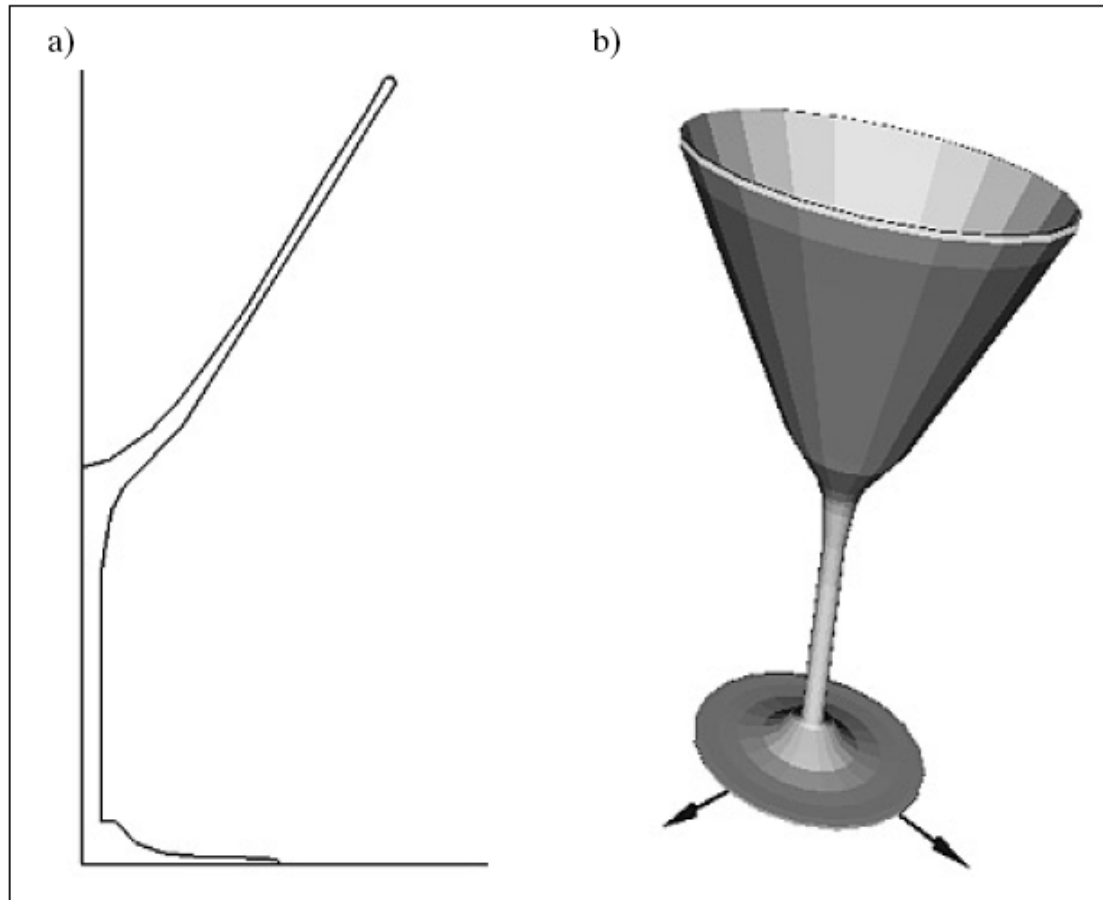
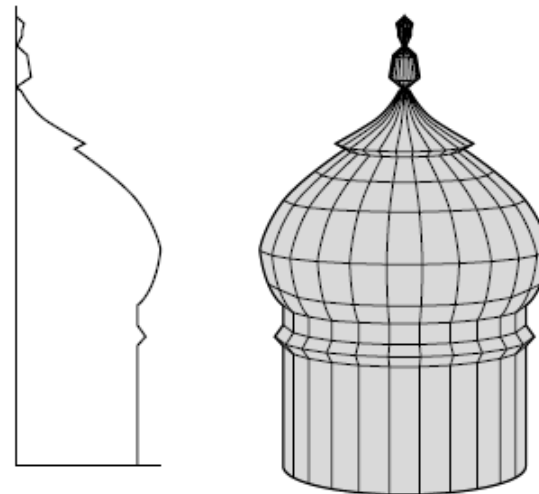


# Construcción de una malla por revolución en 3D

A. Gavilanes  
Departamento de Sistemas Informáticos y Computación  
Facultad de Informática  
Universidad Complutense de Madrid



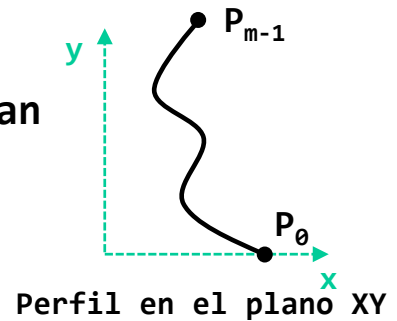
- ❑ Ingredientes para definir una malla por revolución
  - ❑ Un perfil formado por los puntos  $\{P_0, \dots, P_{m-1}\}$  sobre el plano  $XY$
  - ❑ Vértices: los obtenidos aplicando sucesivas rotaciones  $\text{rotacion}(360/n, (0,1,0))$ , con respecto al eje  $Y$ , a los puntos del perfil, donde  $n$  es el número de veces que se van a tomar muestras durante una revolución completa alrededor del eje  $Y$
  - ❑ Algunos vértices pueden repetirse. Por ejemplo, el punto más alto de la cúpula del Taj Mahal. Pero, cuidado, los vértices repetidos pueden dar problemas en el cálculo de vectores normales
  - ❑ Caras: las triangulares que resultan de dividir los cuadriláteros obtenidos uniendo dos puntos de un perfil con los dos correspondientes del perfil siguiente



# Construcción de una malla por revolución

(1) Crear una nueva clase **MbR** (de **Mesh by Revolution**) que hereda de la clase **IndexMesh** y que tiene los siguientes atributos:

(int) m: número de puntos del perfil  
(int) n: número de rotaciones (muestras) que se toman  
(dvec3\*) perfil: perfil original en el plano XY



En las explicaciones que siguen se supone que **los puntos del perfil van de abajo arriba**, tal como se muestra en la figura adjunta. Esto es importante para tener la seguridad de que los índices de las caras se dan en sentido antihorario

(2) La constructora de la clase **MbR** tiene tres parámetros y da valor a los atributos de la forma obvia.


(3) Definir en la clase **MbR** el método

```
static MbR* generaMallaIndexadaPorRevolucion(  
    int mm, int nn, glm::dvec3* perfil);
```

que obtiene los vértices de la malla, los índices y los vectores normales como se explica en las transparencias que siguen

## Construcción de una malla por revolución

```
MbR* MbR:: generaMallaIndexadaPorRevolucion(
    int mm, int nn, glm::dvec3* perfil) {
    MbR* mesh = new MbR(mm, nn, perfil);
    // Definir la primitiva como GL_TRIANGLES
    // Definir el número de vértices como nn*mm
    // Usar un vector auxiliar de vértices
    // dvec3* vertices = new dvec3[mesh->mNumVertices];
    for (int i=0; i<nn; i++) {
        // Generar la muestra i-ésima de vértices
        GLdouble theta = i * 360 / nn;
        GLdouble c = cos(radians(theta));
        GLdouble s = sin(radians(theta));
        // R_y(theta) es la matriz de rotación alrededor del eje Y
        for (int j = 0; j < mm; j++) {
            int indice = i * mm + j;
            GLdouble x = c * perfil[j].x + s * perfil[j].z;
            GLdouble z = -s * perfil[j].x + c * perfil[j].z;
            vertices[indice] = dvec3(x, perfil[j].y, z);
        }
    }
    ... (continúa)
```

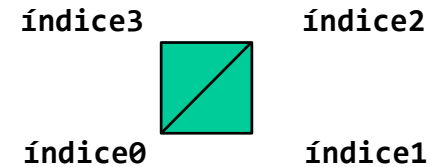


$$R_y(\theta) = \begin{pmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

## Construcción de una malla por revolución

(4) Volcar el array auxiliar **vertices** que acabamos de usar, en el array de vértices **mesh->vVertices** de la malla que está construyendo este método

(5) Construir los índices de las caras triangulares determinando los índices de las caras cuadrangulares y, a partir de ellos, dando los seis índices que corresponden a las dos caras triangulares resultantes de dividir la cara cuadrangular. Por ejemplo, si se conocen los índices de una cara cuadrangular como la de abajo, vista desde el exterior del objeto:



se añadirán al vector de índices los siguientes seis, que corresponden a las dos caras triangulares mencionadas

**índice0, índice1, índice2,**

**índice2, índice3, índice0**

Observa que si **índice2=índice3** entonces uno de los vectores que se usan para construir la normal de la cara triangular superior será nulo y eso resultará en un vector normal nulo que se sumará a los vértices de esa cara. Eso puede ser problemático

# Construcción de una malla por revolución usando índices

(6) Cómo determinar los índices de las caras cuadrangulares

// El contador  $i$  recorre las muestras alrededor del eje  $Y$

```
for (int i=0; i<nn; i++)
```

```
    // El contador  $j$  recorre los vértices del perfil,  
    // de abajo arriba. Las caras cuadrangulares resultan  
    // al unir la muestra  $i$ -ésima con la  $(i+1)\%nn$ -ésima
```

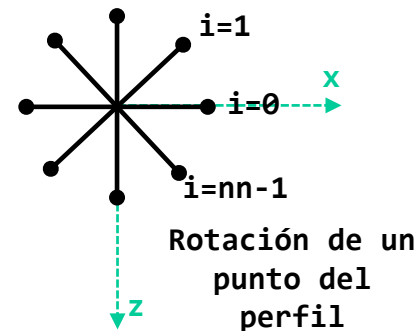
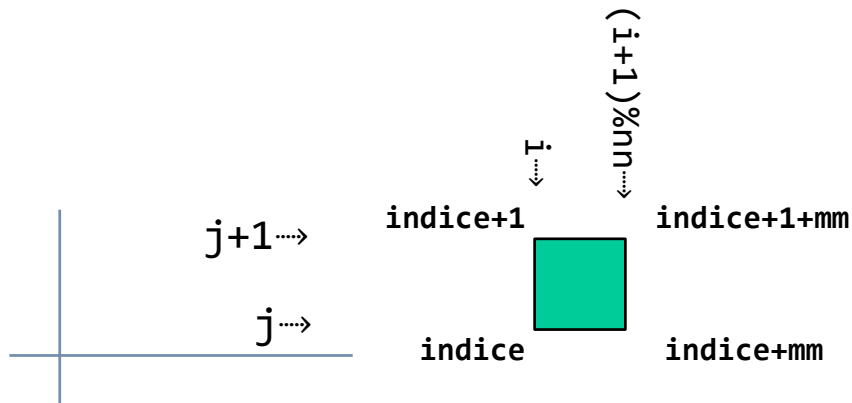
```
        for (int j=0; j<mm-1; j++)
```

```
            // El contador indice sirve para llevar cuenta  
            // de los índices generados hasta ahora. Se recorre  
            // la cara desde la esquina inferior izquierda
```

```
            int indice = i*mm+j;
```

```
            // Los cuatro índices son entonces:
```

```
            indice, (indice+mm)%nn*mm, (indice+mm+1)%nn*mm, indice+1
```



## Construcción de una malla por revolución

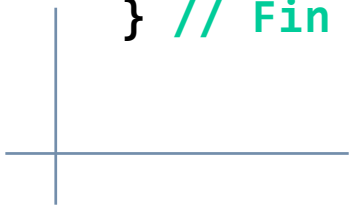
(7) Añadir seis índices al array `mesh->vIndices` usando un contador `indiceMayor` para ir rellenando este array

```
mesh->vIndices[indiceMayor] = indice;  
indiceMayor++;  
mesh->vIndices[indiceMayor] = (indice + mm) % (nn * mm);  
indiceMayor++;  
mesh->vIndices[indiceMayor] = (indice + mm + 1) % (nn * mm);  
indiceMayor++;
```

Y análogamente se añaden los otros tres índices

(8) Construir los vectores normales y devolver la malla

```
mesh->buildNormalVectors();  
return mesh;  
} // Fin del método generaMallaIndexadaPorRevolucion(...)
```





## Ejemplo. Construcción de un cono por revolución

- ❑ Definir una nueva entidad llamada **Cone** que hereda de **AbsEntity**
- ❑ Definir la constructora de **Cone**:

```
Cone::Cone(GLdouble h, GLdouble r, GLuint n) {  
    // h=altura del cono, r=radio de la base  
    // n=número de muestras, m=número de puntos del perfil  
    int m = 3;  
    dvec3* perfil = new dvec3[m];  
    perfil[0] = dvec3(0.5, 0.0, 0.0);  
    perfil[1] = dvec3(r, 0.0, 0.0);  
    perfil[2] = dvec3(0.5, h, 0.0);  
    this->mesh = new MbR(m, n, perfil);  
}
```