Modelado por revolución

Informática Gráfica I

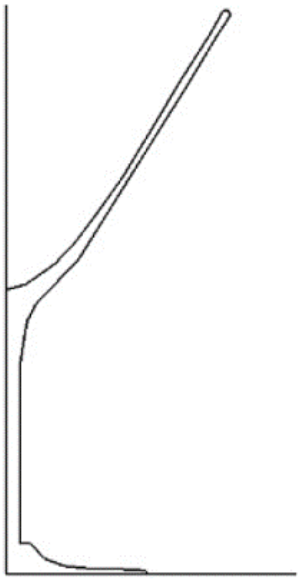
Material de: **Antonio Gavilanes**
Adaptado por: **Elena Gómez** y **Rubén Rubio**
{mariaelena.gomez,rubenrub}@ucm.es

UNIVERSIDAD COMPLUTENSE MADRID

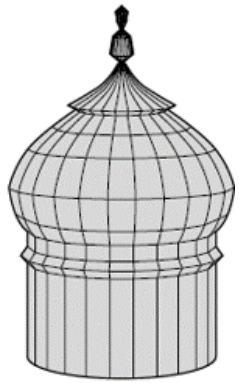
Contenido

Mallas por revolución

Un método muy sencillo para generar mallas interesantes es el modelado por revolución.







Mallas por revolución

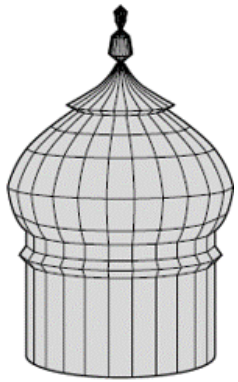
Ingredientes para definir una malla por revolución:

- **Perfil**: formado por los puntos $\{P_0, \cdots, P_{m-1}\}$ sobre el plano XY.
- **Número de muestras**: número n de veces que se repetirá el perfil durante una revolución completa al eje Y.

Mallas por revolución

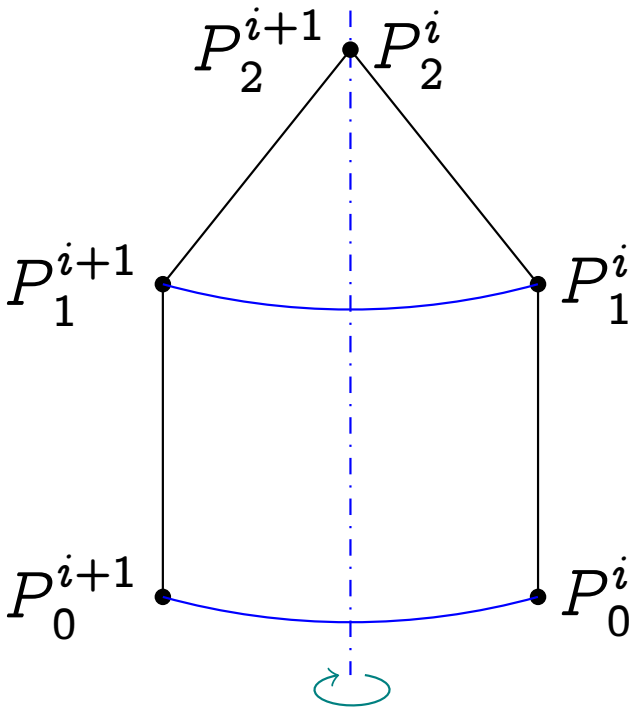
Los vértices de la malla serán los puntos que se obtengan aplicando sucesivas rotaciones de $360/n$ grados con respecto al eje Y a los puntos del perfil.

- Algunos vértices pueden repetirse (por ejemplo, el punto más alto de la cúpula del Taj Mahal) y eso puede dar problemas.



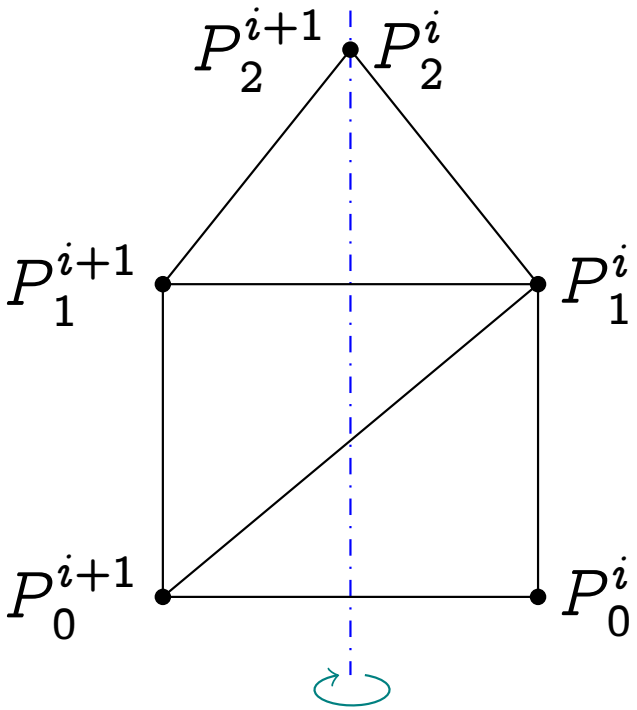
Mallas por revolución

Las **caras** son las triangulares que resultan de dividir los cuadriláteros obtenidos uniendo dos puntos de un perfil con los dos correspondientes del perfil siguiente.



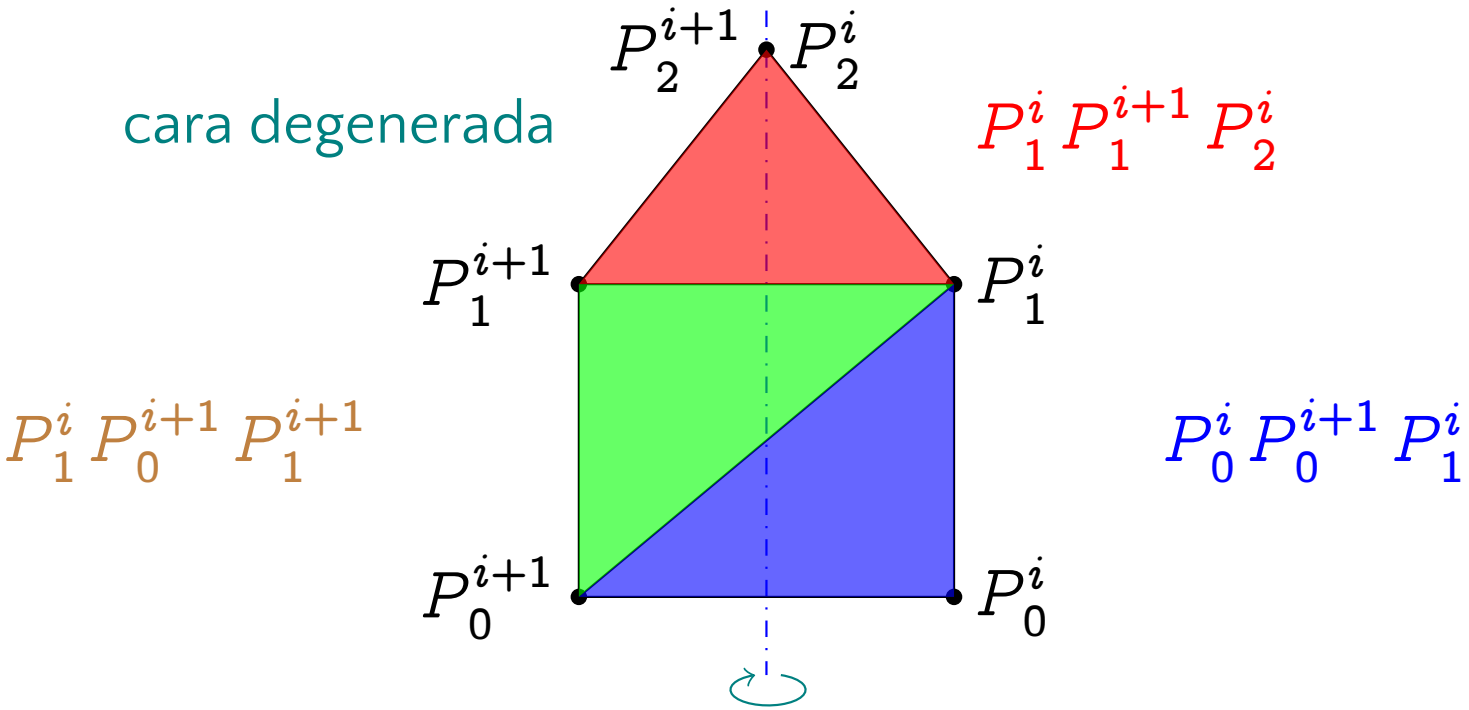
Mallas por revolución

Las **caras** son las triangulares que resultan de dividir los cuadriláteros obtenidos uniendo dos puntos de un perfil con los dos correspondientes del perfil siguiente.

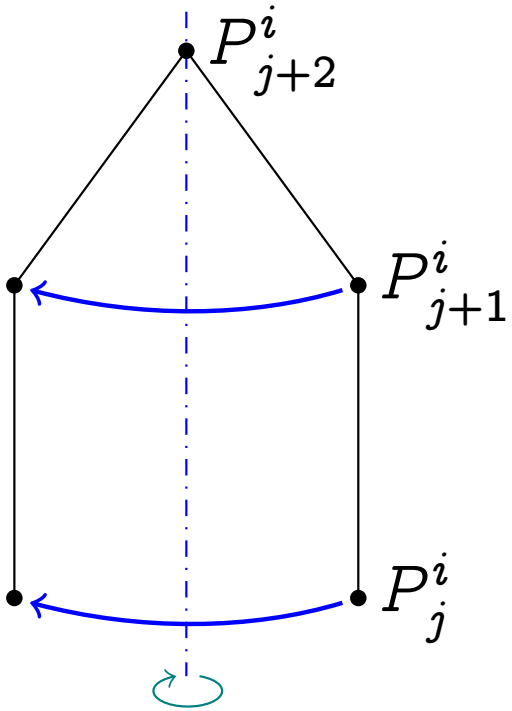


Mallas por revolución

Las **caras** son 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



Generamos n copias del perfil rotándolo sobre el eje Y:

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

donde $\theta_i = \frac{360i}{n}$. Así, si $P_j = (x, y)$,

$$P^i_j = (x\cos\theta_i, y, -x\sin\theta_i)$$

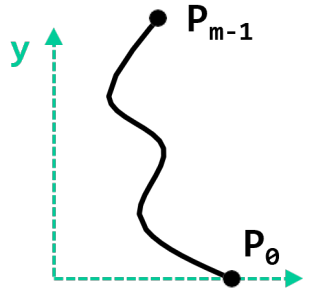
y definimos las caras como se ha visto.

Construcción de una malla por revolución

Creamos un nuevo método estático `generateByRevolution` en la clase `Mesh` con los siguientes atributos:

- `vector<glm::vec2> perfil`: perfil original en el plano XY
- `int nMuestras`: número de rotaciones (muestras) que se toman

En las explicaciones que siguen se supone que los puntos del perfil van de abajo arriba, 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**.

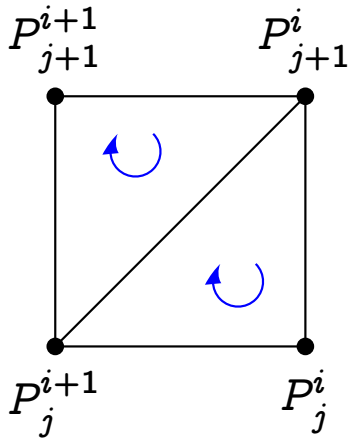


Constructor de la malla (1/3)

```
Mesh* Mesh::generateByRevolution(const vector<vec2>& perfil,
                                 int nMuestras) {
   Mesh* mesh = new Mesh;
   mesh->mPrimitive = GL_TRIANGLES;

   // Genera los vértices de las muestras
   vector<vector<vec3>> vs(nMuestras + 1);
   GLdouble theta1 = 2 * numbers::pi / nMuestras;

   for (int i = 0; i ≤ nMuestras; ++i) {  // muestra i-ésima
      GLdouble c = cos(i * theta1), s = sin(i * theta1);
      vs[i].reserve(perfil.size());
      for (auto p : perfil)  // rota el perfil
         vs[i].emplace_back(p.x * c, p.y, - p.x * s);
   }
```

Constructor de la malla (2/3)

```
for (int i = 0; i < nMuestras; ++i)  // caras i a i + 1
  for (int j = 0; j < perfil.size() - 1; ++j) { // una cara
    // Triángulo inferior (si no es degenerado)
    if (perfil[j].x ≠ 0.0) {
      mesh->vVertices.push_back(vs[i][j]);
      mesh->vVertices.push_back(vs[i+1][j]);
      mesh->vVertices.push_back(vs[i][j+1]);
    }

    // Triángulo superior (si no es degenerado)
    if (perfil[j + 1].x ≠ 0.0) {
      for (auto [s, t] : {pair{i, j+1}, {i+1, j}, {i+1, j+1}})
        mesh->vVertices.push_back(vs[s][t]); // más corto
    }
  }
```



Constructor de la malla (3/3)

```
    // Depende del número de caras degeneradas
    mesh->mNumVertices = mesh->vVertices.size();

    return mesh;
} // fin de Mesh::generateByRevolution
```

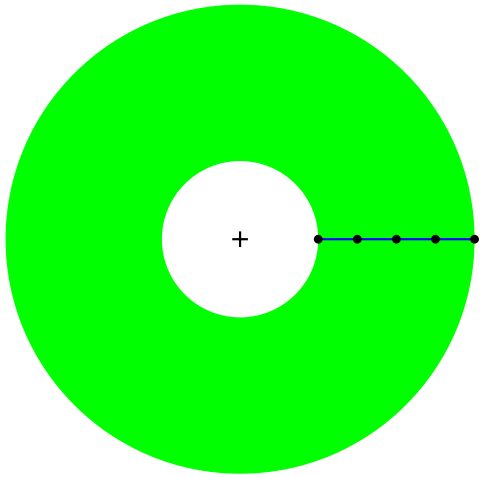
- Se pueden añadir también coordenadas de textura:

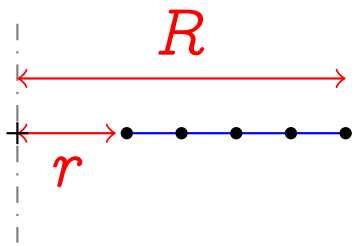
```
vTexCoords.emplace_back(i / nMuestras,
                        j / (perfil.size() - 1));
```

- Se podría hacer una revolución parcial controlada por un parámetro extra `GLdouble angleMax = 2 * numbers::pi` en `generateByRevolution`.

Disco

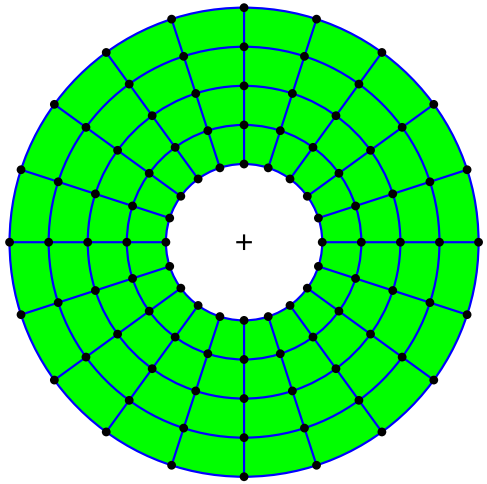
- En el plano $y = 0$, centrado en $(0, 0, 0)$ y de radios interior y exterior $r < R$.
- Si $r = 0$ es un disco completo, si no una corona circular.
- **Perfil:** $(x, 0)$ para $x \in \{r, \dots, R\}$

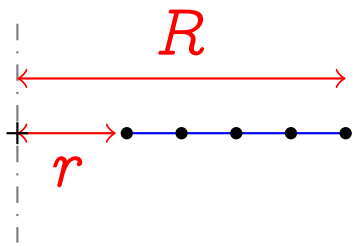




Disco

- En el plano $y = 0$, centrado en $(0, 0, 0)$ y de radios interior y exterior $r < R$.
- Si $r = 0$ es un disco completo, si no una corona circular.
- **Perfil:** $(x, 0)$ para $x \in \{r, \dots, R\}$





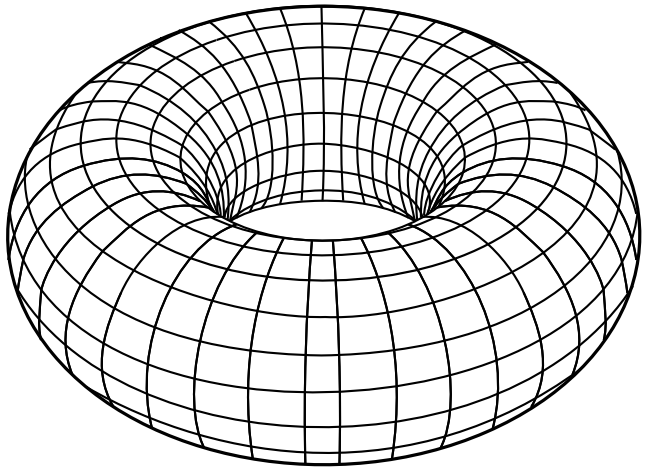
Disco: implementación

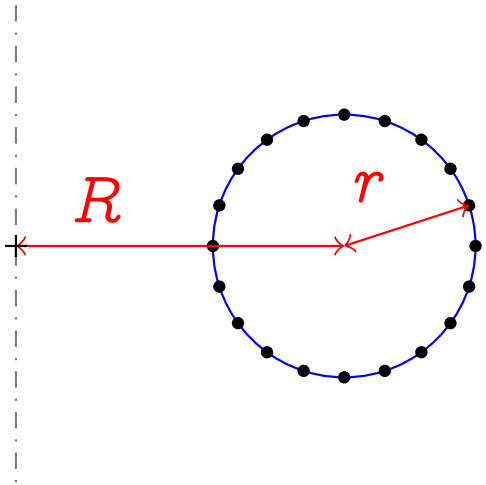
- Definir una nueva entidad llamada `Disc` que hereda de `SingleColorEntity`.
- Definir la constructora de `Disc`:

```
Disc::Disc(GLfloat R, GLfloat r, GLuint n) {
        // r = radio interior
        // R = radio exterior
        // n = número de muestras
        vector<vec2> perfil(2);
        perfil[0] = {r, 0.0};  // o más puntos
        perfil[1] = {R, 0.0};  // intermedios
        mMesh = new MbR(perfil, n);
}
```

Toro

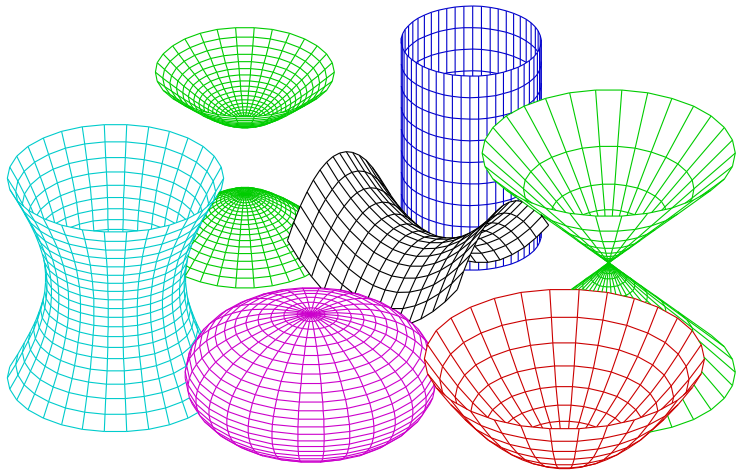
- Como una rosquilla.
- Centrado en $(0, 0, 0)$ con radios r y R.
- **Perfil:** $(r\cos(t) + R, r\sin(t))$ para $t \in \{0, \dots, 2\pi\}$





Entidades cuádricas

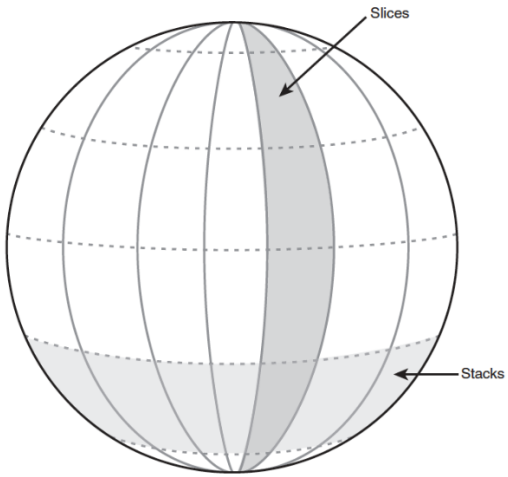
- Superficie geométrica tridimensional definida por una ecuación polinómica de segundo grado en tres variables (por ejemplo, x, y y z).
- Algunas muy conocidas de ellas son figuras de revolución: esferas, cilindros, conos o hiperboloides.
- Otras no lo son en general: elipsoide, paraboloide hiperbólico, etc.

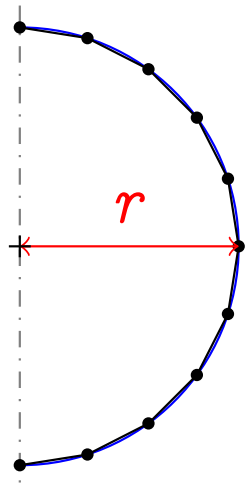


Esfera

$$x^2 + y^2 + z^2 = r^2$$

- Aquí centrada en $(0, 0, 0)$ y de radio r
- Cada punto del perfil define un **paralelo**.
- Cada paso de revolución es un **meridiano**.
- **Perfil:** $(r\cos(t), r\sin(t))$ para $t \in \{\frac{\pi}{2}, \dots, -\frac{\pi}{2}\}$

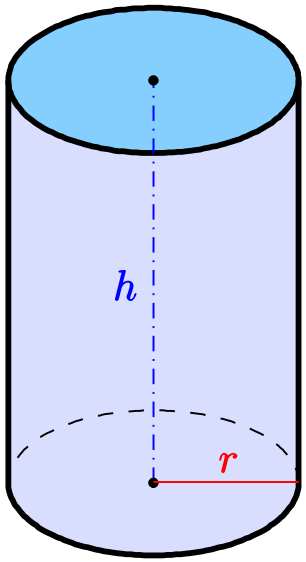


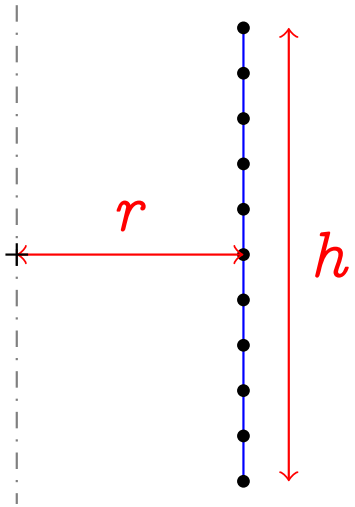


Cilindro

$$x^2 + z^2 = r^2$$

- Centrado en el eje y y de radio r.
- Se pintará solo para cierta altura h ($|y| \leq \frac{h}{2}$).
- **Perfil:** (r, y) para $y \in \{\frac{h}{2}, \dots, -\frac{h}{2}\}$

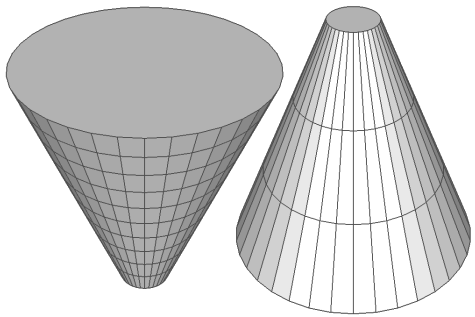


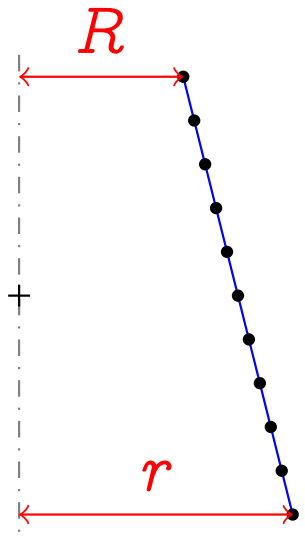


Cono truncado

$$x^2 + z^2 = ay + b \qquad a = \frac{R^2 - r^2}{h}, b = \frac{R^2 + r^2}{2}$$

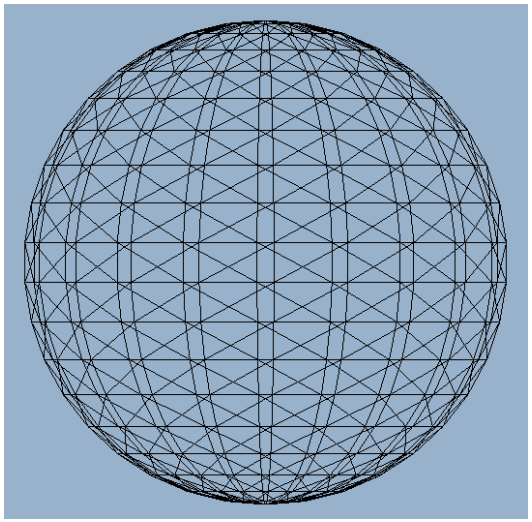
- Centrado en el eje y y de radios r y R.
- Se pintará solo para cierta altura h.
- Si $R = 0$ (o $r = 0$) será un cono (invertido).
- **Perfil:** $(ay + b, y)$ para $y \in \{\frac{h}{2}, \dots, -\frac{h}{2}\}$





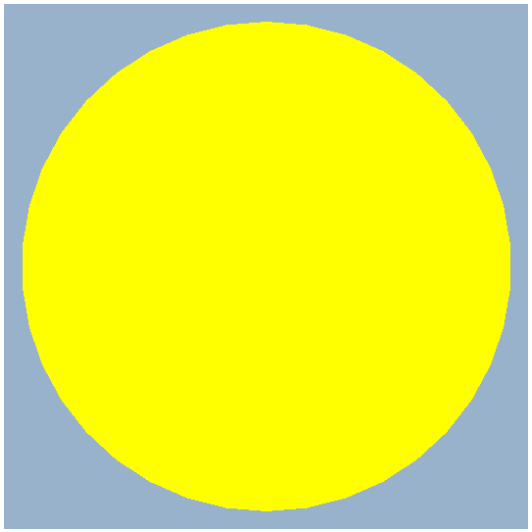
Renderizado de las figuras de revolución

Línea
(GL_LINE)



Textura

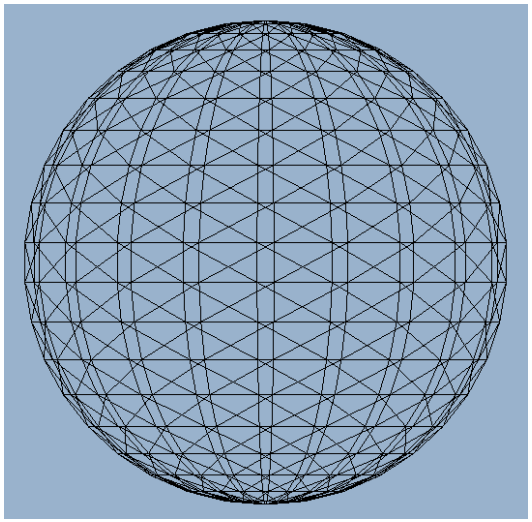
Relleno
(GL_FILL)



Con relleno
de color no se
percibe volumen

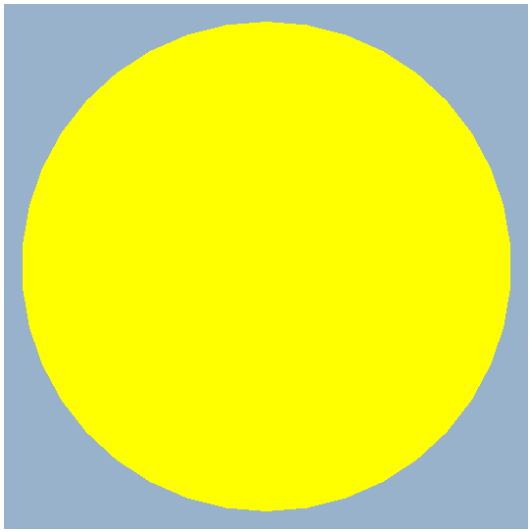
Renderizado de las figuras de revolución

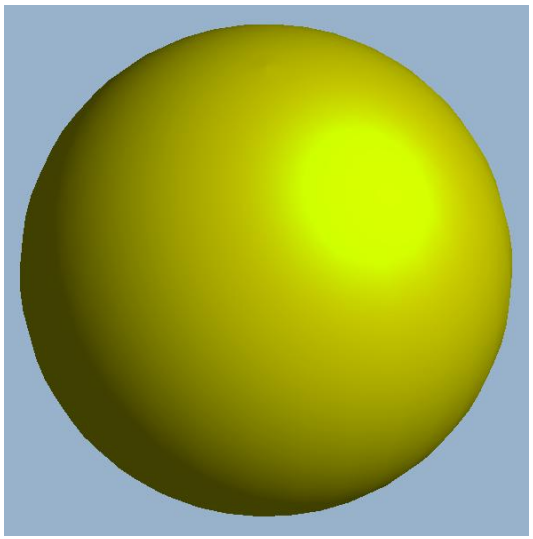
Línea
(GL_LINE)



Textura

Relleno
(GL_FILL)





Luz

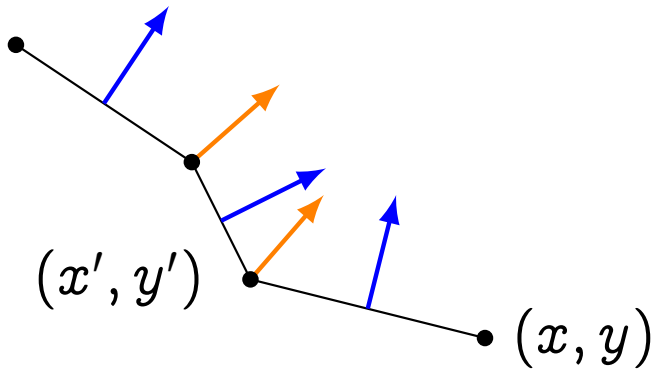
Iluminación y vectores normales

- Para iluminar un objeto (lo veremos en el último tema) es necesario asignar vectores normales a la malla.
- Un **vector normal** es un vector perpendicular al objeto (y a su plano tangente) en un punto.
- En la esfera el vector normal a un punto es su radio, en un cilindro es el radio de su sección horizontal, etc.

- En OpenGL, las normales se añaden como un dato adicional a los vértices (igual que el color o las coordenadas de textura).
- El **vector normal** a un vértice se toma como combinación de los vectores normales a las caras de las que forma parte.

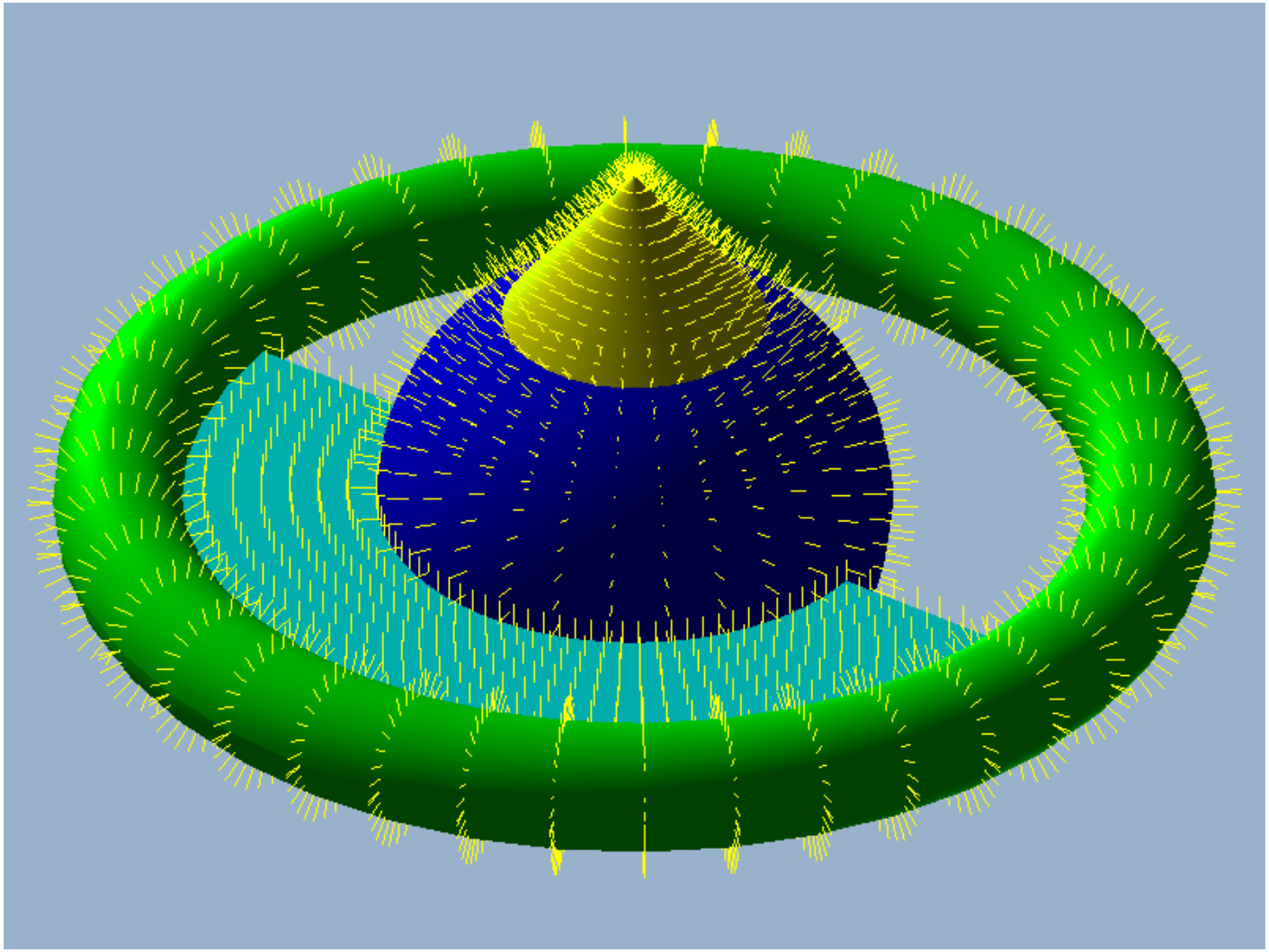
Normales en mallas de revolución

- El vector normal siempre está incluido en el plano que forman el eje de rotación y el perfil rotado.
- Se puede construir a partir del vector perpendicular al perfil.



- Un vector perpendicular al segmento que va de (x, y) a (x', y') es $(u, v) = (y' - y, -(x' - x))$.
- Una normal a la cara correspondiente es $(u \cos \theta_i, v, -u \sin \theta_i)$

Normales en mallas de revolución



Normales en mallas de revolución

- Al construir las mallas de revolución repetimos muchos vértices.
- Si además les añadimos coordenadas de textura, vectores normales... se repite mucha información.
- Veremos como resolverlo con **mallas indexadas**.