PODER EXECUTIVO
MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE RORAIMA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

DCC703 - COMPUTAÇÃO GRÁFICA

RELATÓRIO DO PROJETO DE BÉZIER

ALUNOS:
Marcos Vinícius Tenacol Coêlho - 2021000759

Março de 2025
Boa Vista/Roraima

Resumo

A implementação apresentada tem como objetivo desenhar curvas de Bézier.

O programa permite ao usuário adicionar pontos de controle clicando na tela e visualizar a curva gerada por dois métodos diferentes: a equação paramétrica e o algoritmo de De Casteljau.

O programa define duas abordagens principais para gerar a curva de Bézier:

Equação Paramétrica: Utiliza o polinômio de Bernstein para calcular os pontos da curva.

Algoritmo de De Casteljau: Um método recursivo que divide os segmentos até convergir para um ponto na curva.

Estrutura do Código

Equação paramétrica:

polinomio_bernstein: Calcula o polinômio de Bernstein para um determinado grau e parâmetro t.

curva_bezier_parametrica: Gera pontos da curva de Bézier usando a equação paramétrica.

Castejau:

casteljau: Implementa o algoritmo de De Casteljau para calcular um ponto na curva.

curva_bezier_casteljau: Utiliza De Casteljau para gerar uma curva com um determinado número de pontos.

Como funciona a equação paramétrica?

A equação paramétrica de Bézier utiliza a combinação linear dos pontos de controle, ponderados pelo polinômio de Bernstein.
A implementação do código gera a curva variando em um intervalo e somando os produtos dos pontos de controle com os coeficientes de Bernstein.

Para um conjunto de pontos de controle, a curva é definida por:

$$C(t) = \sum_{i=0}^{n} B_{i,\,n}(t).P_i$$

Como funciona o algoritmo de De Casteljau?

O algoritmo de De Casteljau é um método recursivo que calcula um ponto da curva de Bézier dividindo iterativamente os segmentos formados pelos pontos de controle. Para um dado valor de , os novos pontos intermediários são calculados.

O processo continua até restar apenas um ponto, que pertence à curva.

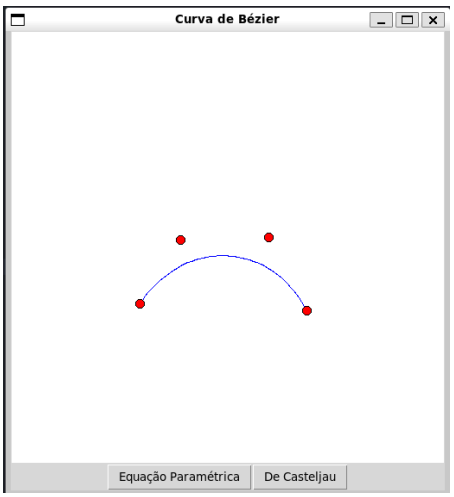
Implementação e testes

Equação paramétrica:

```
def polinomio_bernstein(n, i, t):
    return (np.math.comb(n, i)) * (t ** i) * ((1 - t) ** (n - i))
def curva_bezier_parametrica(pontos_controle, num_pontos=50):
    n = len(pontos_controle) - 1
    t_values = np.linspace(0, 1, num_pontos)
    curva = np.zeros((num_pontos, 2))

    for i in range(n + 1):
         curva += np.array(pontos_controle[i]) * polinomio_bernstein(n,
i, t_values)[:, np.newaxis]
    return curva
```

Curva de bézier utilizando equação paramétrica, com os pontos nas coordenadas: (1.49, 3.16), (1.96, 2.42), (2.98, 2.39), (3.42, 3.24):



Castejau:

```
def casteljau(pontos, t):
    pontos_temp = np.array(pontos, dtype=float)
    n = len(pontos) - 1

    for r in range(1, n + 1):
        for i in range(n - r + 1):
                    pontos_temp[i] = (1 - t) * pontos_temp[i] + t *
pontos_temp[i + 1]
    return pontos_temp[0]

def curva_bezier_casteljau(pontos_controle, num_pontos=4):
    t_values = np.linspace(0, 1, num_pontos)
    return np.array([casteljau(pontos_controle, t) for t in t_values])
```

Curva de bézier utilizando casteljau, com os pontos nas coordenadas:
(1.35, 2.91), (1.88, 1.9), (3.2, 1.91), (3.54, 2.99):