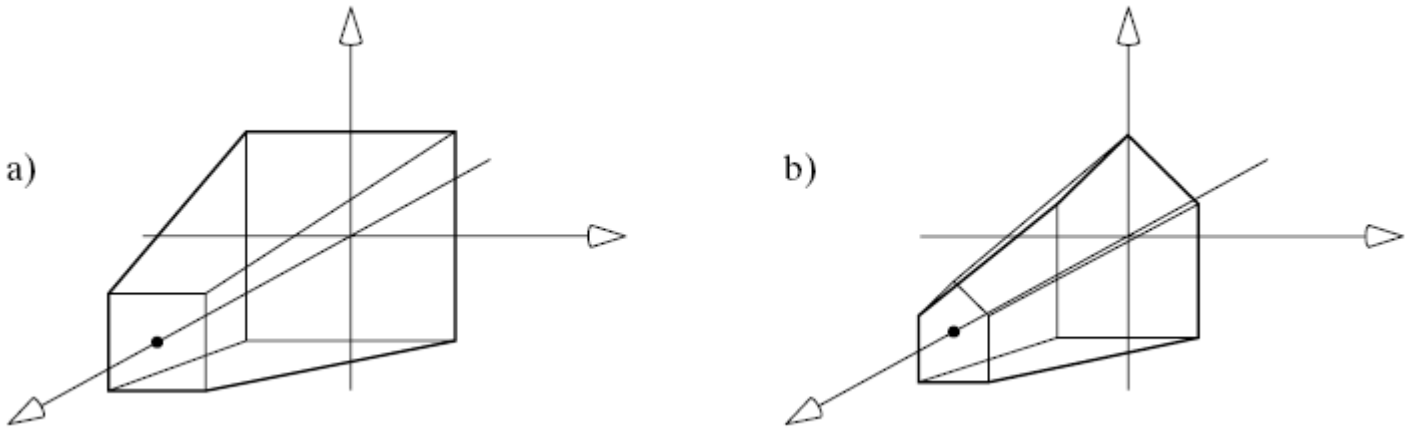
Mallas por extrusión

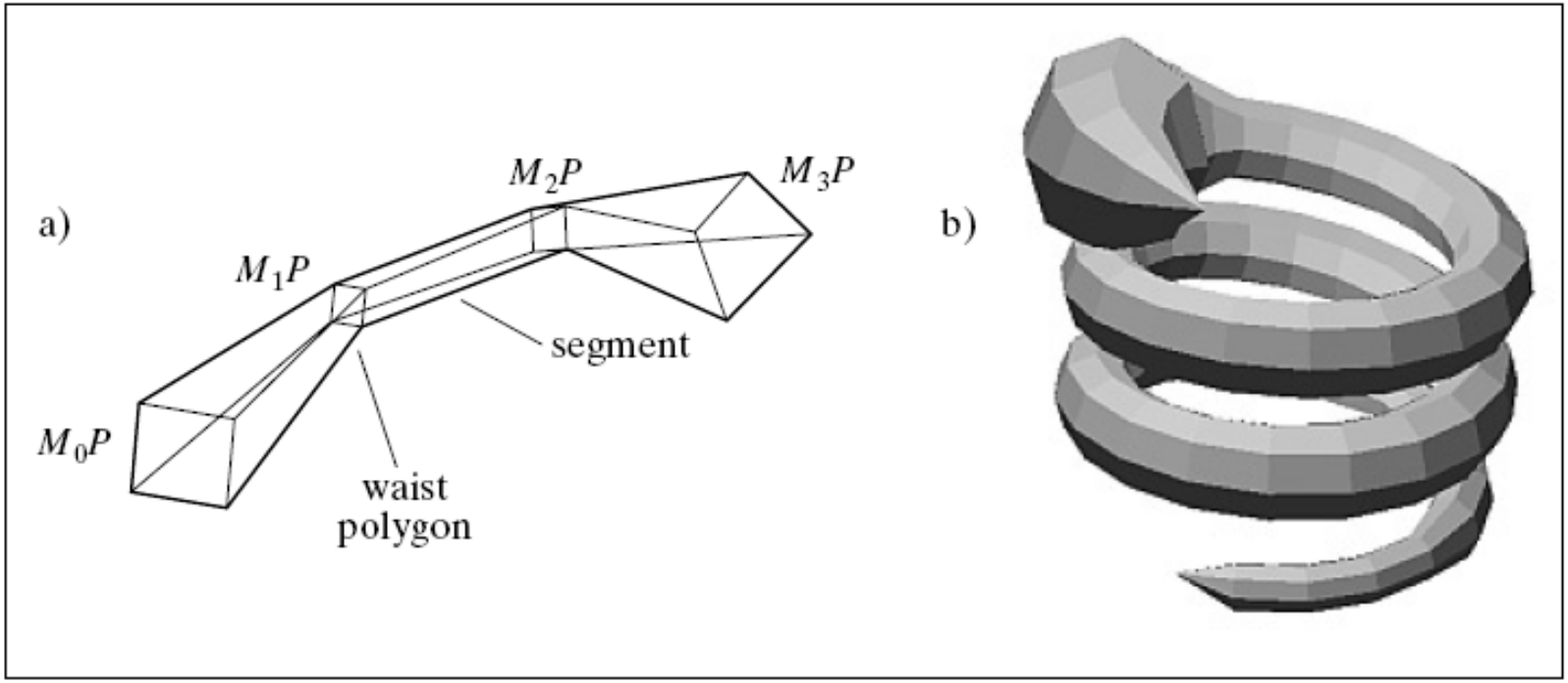
A. Gavilanes
Departamento de Sistemas Informáticos y Computación
Facultad de Informática
Universidad Complutense de Madrid

Mallas por extrusión

- Ingredientes para definir una malla por extrusión
  - Un polígono formado por los puntos $\{P_0, ..., P_{n-1}\}$
  - Una serie de transformaciones $\{M_0, ..., M_k\}$
  - Vértices del perfil j-ésimo: los obtenidos aplicando la transformación M_j a los puntos p_i del polígono inicial: $M_j(p_i)$, $0 \le i \le n-1$, $0 \le j \le k$
  - Caras: las cuadrangulares obtenidas como se vio con las caras laterales en los prismas rectos

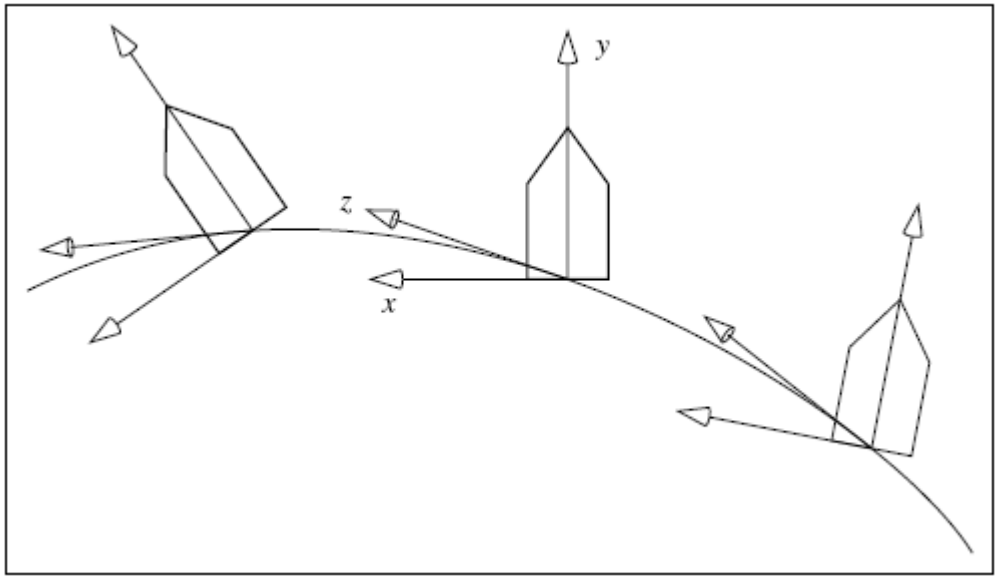


Mallas por extrusión



Construcción de una malla por extrusión usando el marco de Frenet

- Se parte de **un perfil dado** y se producen sucesivos perfiles, por aplicación de una transformación que coloca el sistema de coordenadas local, que hemos usado para modelar el perfil, sobre un punto de la escena con la orientación deseada. Estos puntos recorren **una curva dada**.
- Los puntos de los sucesivos perfiles se unen, como se hace en cualquier malla por extrusión.

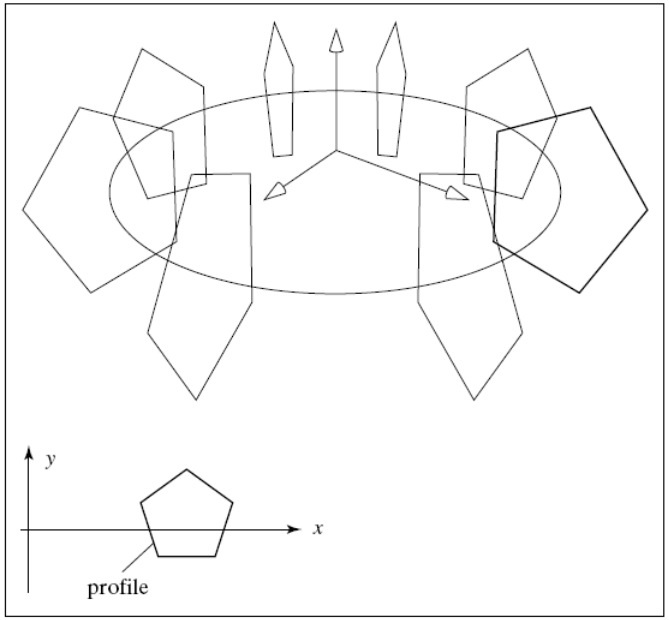


Construcción de una malla por extrusión usando el marco de Frenet. El caso del toro

- Perfil inicial: polígono regular sobre el plano XY de nP lados, inscrito en una circunferencia de radio r

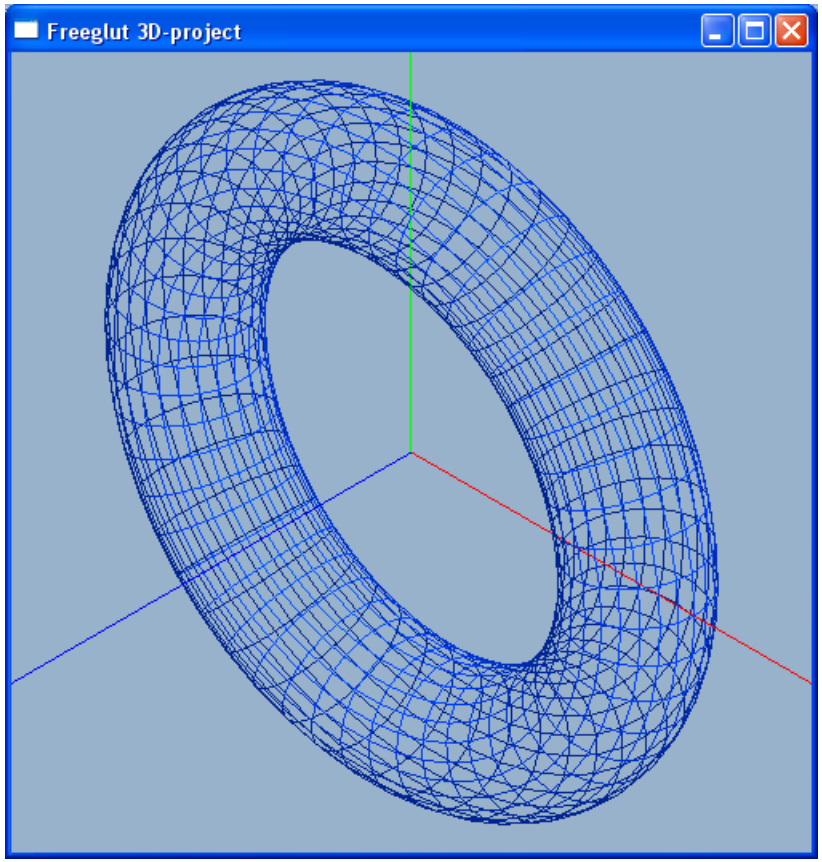
```
inc=(2*PI/nP);
for (int i=0; i<nP; i++)
   perfil[i]=dvec3(r*cos(i*inc), r*sin(i*inc),0);
```

- Curva dada: C(t)=(R*cos(t), 0, R*sin(t)) (ecuaciones paramétricas de la circunferencia centrada en el origen, sobre el plano XZ, de radio R).

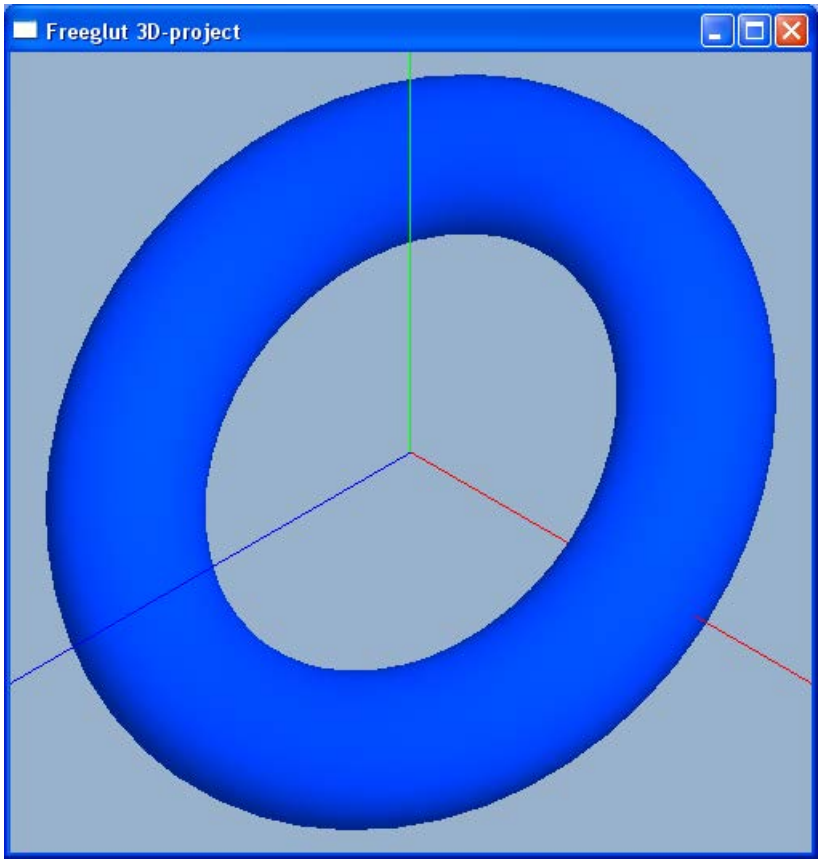


Toro con la librería glut

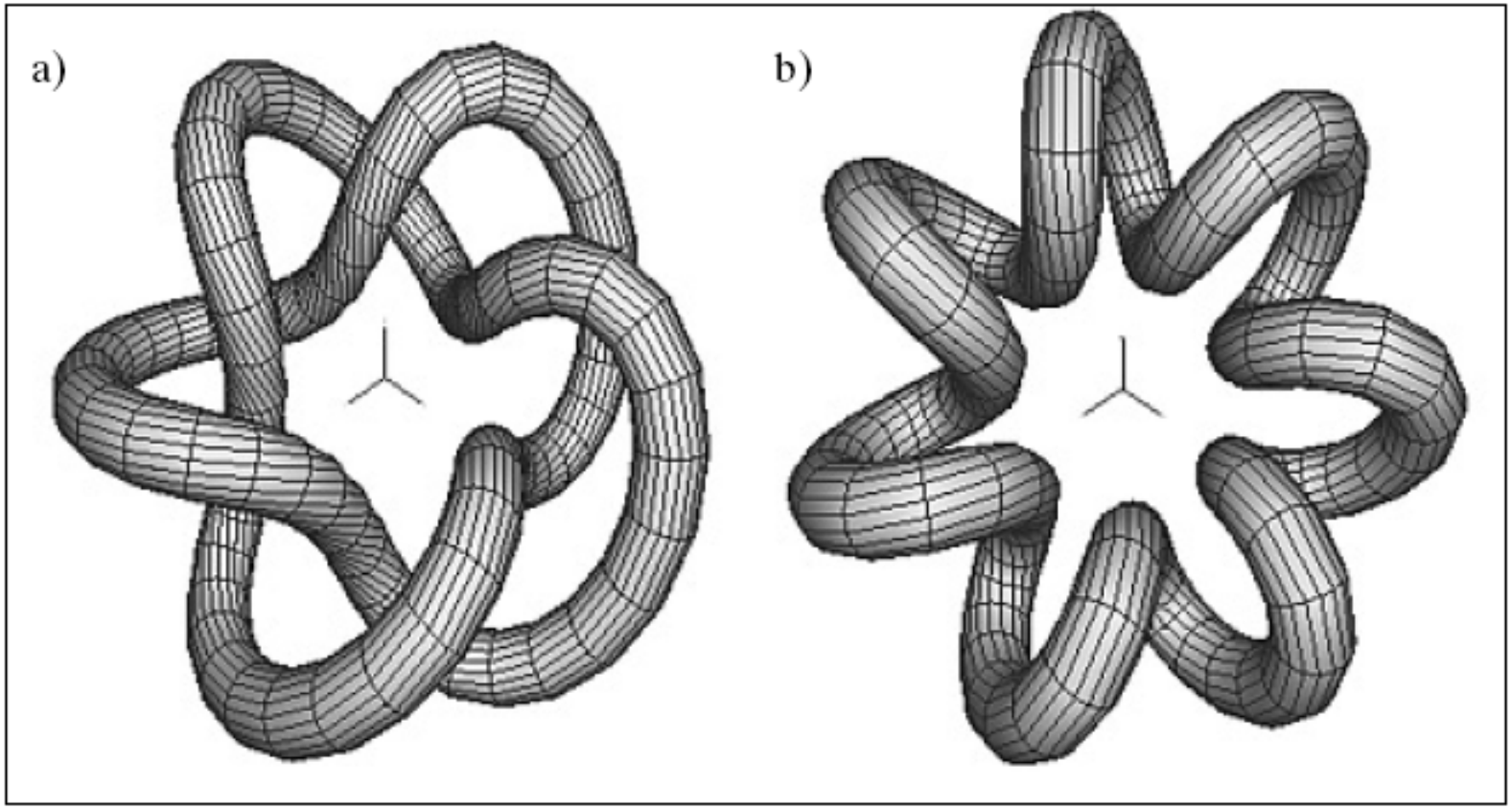
```
glutWireTorus(2, 8, 25, 50);
```




```
glutSolidTorus(2, 8, 25, 50);
```



Otras curvas

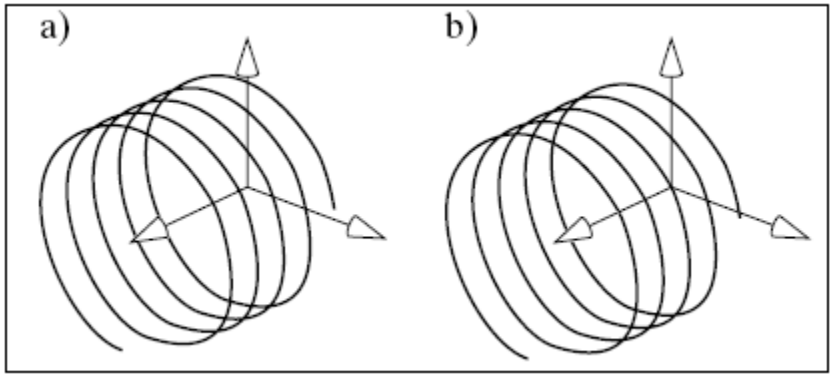


Construcción de una malla por extrusión usando el marco de Frenet

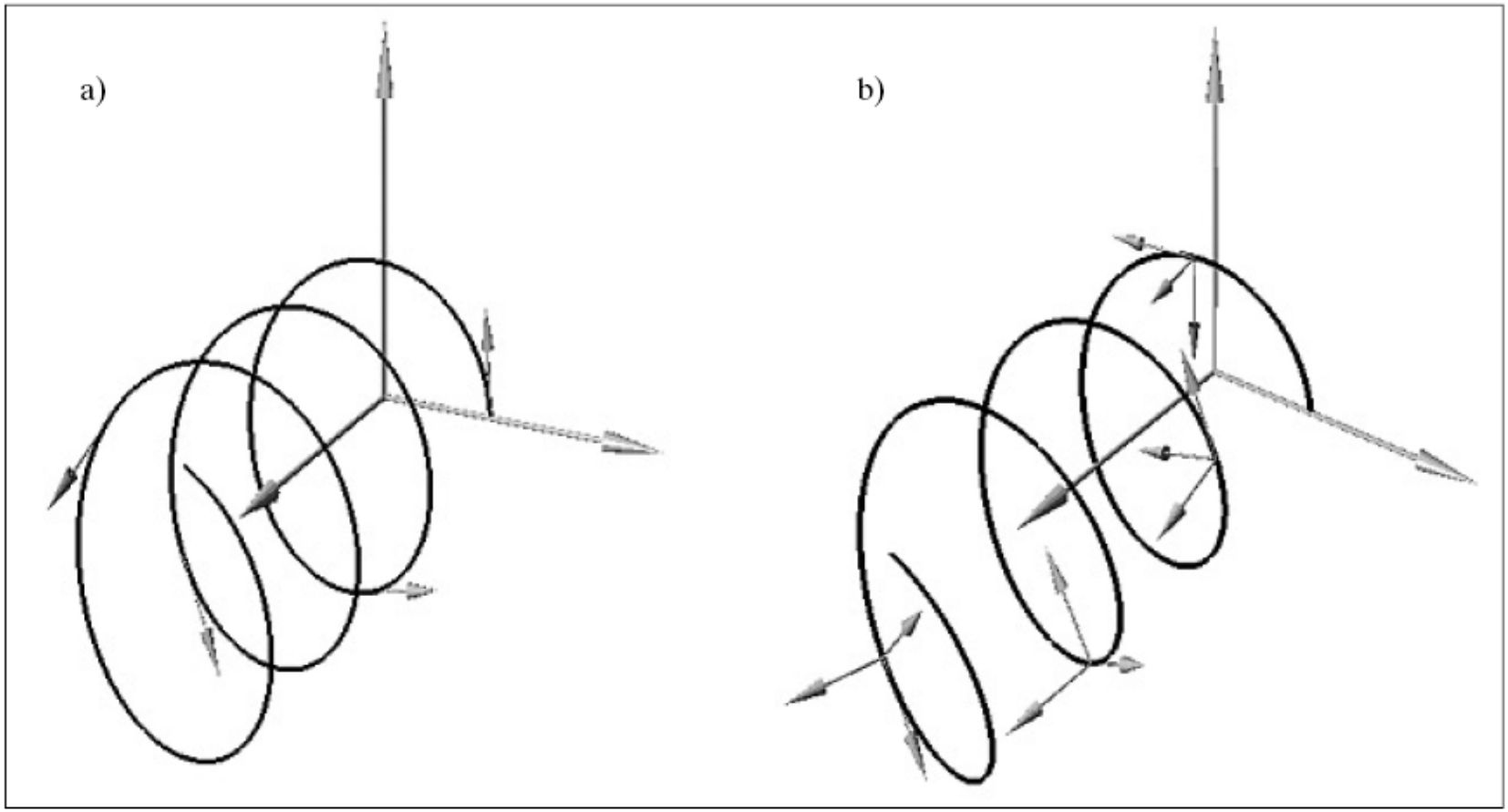
- Se busca obtener la transformación que coloca el sistema de referencia local (del perfil) en un punto de la curva C(t). Además queremos orientar el sistema de referencia que resulta para que su plano XY sea perpendicular a C(t).
- La primera derivada: C'(t) (vector tangente a C(t)).
- La segunda derivada: C''(t) (vector curvatura de C(t)).
- El vector T(t) es la normalización de C'(t).
- El vector binormal B(t) es la normalización de C'(t)×C''(t).
- El vector N(t) es B(t)×T(t). Ya está normalizado.
- La transformación es la matriz 4×4 M(t)=(**N(t)**, **B(t)**, **T(t)**, **C(t)**), donde las tres primeras columnas son vectores (acaban en 0) y la última, punto (acaba en 1).
- Ejemplo sobre la hélice:

  C(t)=(cos(t), sin(t), b*t)

  b constante

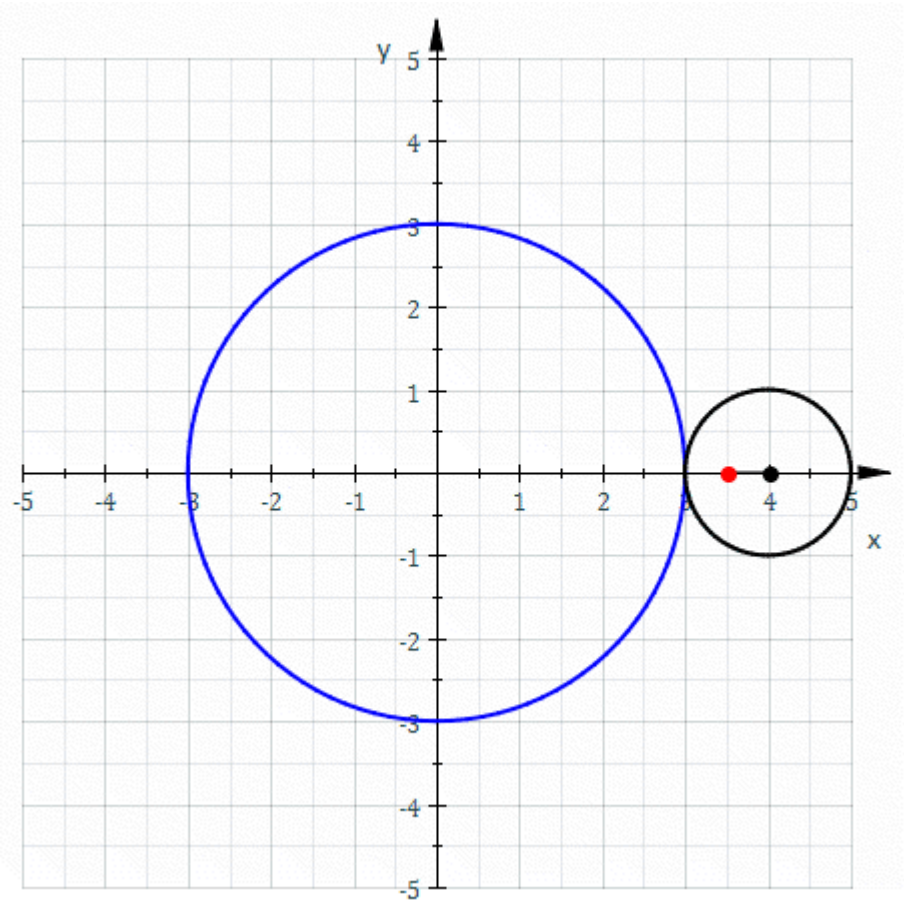


Marco de Frenet



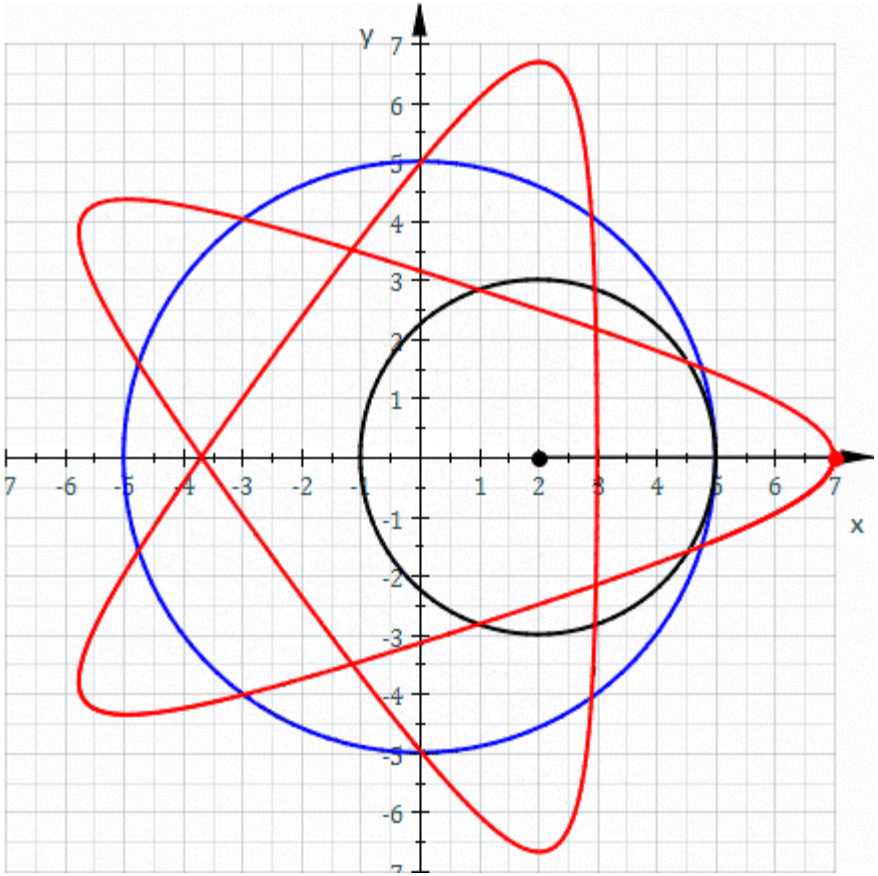
La epitrocoide

La **epitrocoide** es la curva que describe un punto vinculado a una circunferencia generatriz que rueda –sin deslizamiento– sobre una circunferencia directriz, de forma tangencial.

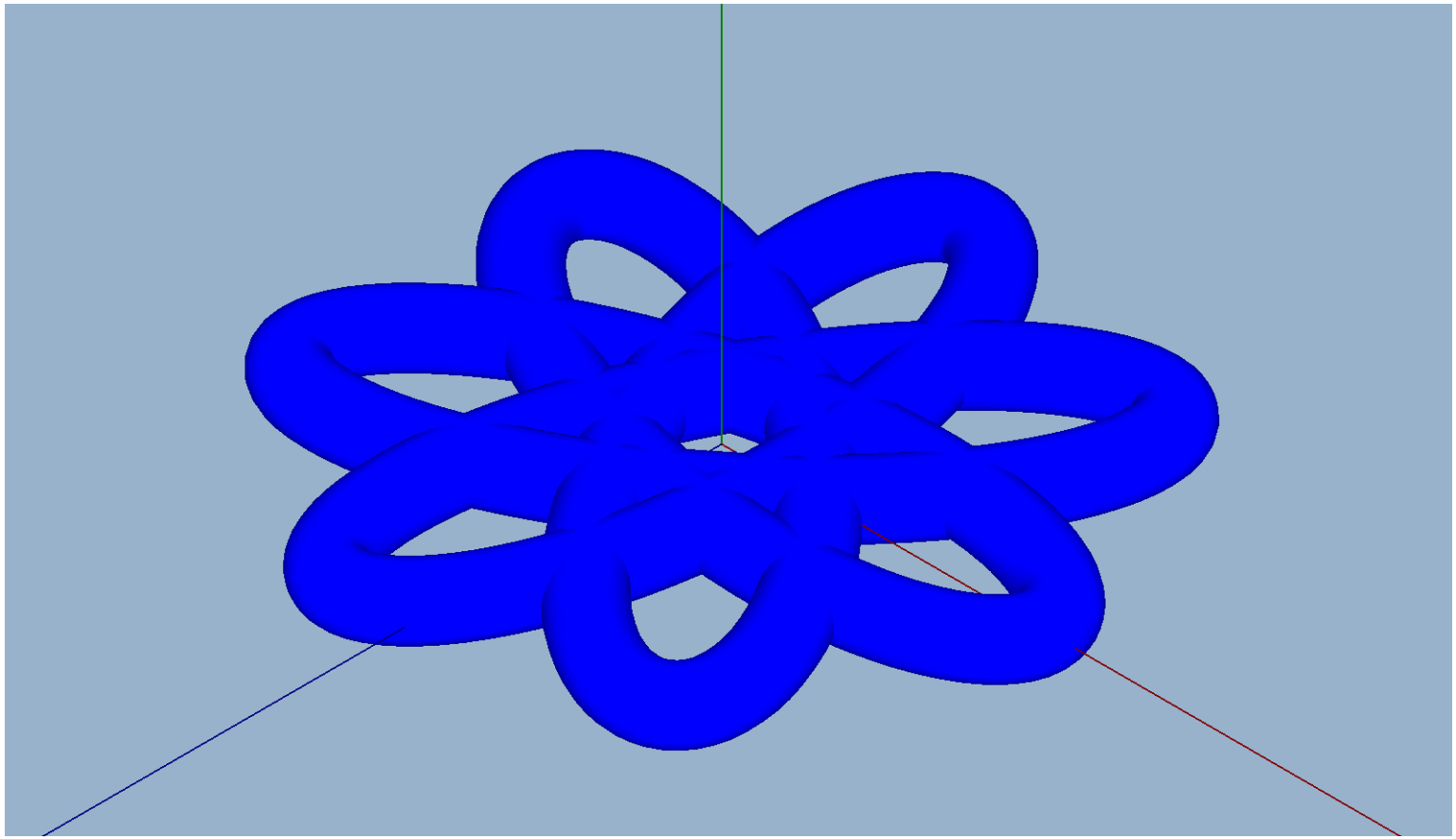


La hipotrocoide

La **hipotrocoide** es la curva que describe un punto vinculado a una circunferencia generatriz que rueda –sin deslizamiento– dentro de una circunferencia directriz, de forma tangencial.



Tubo de una hipotrocoide



Construcción de una malla por extrusión: la hipotrocoide

(1) Definir una clase **HipoMesh**, que hereda de la clase **Mesh**, y que tiene los siguientes atributos:

- **int nP:** Número de lados del polígono que aproxima la circunferencia que define "el tubo"
- **int nQ:** Número de rodajas que forman "el tubo"
- **GLfloat a, b, c:** Valores de los parámetros de las ecuaciones paramétricas de la hipotrocoide
- **dmat4 m:** Matriz de paso de coordenadas locales a globales. En realidad es una matriz **m(t)**
- **dvec3* base:** Perfil del **nP**-ágono que aproxima la circunferencia que define el tubo

Construcción de una malla por extrusión: la hipotrocoide

(2) Definir la siguiente funcionalidad en la clase **HipoMesh**:

- **void creaBase(): Guarda en el array base el polígono que aproxima la circunferencia "del tubo"**
- **void creaVerticesIniciales(): Añade los primeros nP vértices de la malla a base de aplicar m(0) a los vértices de base. Usa pues este otro:**
  - **dvec3 aplica(dvec3 p): Devuelve el resultado de aplicar la matriz m al punto p**
- **void creaRodaja(int v): Añade nP nuevos vértices, a partir de la componente v del array de vértices, como resultado de aplicar m a los vértices de base**
- **void cargaMatriz(GLdouble t): Define la matriz m para el parámetro t**

Construcción de una malla por extrusión: la hipotrocoide

(2) Definir la siguiente funcionalidad en la clase **HipoMesh** (continuación):

- **void cargaMatriz(GLdouble t):** Define la matriz **m** para el parámetro **t**, es decir **m(t)**. Usa los métodos:
  - **dvec3 curva(GLdouble t):** Devuelve el punto de la curva para el parámetro **t**. Para obtener la malla por Frenet de otras curvas solo es necesario cambiar este y los dos métodos siguientes
  - **dvec3 derivada(GLdouble t):** Ídem de la primera derivada
  - **dvec3 segundaDerivada(GLdouble t):** Íd. de la segunda
  - **dvec3 te(GLdouble t):** Devuelve el vector **T(t)** normalizado
  - **dvec3 be(GLdouble t):** Devuelve el vector **B(t)** normalizado
  - **dvec3 ne(GLdouble t):** Devuelve el vector **N(t)** normalizado

Construcción de una malla por extrusión: la hipotrocoide

(3) Reescribir el método **`normalize()`** de **`HipoMesh`** para construir las normales de los vértices de la hipotrocoide. Como en las mallas por revolución, el vector normal a un vértice se obtiene normalizando la suma de los vectores normales a las caras en las que participa dicho vértice.

- Los índices de los vértices de las caras no son los mismos que para el caso de la malla por revolución
- Es más cómodo distinguir el caso de la unión de la última rodaja con la primera como caso especial

Construcción de una malla por extrusión: la hipotrocoide

(4) Definir la constructora de la clase **HipoMesh** como **HipoMesh(int nP, int nQ, GLfloat a, GLfloat b, GLfloat c)**. Observa que el número de vértices es **nP+nP*nQ** y que la constructora, además de dar valor a ciertos atributos de la forma esperada, realiza el siguiente proceso para obtener los vértices:

```
GLdouble t = 0.0; GLdouble saltoEntreRodajas = 4.0*2.0*PI / nQ;

creaBase();

cargaMatriz(t);

creaVerticesIniciales();

        for (int i = 0; i<nQ; i++) {

        t += saltoEntreRodajas;

        cargaMatriz(t);

        creaRodaja(…);

}

normalize();
```

Construcción de una malla por extrusión: la hipotrocoide

(5) Reescribir el método **`render()`** de **`HipoMesh`**, con el comando **`glDrawElements(…)`** que usa índices, como en el caso de las mallas por revolución

(6) Definir la clase **`Hipotrocoide`**, que hereda de la clase **`Entity`**, que tiene los siguientes atributos:

```
int nP;
int nQ;
GLfloat a, b, c;
```

Y cuya constructora concluye con:

```
this->mesh = new HipoMesh(nP, nQ, a, b, c);
```