
2.3 Engine

Nesta segunda aplicação recebe um documento de configuração, escrito em XML. Nesta fase, o arquivo XML contêm apenas a indicação dos arquivos gerados anteriormente para serem carregados. O arquivo XML é lido apenas uma vez quando a aplicação inicia.

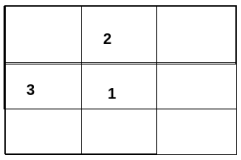
O engine partir do XML que efectua a leitura, fica a conhecer todos os documentos com extensão .3d que tem de representar, este lê esses ficheiros que conhece efectuando a passagem de cada um para uma matriz igual a que a aplicação *generator* criou para escrever os pontos que gerou e posteriormente *engine* efectua a representação no OpenGL percorrendo essa matriz pela ordem correta.

Um pormenor muito importante e particular é que esta matriz tem que ser totalmente percorrida, para que os triângulos sejam todos desenhados, tendo em conta a regra da mão direita percorre-se a matriz em todas as linhas-1 e colunas-1, considerando a variável i, o iterador das linhas e a variável j, o iterador das colunas, sendo que em cada iteração representou-se dois triângulos.

No primeiro triângulo começou-se por desenhar o ponto da linha=i+1 e da coluna=j+1, de seguida o ponto da linha=i e da coluna=j e por último linha=i+1, coluna=j.



No segundo triângulo começou-se por representar o ponto da linha=i+1 e da coluna=j+1, a seguir o ponto da linha linha=i e da coluna=j+1 e por fim o ponto da linha=i+1 e da coluna=j.



Desta forma não desenhou-se nenhum triângulo para "dentro"do sólido.

```
<scene>
     <model file="sphere.3d" />
     <model file="plane.3d" />
</scene>
```

Figura 5: Ficheiro de configuração XML.

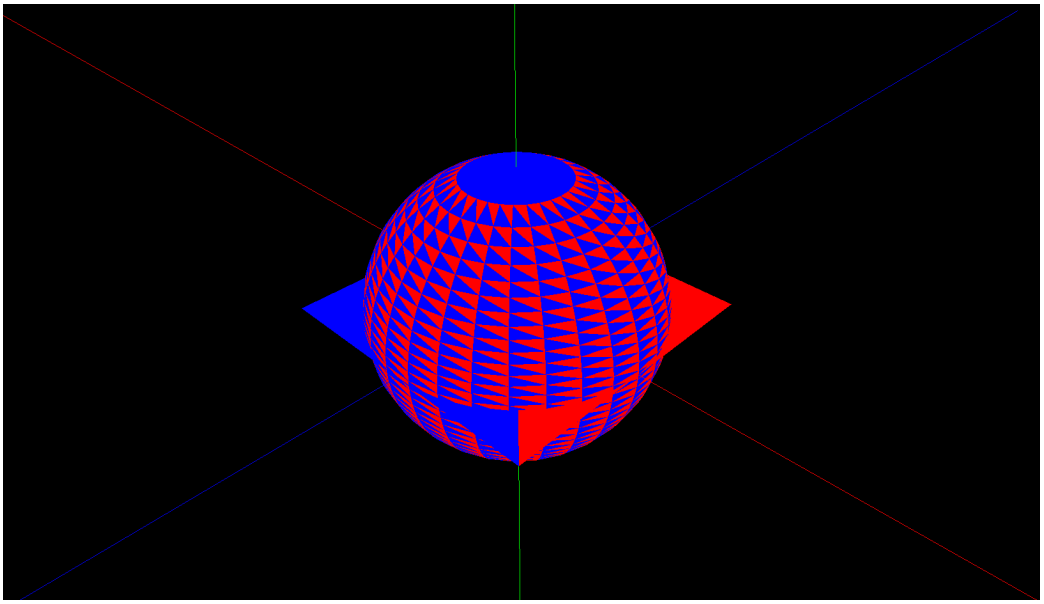


Figura 6: Output da aplicação engine mediante o ficheiro de configuração XML apresentado.

3 Conclusão

O presente relatório descreveu, de forma sucinta, a resolução desta fase mediante os requisitos apresentados.

Consideramos que os principais objectivos foram cumpridos.

Sentimos que a realização deste projecto consolidou os nossos conhecimentos em ferramentas associadas à computação gráfica como o OpenGL e o GLUT, permitiu adquirir conhecimentos da linguagem C++, essencial para o desenvolvimento desta fase e conseguimos obter uma noção dos algoritmos que estão associados à criação de algumas primitivas gráficas.

Em suma, esperamos que o trabalho realizado até ao momento seja essencial e fulcral para as fases seguintes do projecto.