UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
COLEGIADO DE CIÊNCIA DA COMPUTAÇÃO

Relatório da Aula Prática 01 – Computação Gráfica

Alunos: Fabio Kenji Sato
João Leonardo Lívero Lavaqui
Vinicius Mattos Marcos

---

1. INTRODUÇÃO

A superfície de Bézier é uma superfície 3D, curvada e suave, definida matematicamente por 16 pontos de controle, os quais definem o seu formato.

Sua montagem e processo se utiliza de diversas curvas de Bézier misturadas para definir as coordenadas dos pontos em sua superfície, utilizando a modelagem de matriz para curvas de Bézier, com a matriz de mistura, pontos de controle organizados em uma matriz e um valor T de fator interpolador.

2. METODOLOGIA

A matriz de mistura, responsável por definir a relação entre os pontos de controle e os pontos da superfície, está representada no código por uma matriz 4×4 chamada **matriz_bezier**. Ela contém os coeficientes padrão da base de Bézier, o que permite seu uso direto em multiplicações matriciais, sem a necessidade de dividi-la em partes menores. No código, essa multiplicação é realizada com o operador **@**, que representa a multiplicação de matrizes em Python.

Os elementos da geometria no contexto do código fornecido são os pontos de controle fundamentais para definir a forma e a curvatura da superfície de Bézier. Estes pontos estão organizados em uma estrutura de dados específica no código, denominada **elementos_geometria**, que é uma matriz tridimensional com dimensões 4×4×3 . Cada uma das 16 posições (i, j) dentro desta matriz armazena um ponto de controle tridimensional, ou seja, um conjunto completo de coordenadas (x, y, z). A capacidade de manipular diretamente os valores destas coordenadas dentro da matriz **elementos_geometria** oferece um controle preciso sobre a geometria e o formato final da superfície de Bézier que será gerada. Ao ajustar a posição de um ou mais desses pontos de controle, é possível deformar e remodelar a superfície de maneira intuitiva, o que é uma característica poderosa da modelagem com superfícies de Bézier. Esses 16 pontos agem como "âncoras" que influenciam a superfície, mas a superfície não passa necessariamente por todos eles, exceto nos seus cantos (ponto inicial e ponto final). Eles estabelecem um "esqueleto" que guia a curvatura e a suavidade da forma 3D resultante.

A função T(t), definida no código como **def T(t)**, desempenha um papel paramétrico central na geração da superfície de Bézier. Ela representa um vetor de potências do parâmetro t, essencial para a parametrização suave da superfície. Para a construção de uma superfície de Bézier bidimensional, como a que está sendo modelada, são necessários dois parâmetros independentes, **u** e **v**, cada um variando tipicamente de **0** a **1**. Consequentemente, a função **T(t)** é utilizada para gerar dois vetores distintos: **T(u)** (referenciado como **Tu** no laço do código) e **T(v)** (referenciado como **Tv**). A importância de **T(t)** é evidente na fórmula fundamental para o cálculo de cada ponto da superfície de Bézier.

Para calcular cada coordenada (x, y, e z) de um ponto específico na superfície, utiliza-se a seguinte multiplicação matricial:

$$\text{ponto}[\text{coord}] = T(u) \cdot M_B \cdot G_{\text{coord}} \cdot M_B^T \cdot T(v)^T$$

Nesta expressão, $\mathbf{M_B}$ é a **matriz_bezier**, uma matriz constante 4×4 que contém os coeficientes polinomiais da base de Bézier para curvas cúbicas, e $\mathbf{G_{coord}}$ é a submatriz dos pontos de controle para a coordenada específica que está sendo calculada (extraída de **elementos_geometria**). A execução desta operação para cada par de (u, v) no domínio dos parâmetros resulta em uma coleção de pontos tridimensionais que, quando plotados num gráfico 3D, formam a superfície de Bézier.

3. RESULTADOS

Os resultados obtidos foram plotados utilizando a biblioteca ***Matplotlib*** da linguagem Python. A exibição dos gráficos foram feitas da seguinte forma.

Primeiro criamos Os vetores **pontos_x**, **pontos_y** e **pontos_z** armazenam as coordenadas 3D (x, y, z) dos pontos que compõem a superfície de Bézier construída a partir dos pontos de controle da matriz **elementos_geometria**.

```python
# Lista para armazenar os pontos
pontos_x, pontos_y, pontos_z = [], [], []

for uu in u:
    for vv in v:
        Tu = T(uu)
        Tv = T(vv)
        ponto = np.zeros(3)
        for coord in range(3):
            G_coord = elementos_geometria[:, :, coord]
            ponto[coord] = Tu @ matriz_bezier @ G_coord @ matriz_bezier.T @ Tv.T
        pontos_x.append(ponto[0])
        pontos_y.append(ponto[1])
        pontos_z.append(ponto[2])
```

Em seguida, utilizamos os pontos calculados da superfície (armazenados nos vetores **pontos_x**, **pontos_y** e **pontos_z**) junto com os pontos de controle da matriz **elementos_geometria** para plotar o gráfico 3D, destacando tanto a forma da superfície quanto a malha que a define.

```python
# Plot 3D apenas com pontos
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.scatter(pontos_x, pontos_y, pontos_z, color='blue', s=5, label='Pontos da superfície')

# Pontos de controle em vermelho
for i in range(4):
    for j in range(4):
        x, y, z = elementos_geometria[i, j]
        ax.scatter(x, y, z, color='red', s=15)

ax.set_title('Superfície de Bézier (pontos)')
ax.legend()
plt.show()
```

Esses são os passos para podermos ver o resultado entre as operações das matrizes que geram a superfície de Bézier.

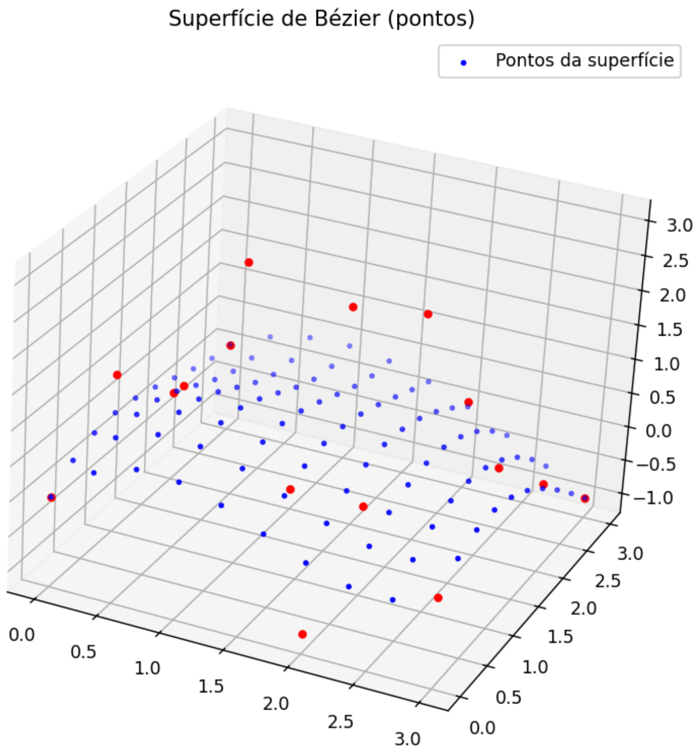
Agora, ao ajustar os valores da matriz tridimensional **elementos_geometria**, responsável pelos pontos de controle da superfície, e a variável **step**, que define a densidade de pontos ao longo da superfície, é possível modificar tanto o formato quanto o nível de detalhamento da superfície de Bézier, como ilustrado na imagem abaixo.

```python
# Matriz base de Bézier
matriz_bezier = np.array([
    [-1, 3, -3, 1],
    [3, -6, 3, 0],
    [-3, 3, 0, 0],
    [1, 0, 0, 0]
])

# Pontos de controle (4x4x3)
elementos_geometria = np.array([
    [[0, 0, 0], [1, 0, 2], [2, 0, -1], [3, 0, 0]],
    [[0, 1, 1], [1, 1, 3], [2, 1, 0], [3, 1, 1]],
    [[0, 2, 0], [1, 2, -1], [2, 2, 2], [3, 2, 0]],
    [[0, 3, 0], [1, 3, 1], [2, 3, 0], [3, 3, -1]],
])

# Resolução
step = 0.1
res = int(1 / step)
u = np.linspace(0, 1, res)
v = np.linspace(0, 1, res)
```

Obtemos o seguinte resultado:



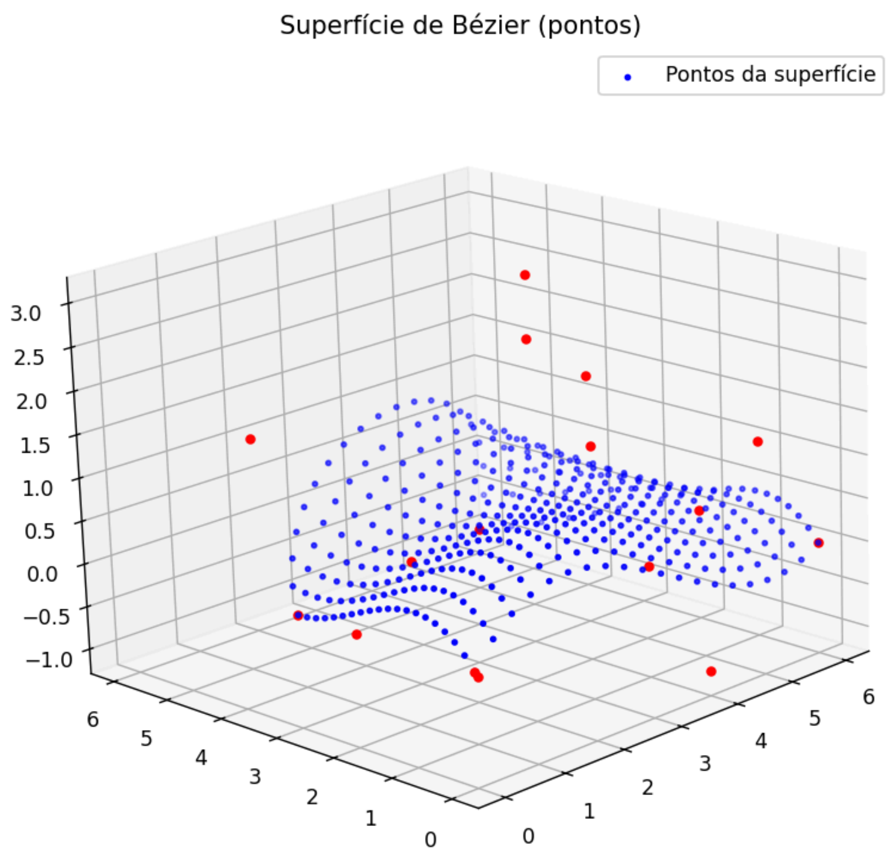
Apenas alterando a matriz do ponto de controle, podemos manipular de diversas formas a superfície. Na imagem a seguir foi feito exatamente isso, apenas a matriz de que têm os pontos de controle e o número do passo, que anteriormente era 0,1, agora está em 0,05.

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

# Pontos de controle (4x4x3)
elementos_geometria = np.array([
    [[0, 0, 0], [2, 0, 2], [4, 0, -1], [6, 0, 0]],
    [[0, 1, 1], [2, 1, 3], [4, 1, 0], [6, 1, 1]],
    [[0, 2, 0], [2, 2, -1], [4, 2, 2], [6, 2, 0]],
    [[0, 3, 0], [2, 6, 1], [4, 3, 3], [6, 6, -1]],
])

# Resolução
step = 0.05
res = int(1 / step)
u = np.linspace(0, 1, res)
v = np.linspace(0, 1, res)
```

Como é possível notar na ilustração a seguir, a malha está bem mais visível devido à alteração no número do passo, resultando num número maior de pontos (azuis).



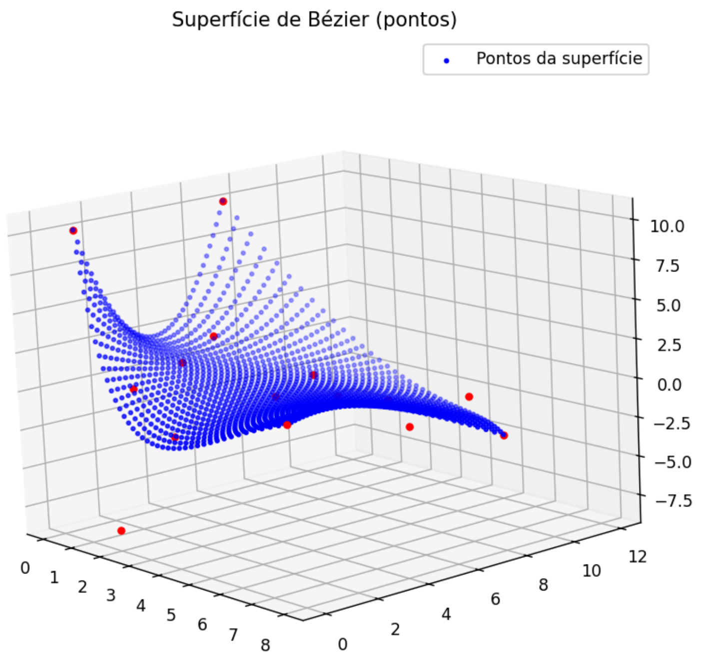
Para um último exemplo, novamente foram alteradas apenas a matriz com os pontos de controle e o número do passo foi reduzido.

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

# Pontos de controle (4x4x3)
elementos_geometria = np.array([
    [[0, 1, 10], [2, 0, -8], [4, 0, -1], [6, 2, 0]],
    [[0, 3, -1], [2, 4, 3], [4, 4, 0], [6, 4, 1]],
    [[0, 5, 0], [2, 8, -1], [4, 8, -2], [6, 6, 0]],
    [[0, 7, 10], [2, 12, -6], [4, 12, -3], [8, 8, -2]],
])

# Resolução
step = 0.025
res = int(1 / step)
u = np.linspace(0, 1, res)
v = np.linspace(0, 1, res)
```

Por fim, obtemos este resultado:



Nota-se que o número de pontos é muito superior que os exemplos anteriores.