Universidad Distrital Francisco Jose de Caldas

Facultad de Ingeniería

Correccion del parcial: Curvas de Bézier en Raspberry Pi Pico W

Jhon Alexsander Moreno Velasquez
Código: 20202005031
Asignatura: Sistemas Embebidos
Docente: Gerardo Muñoz

5 de mayo de 2025

Abstract

Este informe documenta el desarrollo de una función en MicroPython para calcular curvas de Bézier cúbicas (de cuatro puntos) utilizando una Raspberry Pi Pico W. Dada la naturaleza del entorno embebido, se emplea el acceso a memoria con `machine.mem32` y `uctypes` para trabajar directamente con estructuras de datos en memoria. Se describe la formulación matemática utilizada, así como una explicación detallada del código desarrollado.

1 Resumen del Problema

Se requiere generar una función que reciba cuatro puntos en $\mathbb{R}^2$ y una constante $n \in \mathbb{R}$, retornando $n+1$ puntos de la curva de Bézier cúbica que describe la trayectoria interpolada por dichos puntos. El objetivo es implementarlo en una Raspberry Pi Pico W utilizando MicroPython, haciendo uso de operaciones de bajo nivel (acceso directo a memoria).

2 Formulación Matemática

La curva de Bézier cúbica para cuatro puntos P_0, P_1, P_2 y P_3 se define como:

$$\mathbf{B}(t) = (1-t)^3\mathbf{P}_0 + 3(1-t)^2 t\mathbf{P}_1 + 3(1-t)t^2\mathbf{P}_2 + t^3\mathbf{P}_3, \quad t \in [0,1] \tag{1}$$

Donde cada $\mathbf{P}_i = (x_i, y_i)$ es un punto en el plano.

3 Explicación Detallada del Código

A continuación, se describe cada fragmento relevante del código para el cálculo de curvas de Bézier cúbicas en un entorno de microcontrolador.

Importación de librerías

```
import array
import machine
import uctypes
```

Se importan las librerías necesarias para trabajar con arreglos tipados (`array`), acceso a registros de memoria de 32 bits (`machine.mem32`), y manejo de direcciones de memoria (`uctypes`).

Definición de la función

```
def curva_bezier_4_puntos(p0, p1, p2, p3, n):
    LR = []
```

La función `curva_bezier_4_puntos` recibe cuatro puntos en $\mathbb{R}^2$ y una constante entera `n`, devolviendo `n+1` puntos interpolados. Se inicializa la lista `LR` donde se almacenarán los puntos generados.

Acceso a memoria de los puntos

```
    dir_p0 = uctypes.addressof(p0)
    dir_p1 = uctypes.addressof(p1)
    dir_p2 = uctypes.addressof(p2)
    dir_p3 = uctypes.addressof(p3)
```

Con `uctypes.addressof` se obtiene la dirección de memoria base de cada punto. Dado que los puntos son arreglos tipo `array.array('l')`, cada coordenada ocupa 4 bytes.

Iteración y cálculo de puntos Bézier

```
    for i in range(n + 1):
        t = i / n
```

Se itera desde 0 hasta `n`, calculando el valor de `t` entre 0 y 1, para obtener los puntos distribuidos uniformemente a lo largo de la curva.

Fórmula de Bézier cúbica aplicada a x e y

```
        Bx = ((1 - t) ** 3) * machine.mem32[dir_p0] + \
             3 * ((1 - t) ** 2) * t * machine.mem32[dir_p1] + \
             3 * (1 - t) * (t ** 2) * machine.mem32[dir_p2] + \
             (t ** 3) * machine.mem32[dir_p3]
```

Se calcula la componente `x` usando los valores almacenados en `mem32`, que acceden directamente a la posición de memoria donde se guarda cada coordenada x de los puntos de control.

```
        By = ((1 - t) ** 3) * machine.mem32[dir_p0 + 4] + \
             3 * ((1 - t) ** 2) * t * machine.mem32[dir_p1 + 4] + \
             3 * (1 - t) * (t ** 2) * machine.mem32[dir_p2 + 4] + \
             (t ** 3) * machine.mem32[dir_p3 + 4]
```

La componente `y` se accede desde la dirección base + 4 bytes, ya que cada entero ocupa 4 bytes en memoria.

Almacenamiento del punto y retorno

```
        LR.append([Bx, By])
    return LR
```

Cada punto interpolado se agrega a la lista `LR`. Finalmente, la función retorna esta lista.

4 Conclusiones

Este código permite generar curvas suaves mediante interpolación de puntos en MicroPython sobre una Raspberry Pi Pico W. El uso de memoria directa permite optimización en velocidad y espacio, lo cual es ideal en sistemas embebidos. La representación matemática y la implementación se corresponden de manera precisa, haciendo de esta solución una alternativa eficiente para aplicaciones gráficas o de control de trayectoria en dispositivos de bajo costo.

5 Referencias

- Raspberry Pi Pico W Documentation. Raspberry Pi Ltd. https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html
- MicroPython Documentation. https://docs.micropython.org/en/latest/
- Bézier Curves. Paul Bourke. http://paulbourke.net/geometry/bezier/
- Python array module. Python Software Foundation. https://docs.python.org/3/library/array.html