Curvas de Bézier

DCC703 - Computação Gráfica (2024.2)
Prof. - Luciano Ferreira Silva
Aluno - Paulo Ferreira da Silva Júnior - 2019034400

Introdução

A rasterização de curvas de Bézier é amplamente utilizada em computação gráfica para representar curvas suaves através de um conjunto de pontos de controle. Existem diferentes abordagens para a construção dessas curvas, sendo as mais comuns:

1. **Equação Paramétrica**
2. **Algoritmo de De Casteljau**

Neste relatório, exploramos ambos os algoritmos, implementamos e analisamos seus desempenhos e diferenças fundamentais.

1. Equação Paramétrica

Descrição do Algoritmo

A equação paramétrica define a curva de Bézier a partir do polinômio de Bernstein. Para uma curva de Bézier cóbica (grau 3), a fórmula é:

B(t) = (1−t)^3 P0+3(1−t)t^2 P1+3(1−t)t^2 P2+t^3 P3

onde P0,P1,P2,P3 são os pontos de controle e tt varia entre 0 e 1.

Esse método permite calcular diretamente os pontos da curva sem subdivisão iterativa, tornando-se computacionalmente eficiente.

Código Implementado

```
def bernstein(n, i, t):
    """ Calcula o polinômio de Bernstein B_i^n (t) """
    return comb(n, i) * (t ** i) * ((1 - t) ** (n - i))

def bezier_generalized(t, P):
    """ Calcula pontos da curva Bézier de grau n usando a equação paramét
rica """
    n = len(P) - 1  # Grau da curva (n = número de pontos - 1)
    B = np.zeros(2)  # Inicializa coordenadas X e Y

    for i in range(n + 1):  # Para cada ponto de controle
        B += bernstein(n, i, t) * P[i]  # Soma as contribuições dos Bernstein

    return B
```

Resultado

attachment:f17e08d4-b0f0-4ade-94f7-0efe56fca456:2025-03-03-17-11-34.mp4

2. Algoritmo de De Casteljau

Descrição do Algoritmo

O algoritmo de De Casteljau constrói a curva de forma recursiva através da interpolação linear entre os pontos de controle. Ele segue o seguinte processo:

1. Para um dado valor de t, interpola-se linearmente entre os pontos de controle.
2. O processo é repetido até restar um único ponto, que pertencerá à curva.

A fórmula iterativa do algoritmo é:

$$P_i^{(k)}(t) = (1-t)P_i^{(k-1)} + tP_{i+1}^{(k-1)}$$

Código Implementado

```
def casteljau_recursive(points, t):
    """ Calcula um ponto na curva Bézier usando o algoritmo de Casteljau de forma recursiva. """
    if len(points) == 1:
        return points[0]

    new_points = [(1 - t) * points[i] + t * points[i + 1] for i in range(len(points) - 1)]
    return casteljau_recursive(new_points, t)

def bezier_casteljau(points, num_samples=100):
    """ Gera pontos da curva Bézier usando o algoritmo de Casteljau. """
    return np.array([casteljau_recursive(points, t) for t in np.linspace(0, 1, num_samples)])
```

Resultado

attachment:85946d78-908c-486f-97cf-cd484c3548c1:2025-03-03-17-32-29.mp4

Comparativo Geral

Método	Eficiência Computacional	Precisão
Equação Paramétrica	Moderada	Moderada
De Casteljau	Alta	Alta

A equação paramétrica permite calcular diretamente os pontos da curva, sendo mais eficiente para curvas de grau fixo. Por outro lado, o algoritmo de De Casteljau é mais flexível e pode ser aplicado a curvas de qualquer grau, sendo mais intuitivo para manipulação interativa.

Conclusão

Os dois algoritmos são eficazes para gerar curvas de Bézier, cada um com suas vantagens. O método paramétrico é mais rápido para renderização fixa, enquanto De Casteljau é mais flexível para manipulação interativa. A escolha entre os dois depende da aplicação desejada.