UNIVERSIDADE FEDERAL DE RORAIMA CENTRO DE CIÊNCIA E TECNOLOGIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
DCC703 - COMPUTAÇÃO GRÁFICA (2024.2)
Prof. LUCIANO FERREIRA SILVA

Nome: Marcia Gabrielle Bonifácio De Oliveira - 2020011319

TRABALHO 4

1. INTRODUÇÃO

A Curva de Bézier é amplamente utilizada em computação gráfica, animações, design gráfico e modelagem de superfícies. Essa curva é definida por um conjunto de pontos de controle, que influenciam sua forma. O objetivo deste relatório é construir e comparar duas formas diferentes de gerar a curva de Bézier:

- Equação Paramétrica
- Algoritmo de Casteljau

2. CONCEITOS BÁSICOS

2.1 CURVAS DE BÉZIER

A Curva de Bézier é definida pela seguinte equação, baseada nos Polinômios de Bernstein:

$$B(t) = \sum_{i=0}^{n} P_i \binom{n}{i} (1-t)^{n-i} t^i$$

Onde:

- B(t) é o ponto na curva para um valor ttt.
- Pi são os **pontos de controle**.
- (ni) é o **coeficiente binomial**.
- t varia de 0 a 1.

2.2 EQUAÇÃO PARAMÉTRICA DA CURVA DE BÉZIER

A equação paramétrica calcula diretamente os pontos da curva de Bézier a partir dos pontos de controle. O grau da curva é determinado pelo número de pontos de controle:

- **Grau 1** → Linha reta (Interpolação linear).
- **Grau 2** → Curva Quadrática.
- **Grau 3** → Curva Cúbica.
- **Grau n** → Curva de Bézier de ordem superior.

Curva de Bézier Quadrática (Grau 2)

Uma curva de Bézier quadrática é definida por três pontos de controle: P0, P1, P2.

$$C(t) = (1-t)^2 P_0 + 2t(1-t) P_1 + t^2 P_2$$

- Possui um ponto intermediário de controle P1 que atrai a curva.
- Muito utilizada em tipografia e modelagem de fontes.

Curva de Bézier Cúbica (Grau 3)

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

- Mais flexível que a quadrática.
- Padrão na computação gráfica (exemplo: SVG, fontes TrueType).
- Utilizada em curvas suaves para gráficos vetoriais e animações.

3. Implementação em Python

3.1 CÓDIGO PARA A EQUAÇÃO PARAMÉTRICA

```python
# Funções para calcular pontos intermediários da curva de Bézier
def bezier_quad(P0, P1, P2, t):
    """Curva de Bézier Quadrática"""
    return (1 - t) ** 2 * P0 + 2 * (1 - t) * t * P1 + t ** 2 * P2

def bezier_cubic(P0, P1, P2, P3, t):
    """Curva de Bézier Cúbica"""
    return (1 - t) ** 3 * P0 + 3 * (1 - t) ** 2 * t * P1 + 3 * (1 - t) * t ** 2 * P2 + t ** 3 * P3
```

3.2 CÓDIGO PARA O ALGORITMO DE CASTELJAU

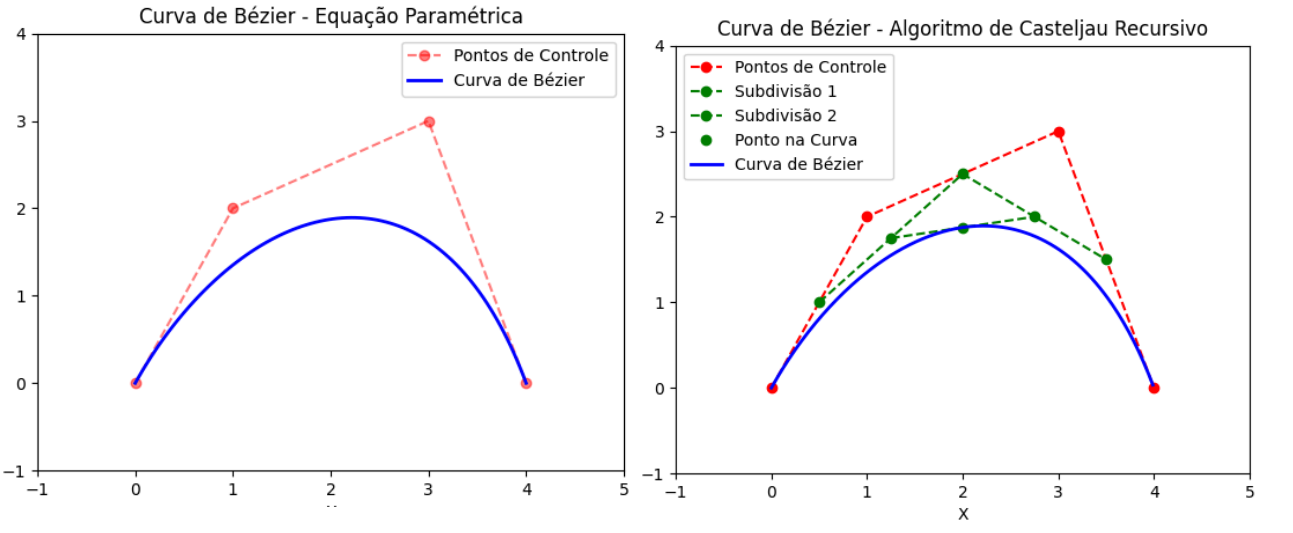
```python
def PMCurva(P0, P1, P2, P3):
    """
    Calcula os pontos intermediários da curva de Bézier usando o método de De Casteljau.
    Retorna os novos pontos para subdivisão.
    """
    # Primeira subdivisão
    P0xN1 = (P0 + P1) / 2
    P1xN1 = (P1 + P2) / 2
    P2xN1 = (P2 + P3) / 2

    # Segunda subdivisão
    P0xN2 = (P0xN1 + P1xN1) / 2
    P1xN2 = (P1xN1 + P2xN1) / 2

    # Terceira subdivisão (ponto na curva)
    P0xN3 = (P0xN2 + P1xN2) / 2

    return P0xN1, P1xN1, P2xN1, P0xN2, P1xN2, P0xN3
```

```python
def casteljau_recursive(P0, P1, P2, P3, t, curve_points):
    """
    Algoritmo de De Casteljau recursivo para calcular pontos da curva de Bézier.
    """
    if t > 0.005:
        e = t / 2
        P0xN1, P1xN1, P2xN1, P0xN2, P1xN2, P0xN3 = PMCurva(P0, P1, P2, P3)
        casteljau_recursive(P0, P0xN1, P0xN2, P0xN3, e, curve_points)
        casteljau_recursive(P0xN3, P1xN2, P2xN1, P3, e, curve_points)
    else:
        curve_points.append(P0)
```

4. RESULTADOS COMPARATIVOS

A curva de Bézier cúbica foi definida com quatro pontos de controle: P0,P1,P2,P3.

```
P0 = np.array([0, 0])
P1 = np.array([1, 3])
P2 = np.array([3, 3])
P3 = np.array([4, 0])
```



5. CONCLUSÃO

- Ambos os métodos geram a mesma curva de Bézier.
- O Casteljau é amplamente utilizado em gráficos computacionais devido à sua robustez.