Auxiliar 8

Curvas y Superficies

Pablo Pizarro, pablo@ppizarror.com

Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ciencias de la Computación

1 de junio de 2022

¿Qué veremos hoy?

1. Curvas
   - Curvas de Hermite
   - Curvas de Catmull-Rom
   - Curvas de Bézier
2. Extendiendo a superficies
   - Bézier
   - Catmull-Rom
   - Superficies con sólidos de revolución

Curvas

Las curvas se definen por expresiones matriciales, las que consideran tanto las restricciones geométricas y topológicas.

$$Q(t) = \underbrace{\begin{bmatrix} g_1 & g_2 & g_3 & g_4 \end{bmatrix}}_{G} \underbrace{\begin{bmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \\ a_4 & b_4 & c_4 & d_4 \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix}}_{T(t)}$$

Figura 1: Formulación clásica generador de curvas; G matriz geometria, M matriz base, t permite variar el parámetro para generar la curva entre $[0, 1]$.

Curvas de Hermite

Están definidas por dos puntos P_1 y P_2 y dos direcciones tangentes T_1 y T_2.

$$H(t) = (1 - 3t^2 + 2t^3)P_1 + t^2(3 - 2t)P_2 + t(t - 1)^2 T_1 + t^2(t - 1) T_2$$

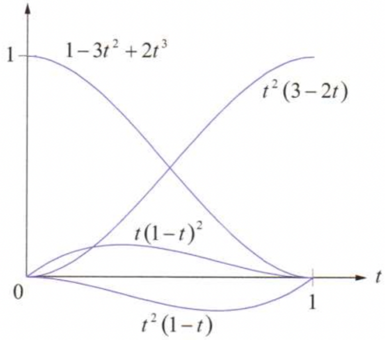


Figura 2: Formulación matemática de las curvas de Hermite.

Ejemplos Hermite

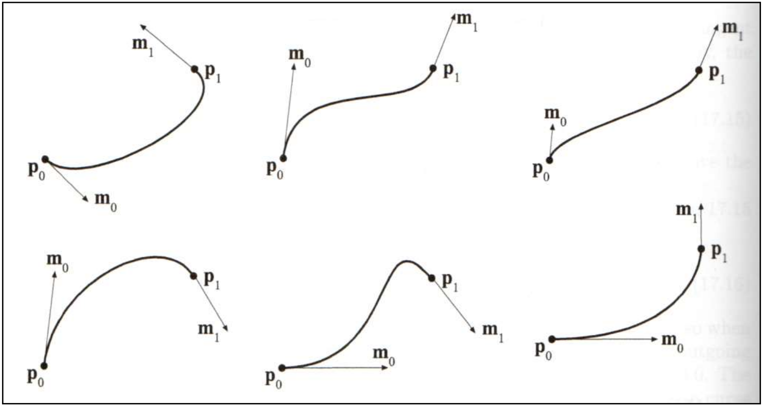


Figura 3: Ejemplos de curvas de Hermite.

Implementación Hermite

Para implementar las curvas de Hermite, necesitamos básicamente definir las operaciones matriciales y la función generadora, que recibe estas matrices generadoras y retorna un vector de coordenadas. En particular, usando las librerías `numpy` y `matplotlib`.

Código 1: Implementación de Hermite

```python
import numpy as np

def hermiteMatrix(P1, P2, T1, T2):

    # Concatena los puntos en una columna
    G = np.concatenate((P1, P2, T1, T2), axis=1)

    # Matriz base
    Mh = np.array([[1, 0, -3, 2], [0, 0, 3, -2], [0, 1, -2, 1], [0, 0, -1, 1]])

    # Retorna la matriz de Hermite
    return np.matmul(G, Mh)
```

Si quisiéramos graficar una curva...

Para graficar la curva, necesitamos generar todos los puntos de la misma, modificando el parámetro t entre $[0, 1]$ y almacenando sus puntos dentro de un vector.

Código 2: Graficando Hermite

```
def generateT(t):
  return np.array([[1, t, t ** 2, t ** 3]]).T

def evalCurve(M, N):
  # Varía el parámetro de tiempo en N samples
  ts = np.linspace(0.0, 1.0, N)

  # Vector para almacenar la curva
  curve = np.ndarray(shape=(N, 3), dtype=float)

  # Calcula cada valor de las coordenadas y las almacena
  for i in range(len(ts)):
  T = generateT(ts[i])
  curve[i, 0:3] = np.matmul(M, T).T

  return curve
```

Ejemplo Hermite

Finalmente, para graficarla, requerimos de las funciones anteriores:

Código 3: Graficando Hermite

```
from curvas import *

# Ejemplo para Hermite
P1 = np.array([[0, 0, 1]]).T; P2 = np.array([[1, 0, 0]]).T
T1 = np.array([[10, 0, 0]]).T; T2 = np.array([[0, 10, 0]]).T

GMh = hermiteMatrix(P1, P2, T1, T2)
hermiteCurve = evalCurve(GMh, N=50)

# Definimos la figura para 3D
fig = plt.figure()
ax = fig.gca(projection='3d')

plotCurve(ax, hermiteCurve, 'Hermite curve', (1, 0, 0))

ax.set_xlabel('x');ax.set_ylabel('y');ax.set_zlabel('z');ax.legend()
plt.show()
```

El ejemplo anterior debería verse algo así:

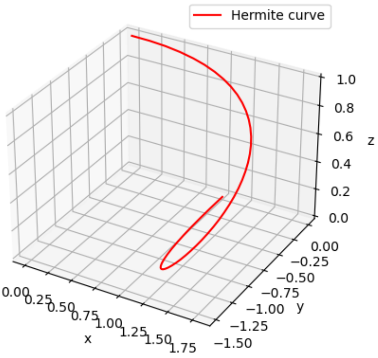


Figura 4: Gráfico ejemplo Hermite.

Curvas de Catmull-Rom

Están definidas por cuatro puntos de control P_1, P_2, P_3 y P_4, con una restricción tangente a cada uno.

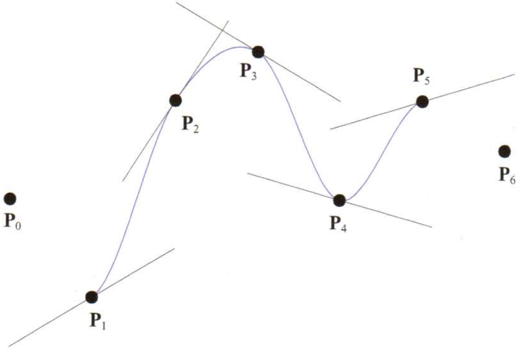


Figura 5: Curvas de Catmull-Rom.

Catmull-Rom, defininición matricial

Al igual que las curvas de Hermite, Catmull-Rom también requiere de la definición matricial del operador para generar la curva

$$C_i(t) = \begin{bmatrix} P_{i-1} & P_i & P_{i+1} & P_{i+2} \end{bmatrix} M_{CR} T(t)$$

$$M_{CR} = \frac{1}{2} \begin{bmatrix} 0 & 0 & -1 & 0 \\ 2 & 0 & 0 & -1 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -3 & 2 \\ 0 & 0 & 3 & -2 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -1 & 2 & -1 \\ 2 & 0 & -5 & 3 \\ 0 & 1 & 4 & -3 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

Curvas de Bézier

Las curvas de Bézier, finalmente, se ajustan para un polinomio de grado n. Pero, si usamos la misma definición anterior con cuatro puntos:

$$B(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t) P_2 + t^3 P_3$$

$$B(t) = \underbrace{\begin{bmatrix} P_0 & P_1 & P_2 & P_3 \end{bmatrix}}_{G_B} \underbrace{\begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{M_B} \underbrace{\begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix}}_{T(t)}$$

Implementando Bézier

Como lo hicimos en el caso de las curvas de Hermite, necesitamos definir la operación anterior de manera matricial.

Código 4: Implementación de Bézier.

```
def bezierMatrix(P0, P1, P2, P3):
    # Concatena los puntos como una columna
    G = np.concatenate((P0, P1, P2, P3), axis=1)

    # Matriz Bézier base
    Mb = np.array([[1, -3, 3, -1], [0, 3, -6, 3], [0, 0, 3, -3], [0, 0, 0, 1]])

    return np.matmul(G, Mb)
```

Ejemplo Bézier

Haciendo uso de las funciones anteriormente definidas, podemos graficar una curva de Bézier como sigue:

Código 5: Ejemplo de Bézier.

```
from curvas import *

R0 = np.array([[0, 0, 1]]).T; R1 = np.array([[0, 1, 0]]).T
R2 = np.array([[1, 0, 1]]).T; R3 = np.array([[1, 1, 0]]).T

GMb = bezierMatrix(R0, R1, R2, R3)
bezierCurve = evalCurve(GMb, N=50)

#  Definimos la figura para 3D
fig = plt.figure(); ax = fig.gca(projection='3d')
plotCurve(ax, bezierCurve, 'Bezier curve')

# Visualizamos los puntos de control
controlPoints = np.concatenate((R0, R1, R2, R3), axis=1)
ax.scatter(controlPoints[0, :], controlPoints[1, :], controlPoints[2, :], color=(1, 0, 0))

ax.set_xlabel('x'); ax.set_ylabel('y'); ax.set_zlabel('z'); ax.legend(); plt.show()
```

Ejemplo Bézier

El ejemplo anterior debería verse algo así:

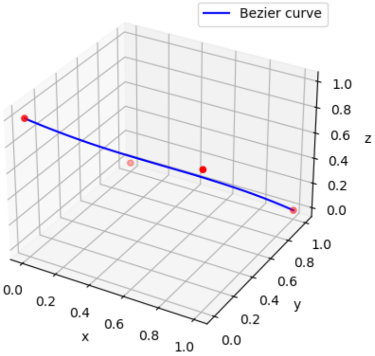


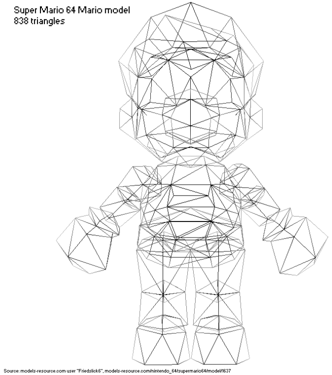
Figura 6: Gráfico ejemplo Bézier.

Qué nos queda...

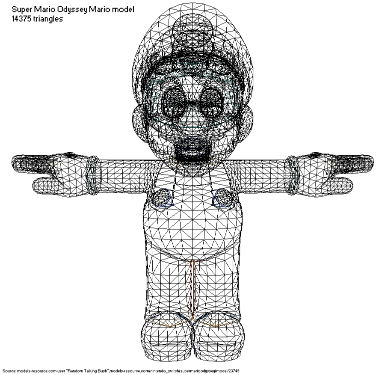
Generación de superficies con curvas

- Existen varios métodos para poder generar superficies curvas:
  - Usando splines
  - Sólidos de revolución
  - A mano ☹, ingresando vértice a vértice los puntos
  - Usando un software de modelado 3D, como Maya o Blender
- Todas estas metodologías se basan en el mismo principio:
  1. Generar los vértices $v_i = (x_i, y_i, z_i)$
  2. Conectar los vértices generados de manera ordenada usando triángulos y agrupándolos en Shapes

Modelos, compuestos por muchas geometrías, ¿Cómo definirlas?



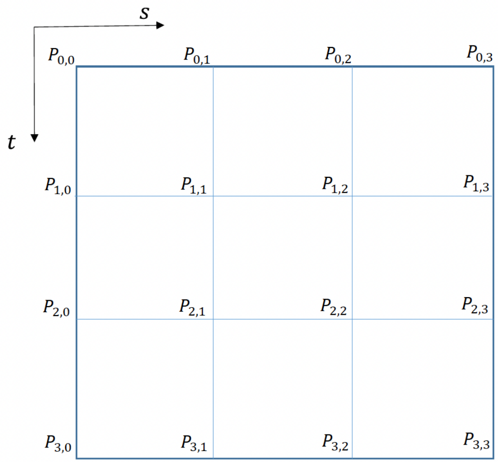
(a) Bajo nivel de detalle (LOD)



(b) Alto nivel de detalle (LOD)

Figura 7: Un mario compuesto por triángulos.

Superficies con Bézier



- Definir matriz de puntos de control
- Por cada posible valor del parámetro s, calcular una curva en la dirección de t
- Para un determinado par de parámetros (s, t) se calcula un punto 3D mediante la extensión de la definición de curvas

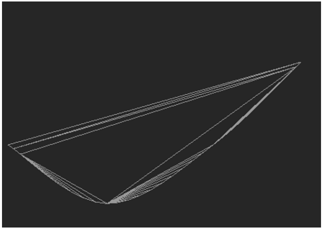
$$\mathrm{p}(u, v) = \sum_{i=0}^{3} \sum_{j=0}^{3} B_i^3(u) B_j^3(v) P_{i,j}$$

donde

$$B_i^3(u) = \binom{3}{i} u^i (1-u)^{n-i}$$

Superficies con Catmull-Rom

Ejemplo con curvas de Catmull-Rom: Generando la superficie triangulando el plano que generen estos puntos. Se puede usar triangulación de Delaunay u otros.



(a) Wireframe de una superficie con Catmull-Rom

(b) Superficie creada

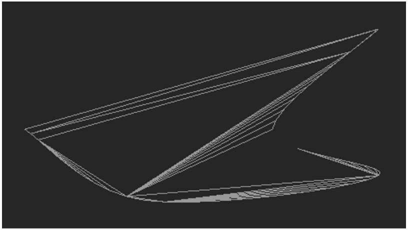
Figura 8: Superficie con Catmull-Rom.

Superficies con Catmull-Rom no convexas

Ejemplo con curvas de Catmull-Rom: Generando la superficie triangulando el plano que generen estos puntos. ¿Qué pasa en el caso de figuras no convexas?

Superficies con Catmull-Rom no convexas

Ejemplo con curvas de Catmull-Rom: Generando la superficie triangulando el plano que generen estos puntos. ¿Qué pasa en el caso de figuras no convexas?



(a) Wireframe superficie no convexa

(b) Superficie creada

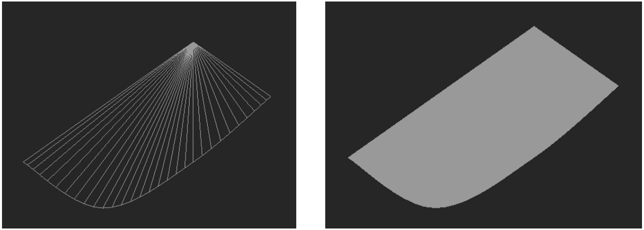
Figura 9: Superficie con Catmull-Rom no conexa.

Superficies con Catmull-Rom, conectadas por un centro

Ejemplo con curvas de Catmull-Rom: Generando la superficie conectando cada par de puntos P_i, P_{i+1} con un centro C_i, así cada triángulo queda definido por (P_i, P_{i+1}, C_i).

Superficies con Catmull-Rom, conectadas por un centro

Ejemplo con curvas de Catmull-Rom: Generando la superficie conectando cada par de puntos P_i, P_{i+1} con un centro C_i, así cada triángulo queda definido por (P_i, P_{i+1}, C_i).



(a) Wireframe de una superficie con Catmull-Rom

(b) Superficie creada

Figura 10: Superficie con Catmull-Rom y centro.

Superficies con sólidos de revolución

Misma metodología:

1. Generar los vértices $v_i = (x_i, y_i, z_i)$
2. Conectar los vértices generados de manera ordenada usando triángulos y agrupándolos en Shapes

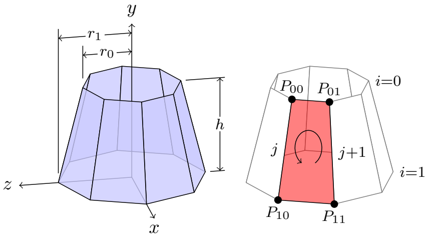


Figura 11: Creando un cono, usando sólidos de revolución.

Gracias por su atención