Universidade do Minho
Escola de Engenharia

Licenciatura em Engenharia Informática 2024/25

Computação Gráfica:
Relatório de Projeto (1ª Fase)

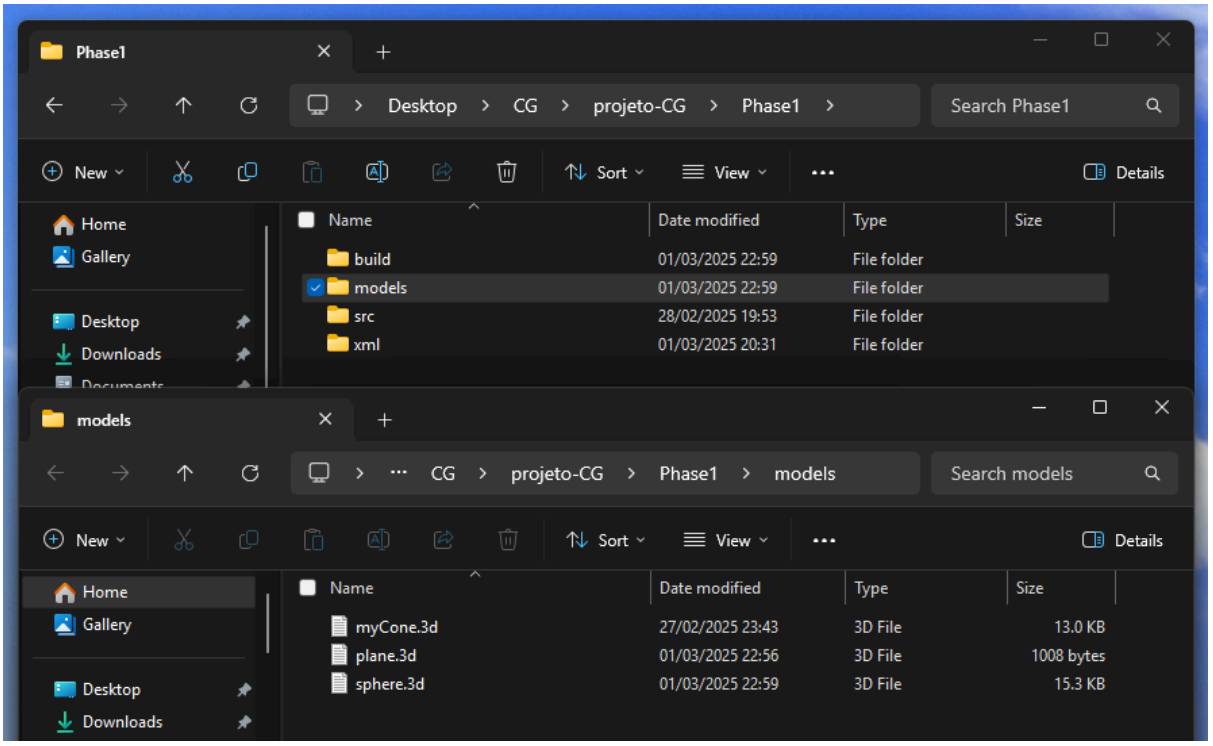
GRUPO 8

Gustavo Barros	Pedro Ferreira	Enzo Vieira	Luís Figueiredo
A100656	A97646	A98352	A100549

Gerador

Foi solicitada a escrita dum gerador de vértices à base de comandos com sintaxe predefinida, que efetuasse o cálculo e armazenamento em ficheiro de extensão “.3d” das coordenadas cartesianas dos vértices do modelo pedido, de modo a que sejam desenháveis por um motor de renderização, triângulo a triângulo, sem conhecimento de quaisquer outros detalhes.

Os ficheiros resultantes do gerador são criados na diretoria “models” estabelecida ao lado da diretoria “build”, sendo nesta última onde o programa é compilado.



Como cada modelo tem a sua estratégia específica para cálculo de vértices (lembrar que foi solicitado suporte para geração de planos quadrados, cubos, esferas e cones), foram definidas as classes ***SquarePlane***, ***SquareBox***, ***Sphere*** e ***Cone***, cada com a sua versão característica do método ***generateVertices()***.

Nos capítulos seguintes serão explicados, modelo a modelo, os raciocínios tomados para este cálculo das coordenadas.

Plano

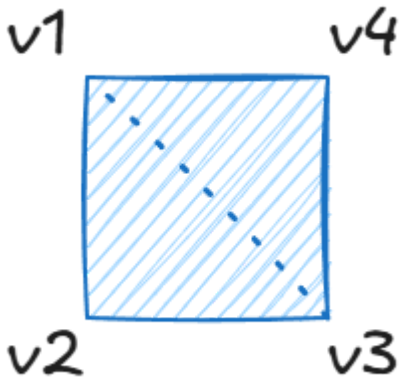
Os vértices dum plano quadrado centrado na origem com certo número de divisões podem ser calculados considerando o seguinte:

- ***length*** é o comprimento total do lado do plano.
- ***divisions*** é o número de divisões.
- ***offset = length/2*** é a ordenada na origem necessária para centrar o plano, senão seria desenhado partindo do canto (0,0).
- ***divLength = length/divisions*** é o comprimento duma só divisão.
- ***iRow*** e ***iColumn*** são os índices de certa linha e coluna na grelha formada pelas divisões do plano.

As coordenadas dos vértices podem ser definidas do seguinte modo, partindo do canto superior esquerdo:

$$\begin{pmatrix} x_{iColumn} \\ y \\ z_{iRow} \end{pmatrix} = \begin{pmatrix} -\text{offset} + iColumn \cdot \text{divLength} \\ \text{height} \\ -\text{offset} + iRow \cdot \text{divLength} \end{pmatrix}, \quad iColumn = 0, 1, \ldots, \text{divisions} - 1, \quad iRow = 0, 1, \ldots, \text{divisions} - 1$$

E assim, cada quadrado ou célula pode ser definida por quatro vértices obtidos pelos índices de linha/coluna atuais (***iRow***, ***iColumn***) e seguintes (***iRow+1***, ***iColumn+1***):



v1 = (x_(iColumn), h, z_(iRow))
v2 = (x_(iColumn), h, z_(iRow+1))
v3 = (x_(iColumn+1), h, z_(iRow+1))
v4 = (x_(iColumn+1), h, z_(iRow))

De seguida, declara-se a célula para desenho segundo dois triângulos em ordem antihorária:

Primeiro: ***v1*** -> ***v2*** -> ***v3***
Segundo: ***v3*** -> ***v4*** -> ***v1***

Caixa

A declaração dos vértices da caixa segue as mesmas fórmulas do plano, mas com alguns detalhes que vão mudando consoante a face abordada:

- Os triângulos são definidos em ordem horária ou antihorária, visto que as faces de trás estão naturalmente de costas para a câmara?
- Qual a coordenada que fica constante e qual constante será, visto que se declaram vértices em seis planos diferentes?

Como se quer a caixa centrada na origem, vão ser calculados vértices nos seguintes planos:

- direita e esquerda, que terão constantes ***x = offset*** ou ***-offset*** respetivamente
- cima e baixo, que terão constantes ***y = offset*** ou ***-offset*** respetivamente
- frente e trás, que terão constantes ***z = offset*** ou ***-offset*** respetivamente

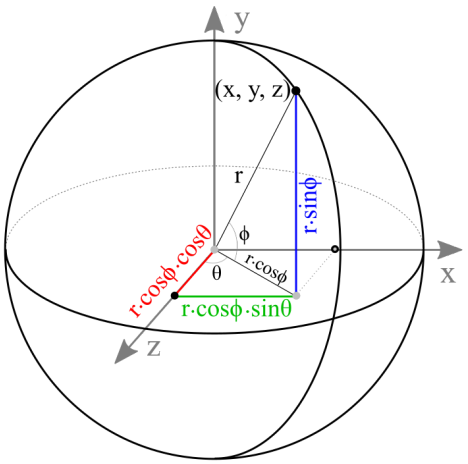
As faces direita, cima e frente estão naturalmente de frente para a câmara, logo serão desenhadas em ordem antihorária. Esquerda, baixo e trás são o oposto.

As coordenadas dos vértices são então calculadas com as fórmulas anteriores partindo do canto superior esquerdo, mas não necessariamente atribuindo os cálculos ao X e ao Z, dada a variação de plano vista antes:
À conta disso, foram designados de ***colCoord*** e ***rowCoord*** os valores calculados segundo a coluna e linha para servirem de componentes axiais diferentes segundo a face em qual se trabalha.

De Frente (Declaração Antihorária) v1 -> v2 -> v3 || v3 -> v4 -> v1	De Costas (Declaração Horária) v1 -> v3 -> v2 || v3 -> v1 -> v4
DIREITA ***v1 = (offset, rowCoord, colCoord)*** ***v2 = (offset, rowCoordNext, colCoord)*** ***v3 = (offset, rowCoordNext, colCoordNext)*** ***v4 = (offset, rowCoord, colCoordNext)***	ESQUERDA ***v1 = (-offset, rowCoord, colCoord)*** ***v2 = (-offset, rowCoordNext, colCoord)*** ***v3 = (-offset, rowCoordNext, colCoordNext)*** ***v4 = (-offset, rowCoord, colCoordNext)***
CIMA ***v1 = (rowCoord, offset, colCoord)*** ***v2 = (rowCoordNext, offset, colCoord)*** ***v3 = (rowCoordNext, offset, colCoordNext)*** ***v4 = (rowCoord, offset, colCoordNext)***	BAIXO ***v1 = (rowCoord, -offset, colCoord)*** ***v2 = (rowCoordNext, -offset, colCoord)*** ***v3 = (rowCoordNext, -offset, colCoordNext)*** ***v4 = (rowCoord, -offset, colCoordNext)***
FRENTE ***v1 = (rowCoord, colCoord, offset)*** ***v2 = (rowCoordNext, colCoord, offset)*** ***v3 = (rowCoordNext, colCoordNext, offset)*** ***v4 = (rowCoord, colCoordNext, offset)***	TRÁS ***v1 = (rowCoord, colCoord, -offset)*** ***v2 = (rowCoordNext, colCoord, -offset)*** ***v3 = (rowCoordNext, colCoordNext, -offset)*** ***v4 = (rowCoord, colCoordNext, -offset)***

Esfera

A esfera é definida por 3 atributos: ***radius*** (raio da esfera), ***slices*** (subdivisões em intervalos regulares do ângulo horizontal theta, alusivas a fatias dum bolo) e ***stacks*** (subdivisões em intervalos regulares entre valores Y, alusivas a camadas dum bolo).



A interseção duma stack com uma slice forma um quadrilátero já familiar nos anteriores plano e caixa.

Um vértice num círculo pode ser definido por coordenadas polares, sabendo:

- ***theta* θ** (ângulo correspondente à ***slice*** atual)
- ***phi* ϕ** (ângulo correspondente à ***stack*** atual)
- ***radius*** (o raio é dado adquirido)

O ϕ permite calcular Y e a projeção do vértice em XZ, enquanto que θ permite extrair dessa projeção as coordenadas X e Z propriamente ditas.

Sendo assim, um vértice terá as coordenadas cartesianas:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} r\cos(\phi)\sin(\theta) \\ r\sin(\phi) \\ r\cos(\phi)\cos(\theta) \end{pmatrix}$$

Os ângulos ϕ e **θ** variam respetivamente entre [-π/2, π/2] e [0, 2π] segundo incrementos deduzidos pela quantidade de stacks e slices.

Variações:

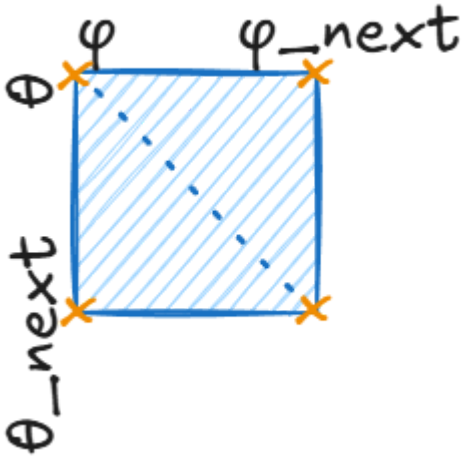
- ϕ varia entre [-π/2, π/2] segundo o número de stacks. Assim, cresce em incrementos de ***phiStep = π/stacks***. Há portanto uma iteração pelos índices ***iStack*** em que, para cada, se obtém:

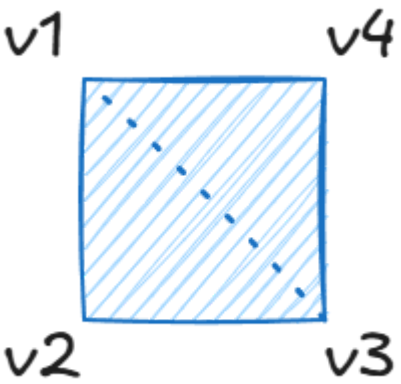
```
phi = M_PI / 2 - iStack * phiStep;
phiNext = M_PI / 2 - (iStack + 1) * phiStep;
```

- **θ** varia entre [0, 2π] segundo o número de slices. Assim, cresce em incrementos de ***thetaStep = π/slices***. Há portanto uma iteração pelos índices ***iSlice*** em que, para cada, se obtém:

```
theta = iSlice * thetaStep;
thetaNext = (iSlice + 1) * thetaStep;
```

Com isto em mente, o quadrilátero resultante da interseção duma stack e duma slice pode ser definido por:



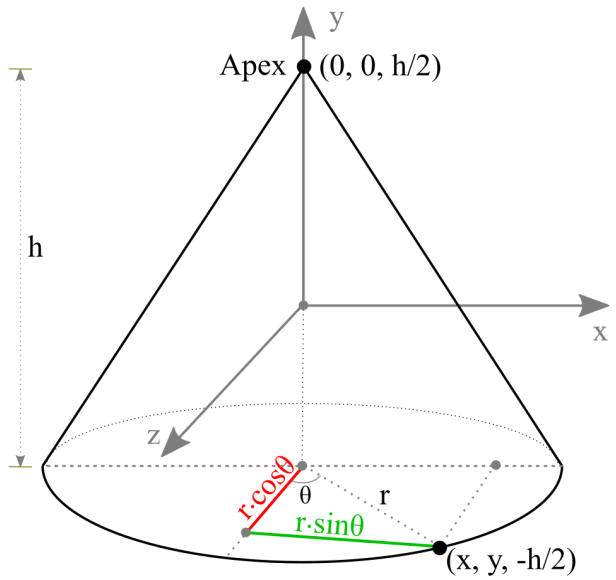


v1 = (***r*cos(ϕ)*sin(θ),*** ***r*sin(ϕ),*** ***r*cos(ϕ)*cos(θ)*** ***)***	***v4 = (*** ***r*cos(ϕ)*sin(θnext),*** ***r*sin(ϕ),*** ***r*cos(ϕ)*cos(θnext)*** ***)***
v2 = (***r*cos(ϕnext)*sin(θ),*** ***r*sin(ϕnext),*** ***r*cos(ϕnext)*cos(θ)*** ***)***	***v3 = (*** ***r*cos(ϕnext)*sin(θnext),*** ***r*sin(ϕnext),*** ***r*cos(ϕnext)*cos(θnext)*** ***)***

Cone

O cone é definido por 3 atributos: ***radius*** (raio da base), ***slices*** (subdivisões em intervalos regulares do ângulo horizontal theta, alusivas a fatias dum bolo) e ***stacks*** (subdivisões em intervalos regulares entre valores Y, alusivas a camadas dum bolo).

A obtenção do cone consiste primeiro na definição da base em ordem horária (por estar naturalmente de costas para a câmara) e depois das faces laterais. Seguiu-se a lógica mostrada na imagem tirando o facto da base ficar definida no plano XZ ao invés de centrar em altura.



A base nada mais é que a definição dum círculo. Itera-se pela quantidade de slices, incrementando theta segundo um coeficiente obtido pelo mesmo raciocínio da esfera: ***thetaStep = π/slices***.

Os lados são desenhados iterativamente por cada célula quadrangular formada pela interseção duma stack com uma slice. A partir duma stack, podem ser calculados a altura e o raio das circunferências inferior e superior, obtendo os valores Y:

```
float yBottom = iStack * stackHeight;
float yTop = (iStack + 1) * stackHeight;
float bottomRadius = radius * (1.0f - (float)iStack / stacks);
float topRadius = radius * (1.0f - (float)(iStack + 1) / stacks);
```

Para determinada stack, itera-se por todas as slices em volta do cone. A partir duma slice, podem ser calculados os ângulos theta, obtendo os valores X e Z:

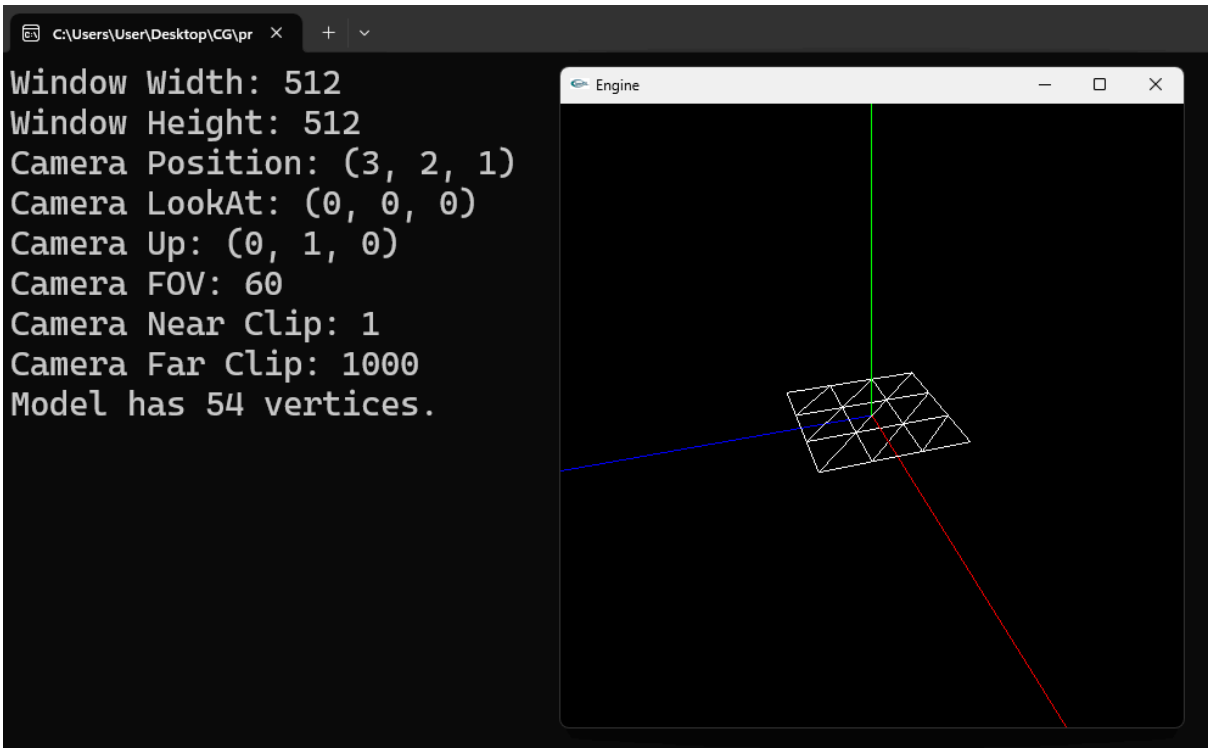
```
float theta = thetaStep * iSlice;
float thetaNext = thetaStep * (iSlice + 1);

// Bottom ring vertices
float x1 = bottomRadius * sin(theta);
float x2 = bottomRadius * sin(thetaNext);

float z1 = bottomRadius * cos(theta);
float z2 = bottomRadius * cos(thetaNext);

// Top ring vertices
float x3 = topRadius * sin(theta);
float x4 = topRadius * sin(thetaNext);

float z3 = topRadius * cos(theta);
float z4 = topRadius * cos(thetaNext);
```

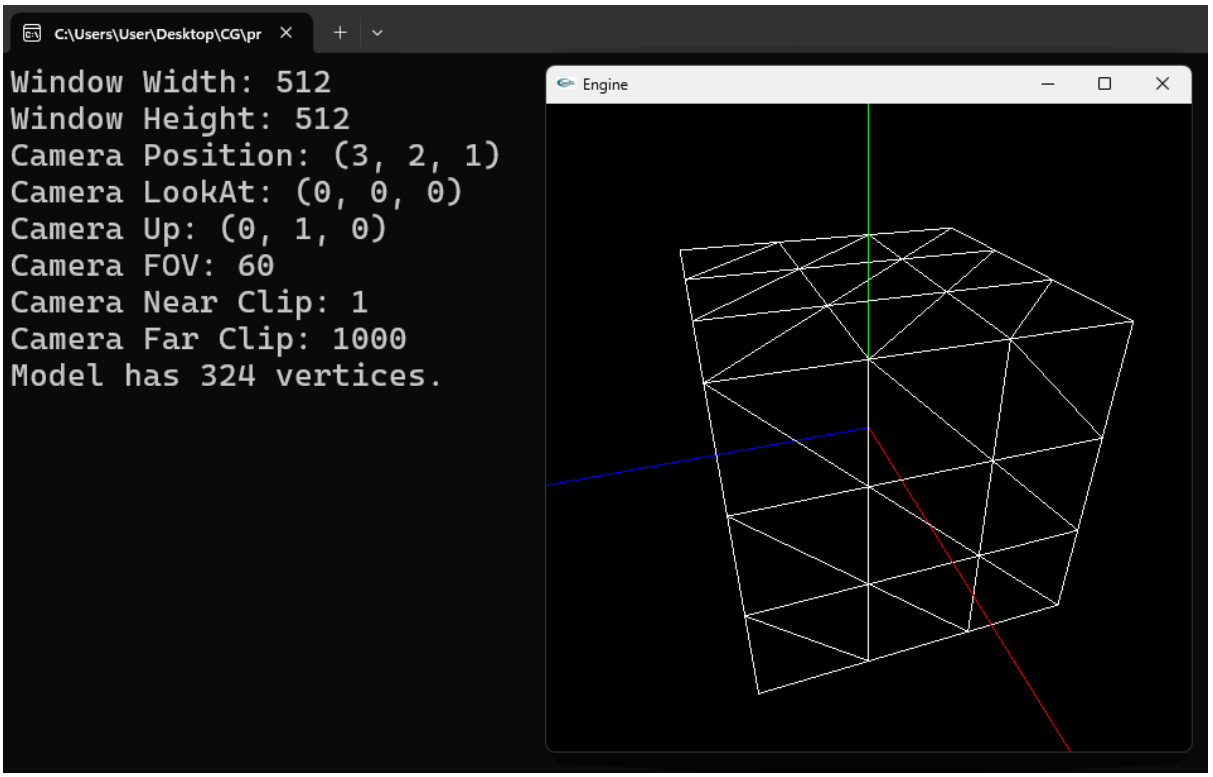
Com isto, já se consegue definir os quatro pontos necessários, tal como os triângulos:

```
// Cell vertices
Vec3 bottomLeft(x1, yBottom, z1);
Vec3 bottomRight(x2, yBottom, z2);
Vec3 topLeft(x3, yTop, z3);
Vec3 topRight(x4, yTop, z4);

// First triangle
vertices.push_back(bottomLeft);
vertices.push_back(bottomRight);
vertices.push_back(topLeft);

// Second triangle
vertices.push_back(topLeft);
vertices.push_back(bottomRight);
vertices.push_back(topRight);
```

Motor de Renderização

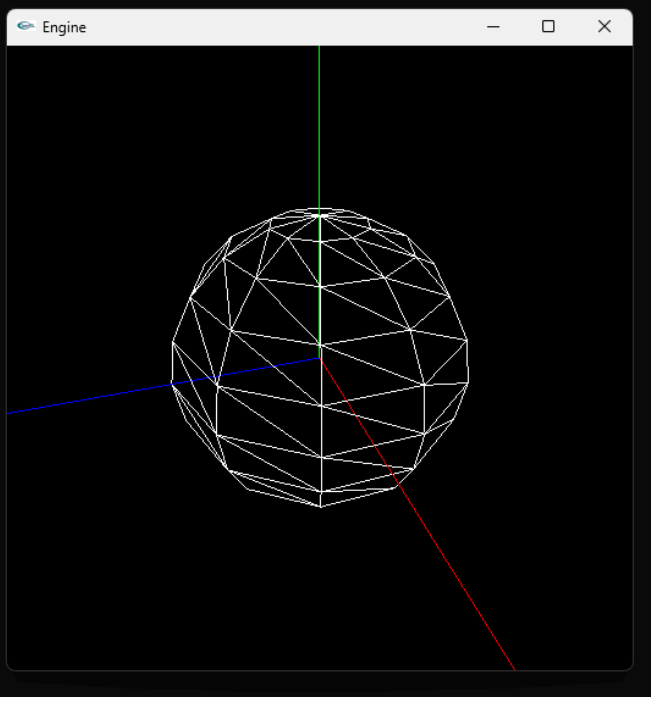
As capturas de ecrã seguintes são resultantes da leitura do ficheiro XML exemplar do enunciado (colocado na pasta “xml”) para cada um dos modelos gerados com os comandos apresentados na secção do gerador. É possível transladar o modelo com as setas do teclado, rodá-lo com Q/E ou trocar o modo de polígono com a barra de espaço.





C:\Users\User\Desktop\CG\pr

```
Window Width: 512
Window Height: 512
Camera Position: (3, 2, 1)
Camera LookAt: (0, 0, 0)
Camera Up: (0, 1, 0)
Camera FOV: 60
Camera Near Clip: 1
Camera Far Clip: 1000
Model has 540 vertices.
```



C:\Users\User\Desktop\CG\pr

```
Window Width: 512
Window Height: 512
Camera Position: (3, 2, 1)
Camera LookAt: (0, 0, 0)
Camera Up: (0, 1, 0)
Camera FOV: 60
Camera Near Clip: 1
Camera Far Clip: 1000
Model has 84 vertices.
```