DOCUMENTO TÉCNICO – Arquitetura, Algoritmos e Decisões de Projeto

1. Visão Geral da Aplicação

A aplicação é um **web app interativo de Computação Gráfica**, implementado em HTML5, CSS, JavaScript e Three.js, contendo quatro módulos principais:

1. **Curvas de Bézier (2D)**
2. **Curvas B-Spline (2D)**
3. **Superfície de Revolução (3D)**
4. **Voo de Alus – animação 3D baseada em Fibonacci**

Cada módulo é totalmente interativo, permitindo adição de pontos, arrasto, configuração de parâmetros e exportação de dados (JSON ou OBJ).

---

2. Arquitetura da Aplicação

A aplicação é implementada em **uma única página (SPA)** composta por:

2.1. Estrutura HTML

- Quatro *tabs* independentes, cada uma responsável por um módulo.
- Canvases 2D renderizados via **CanvasRenderingContext2D**.
- Canvases 3D criados dinamicamente via **Three.js (WebGL)**.
- Controles HTML para ajuste de parâmetros.

2.2. Organização do Código JavaScript

O arquivo concentra a lógica em blocos:

Módulo	Principais Variáveis	Principais Funções
Bézier	`bezierPoints`, `bezierWeights`	`deCasteljau()`, `drawBezier()`
B-Spline	`splinePoints`, `basisFunction()`	`drawSpline()`, `evaluateBSpline()`

Revolução 3D	`profilePoints`, `mesh3d`	`generateSurface()`, `initScene3D()`
Alus 3D	`alusCurve`, `alusBird`	`generateAlusTrajectory()`, `animateAlus()`

2.3. Biblioteca externa

- **Three.js r128**, carregada via CDN.
- Usada para:
    - Gerar superfícies 3D
    - Renderizar animações
    - Luzes e câmeras
    - Criação e manipulação de mesh

2.4. Componentes Interativos

- Eventos de mouse:
    - `mousedown`, `mousemove`, `mouseup` para arrastar pontos.
- Controles de interface:
    - *sliders*, *select*, *buttons*.
- Atualização reativa:
    - Cada mudança de parâmetro redesenha a curva imediatamente.

---

3. Algoritmos Utilizados

A seguir estão todos os algoritmos implementados e sua função dentro da aplicação.

3.1. Curvas de Bézier – Algoritmo de De Casteljau

O sistema usa a formulação de **Bézier racional**, já que incorpora pesos.

Função central

function deCasteljau(points, weights, t)

Características

- Calcula pontos intermediários por interpolação linear recursiva.
- Permite manipulação de *N* pontos arbitrários.
- Suporta pesos individuais → curvas racionais.

3.2. Curvas B-Spline – Função de Base de Cox–de Boor

A B-Spline utiliza o algoritmo clássico baseado na recursão:

1. Cálculo do vetor de nós

generateKnotVector(n, p)

2. Função base (Cox–de Boor)

basisFunction(i, p, u, knots)

3. Avaliação da curva

evaluateBSpline(points, p, u, knots)

Propriedades obtidas

- Continuidade **C²** (exceto nos extremos)
- Controle local
- Curvas suaves com grau configurável (2, 3 ou 4)

3.3. Superfície de Revolução – Amostragem e Geração de Malha

A superfície é gerada pela rotação de um perfil 2D em torno de um eixo.

Passos do algoritmo

1. **Obter curva do perfil** Pode ser Bézier ou B-Spline, usando as mesmas funções do módulo 2D.

2. **Converter coordenadas do canvas → coordenadas 3D**

3. **Rotacionar cada ponto** Para cada subdivisão `i`:

```
angle = i * angleStep
x’ = r * cos(angle)
z’ = r * sin(angle)
```

4. **Construção dos vértices e índices**

vertices.push(...)
indices.push(a, b, c)

5. **Geração da geometria final**

geometry.setAttribute('position', ...)
geometry.setIndex(indices)
geometry.computeVertexNormals()

6. **Material e renderização Three.js**

Exportação OBJ

- A aplicação gera um arquivo `.obj` simples contendo:
    - Lista de vértices (`v`)
    - Lista de faces (`f`)

3.4. Voo de Alus – Trajetória Fibonacci 3D

O pássaro Alus segue uma espiral inspirada na sequência de Fibonacci.

1. Geração da sequência

fibonacci(n)

2. Cálculo da trajetória

generateAlusTrajectory()

Fatores usados

- O raio da espiral cresce proporcionalmente ao termo de Fibonacci atual.
- A altura varia suavemente (“sobe → desce”).
- O ângulo cresce em proporção ao **número áureo (PHI)**:

angleIncrement = PHI * 2π / cycles

3. Suavização com curva Catmull-Rom

new THREE.CatmullRomCurve3(points)

4. Animação frame a frame

animateAlus()

- O pássaro se move ao longo da curva.
- As asas batem com função senoidal.
- A câmera segue suavemente.

---

4. Decisões de Projeto

4.1. Por que uma SPA em um único arquivo?

- Simplifica entrega e execução local.
- Não requer servidor.
- Facilita manutenção para projetos acadêmicos.

4.2. Uso de Canvas 2D + Three.js

- Canvas 2D → operações simples e baratas para curvas.

- Three.js → necessário para:

    - Renderização 3D
    - Cálculo de normais
    - Luz e profundidade
    - Animação complexa

4.3. Separação por módulos (tabs)

- Evita sobrecarga visual.
- Cada módulo é independente em funcionalidade e estado.

4.4. Curvas paramétricas escolhidas

- Bézier → fácil manipulação direta.
- B-Spline → controle avançado, suavidade contínua.
- Catmull-Rom → suavização natural para animação (Alus).

4.5. Interatividade como prioridade

Cada curva é **arrastável, editável, recalculada em tempo real**, oferecendo experiência didática de Computação Gráfica.

4.6. Exportações

- Exportar JSON facilita reutilização dos dados de curvas.

- Exportar OBJ permite uso em Blender e outros softwares 3D.

---

5. Conclusão

A aplicação integra de forma completa:

- **Modelagem paramétrica 2D**
- **Construção de superfícies 3D**
- **Interatividade em tempo real**
- **Animação avançada em Three.js**

Ela demonstra conceitos fundamentais de Computação Gráfica, incluindo interpolação, curvas paramétricas, normalização, transformações 3D e animação.