

# Computação Gráfica (MIEIC)

## Trabalho Prático 3


### *Shading e aproximação de superfícies curvas*


## Objetivos


- Explorar os diferentes modos de sombreadimento
- Criar geometria composta, incluindo superfícies não-planas que explorem o sombreadimento

## Trabalho prático

Ao longo dos pontos seguintes são descritas várias tarefas a realizar. Algumas delas estão anotadas

com o ícone  (captura de imagem). Nestes pontos deverão, com o programa em execução, capturar uma imagem da execução. Devem nomear as imagens capturadas seguindo o formato "CGFImage-tp3-TtGgg-x.y.png", em que TtGgg referem-se à turma e número de grupo e x e y correspondem ao ponto e subponto correspondentes à tarefa (p.ex. "CGFImage-tp3-T3G10-2.4.png").

Nas tarefas assinaladas com o ícone  (código), devem criar um ficheiro .zip do vosso projeto, e nomeá-lo como "CGFCode-tp3-TtGgg-x.y.zip", (com TtGgg, x e y identificando a turma, grupo e a tarefa tal como descrito acima).

Quando o ícone  surgir, é esperado que executem o programa e observem os resultados. No final, devem submeter todos os ficheiros via Moodle, através do link disponibilizado para o efeito.

Devem incluir também um ficheiro **ident.txt** com a lista de elementos do grupo (nome e número). Só um elemento do grupo deverá submeter o trabalho.

## Preparação do Ambiente de Trabalho

Este trabalho deve ser baseado no trabalho anterior. Devem criar uma cópia desse trabalho, e acrescentar ao projeto o ficheiro adicional fazendo uma cópia do **MyQuad.js** (não se esqueça de adicionar **MyPrism.js** ao **includeSerial** no ficheiro **main.js**). Altere o construtor de **MyPrism** para conter as variáveis *slices* e *stacks*:

```
constructor(scene, slices, stacks)
{
    super(scene);

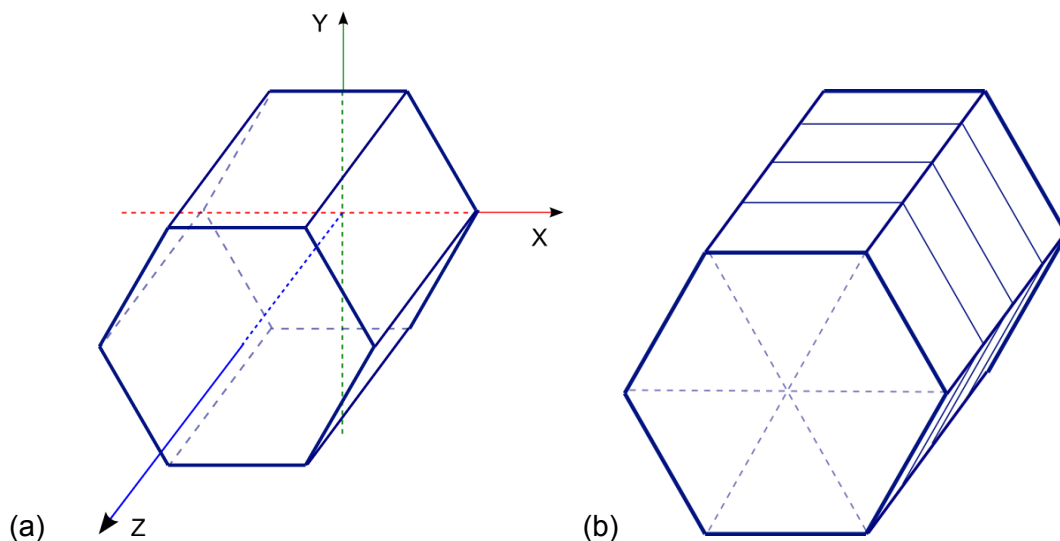
    this.slices = slices;
    this.stacks = stacks;

    this.initBuffers();
}
```

Esconda temporariamente (comente) todos os elementos da cena exceto os eixos.

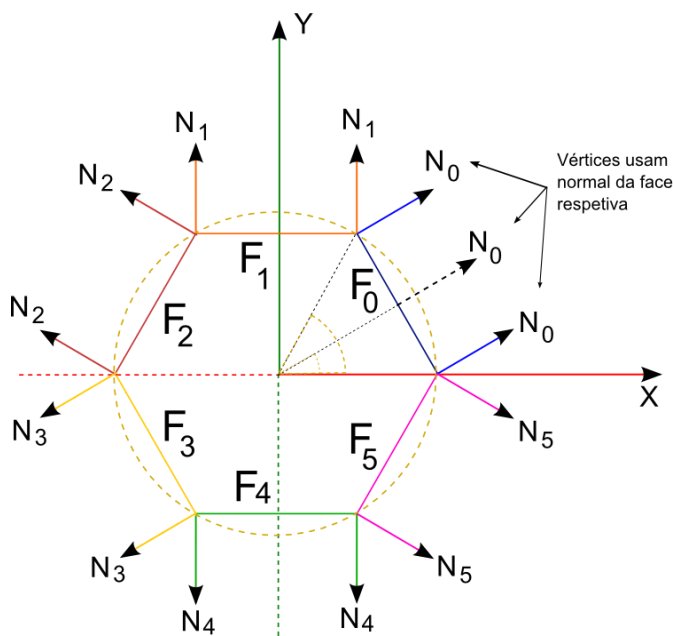
# 1. Desenho de um prisma

Pretende-se completar a classe **MyPrism** para que esta consiga desenhar um prisma com um **número variável de "lados" e de "andares"** (**slices** e **stacks**, parâmetros já incluídos no construtor da classe, ver Figura 1) como se estivesse contido (inscrito) num cilindro de raio igual a uma unidade, base coincidente com o plano XY e centrada na origem, e com comprimento também unitário, em Z. O cilindro pode ser aberto nas extremidades (sem tampas).






**Figura 1:** Prisma (a) de seis lados (**slices**) e um andar (**stack**) e (b) de seis lados e quatro andares. (a escala não é a real)

1. Numa primeira versão do prisma, considere que o cilindro tem apenas um “andar” (tal como o exemplo na Figura 1 (a)). Note que, para cada face, as normais dos seus vértices devem ser perpendiculares a essa face (Figura 2). **Poderá por isso ter de definir o mesmo vértice mais do que uma vez, na lista de vértices, de forma a atribuir-lhe normais diferentes.**



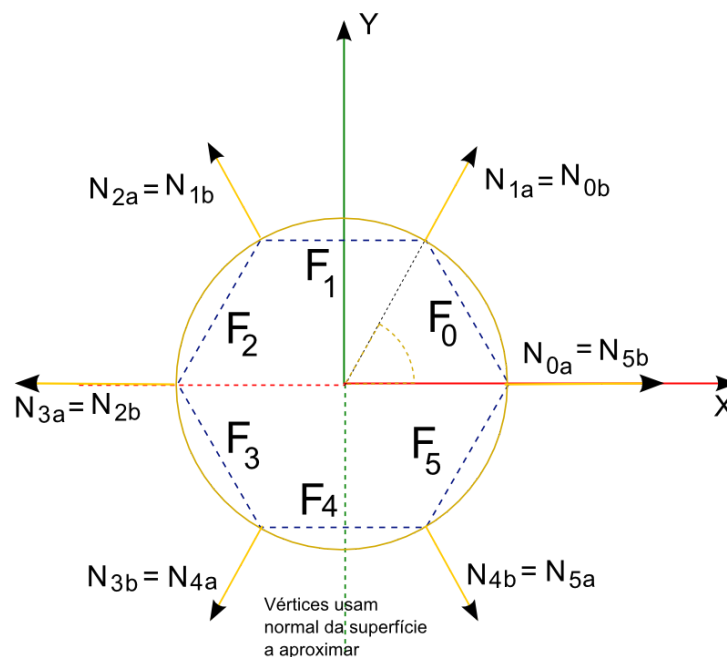
**Figura 2:** Ilustração das normais a atribuir a cada vértice no caso de seis lados. As normais serão equivalentes nos vértices de todas as **stacks**

2. Acrescente uma instância do mesmo com 8 lados ao método **init** da cena, representando uma coluna vertical com as dimensões que achar apropriadas, e desenhe-a no método **display** da cena. Poderá ter de usar rotações, escalas ou translações antes de desenhar o prisma. 
3. Em 1.1 deve ter implementado vários lados e apenas um andar. Ajuste a implementação para suportar o desenho de vários andares (**stacks**) como no exemplo na Figura 1 (b), e garanta que a coluna desenhada em 1.2 é agora desenhada com 20 andares.


(1.3  ) (1.3  )

## 2. Adaptação do prisma para simular um cilindro

1. Crie uma nova classe **MyCylinder** fazendo uma cópia do ficheiro **MyPrism.js** desenvolvido no exercício anterior. Não se esqueça de alterar o nome do novo ficheiro, da classe nele contida, e de adicionar **MyCylinder.js** ao **serialInclude** do **main.js**.
2. Altere as normais do método **initBuffers** da classe **MyCylinder**, de forma a que a normal de cada vértice seja perpendicular à superfície do cilindro perfeito em que o prisma original está inscrito. Ou seja, cada vértice que seja usado em dois lados adjacentes, terá sempre a mesma normal (Figura 3).





**Figura 3:** Ilustração das normais a atribuir a cada vértice no caso de um cilindro aproximado por seis lados.

3. Acrescente uma segunda coluna à cena, com o mesmo número de lados e andares do prisma (**8 slices, 20 stacks**), agora usando uma instância de **MyCylinder**. 
4. Simplifique a lista de vértices e de normais, de modo a eliminar duplicados (implica obviamente passar a referir o mesmo vértice mais do que uma vez na lista de índices)

(2.4  ) (2.4  )



## Extra: Criação de uma semi-esfera

Crie a classe **myLamp** que desenhe uma semi-esfera (representando um candeeiro de tecto). Utilize os mesmos princípios de "stacks" e "slices" que usou no **myCylinder**. Acrescente também as respetivas normais. Adicione-a ao centro do tecto com as dimensões apropriadas.

(extra  ) (extra  )

## Checklist

Até ao final do trabalho deverá submeter as seguintes imagens e versões do código via Moodle, respeitando estritamente a regra dos nomes, bem como o ficheiro ident.txt com a identificação dos membros do grupo:

-  Imagens (3): 1.3, 2.4, extra (nomes do tipo "CGFImage-tp3-TtGgg-x.y.png" )
-  Código em arquivo zip (3): 1.3, 2.4, extra (nomes do tipo "CGFCode-tp3-TtGgg-x.y.zip")