UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE

Departamento Ciencias de la Computación

INFORME TÉCNICO

Implementación de Algoritmos de Generación de Curvas: Bézier y B-Spline

Autor: Ricardo Laínez Suárez
Asignatura: Computación Gráfica
Docente: Ing. Dario Morales
Fecha: 14 de diciembre de 2025

Quito, Ecuador

Índice

1. Introducción

En el campo de la computación gráfica y el diseño asistido por computadora (CAD), la representación precisa de formas suaves y orgánicas es fundamental. A diferencia de las líneas rectas y polígonos simples, las curvas requieren modelos matemáticos complejos para su definición y renderizado.

Este proyecto aborda la implementación de dos de los algoritmos más influyentes en la historia de los gráficos por ordenador: las **Curvas de Bézier** y las **Curvas B-Spline**. El objetivo central es desarrollar una herramienta de software interactiva que no solo visualice estas curvas, sino que permita al usuario manipular sus puntos de control en tiempo real, demostrando las diferencias matemáticas y funcionales entre ambas aproximaciones. La solución se ha construido bajo estrictos estándares de ingeniería de software, priorizando la modularidad y la robustez mediante el uso de patrones de diseño.

2. Objetivos

2.1. Objetivo General

Implementar una aplicación de computación gráfica funcional y demostrable que permita la generación, visualización y manipulación interactiva de Curvas de Bézier y B-Spline, cumpliendo con estándares de codificación modular y validación de entradas.

2.2. Objetivos Específicos

- **Modelado Matemático:** Implementar computacionalmente las ecuaciones paramétricas que definen las curvas de Bézier (mediante el algoritmo de De Casteljau) y las B-Spline Cúbicas.
- **Arquitectura de Software:** Diseñar el sistema utilizando el *Patrón de Diseño Strategy* para garantizar la modularidad, permitiendo la intercambiabilidad y mantenibilidad de los algoritmos.
- **Interfaz Interactiva:** Desarrollar un lienzo gráfico utilizando GDI+ que permita la manipulación de puntos de control mediante eventos del ratón (Drag & Drop) con redibujado en tiempo real.
- **Robustez:** Implementar mecanismos de validación de entradas y manejo de excepciones para asegurar que la aplicación sea estable ante datos inconsistentes o insuficientes.

3. Materiales y Herramientas

- **Lenguaje de Programación:** C# .NET (versión 10.0 o superior).
- **Entorno de Desarrollo (IDE):** Microsoft Visual Studio 2022.
- **Framework Gráfico:** Windows Forms con librería GDI+ (`System.Drawing`) para el renderizado de primitivas gráficas 2D.

- **Gestión de Memoria:** Uso de `DoubleBuffered` para evitar el parpadeo (*flickering*) en la animación de redibujado.

4. Marco Teórico y Fundamentación Matemática

La generación de curvas en gráficos por ordenador se basa en la **geometría paramétrica**. Una curva en el plano se define como una función vectorial $P(t) = (x(t), y(t))$ donde el parámetro t varía típicamente en el intervalo $[0, 1]$.

4.1. Curvas de Bézier

Las curvas de Bézier son definidas completamente por un conjunto de puntos denominados *Puntos de Control*. Una característica clave es que la curva pasa por el primer y el último punto de control, mientras que los puntos intermedios actúan como ïmanes"que atraen la curva.

4.1.1. Definición Matemática (Polinomios de Bernstein)

Una curva de Bézier de grado n con $n + 1$ puntos de control $P_0, P_1, ..., P_n$ se define como:

$$P(t) = \sum_{i=0}^{n} B_{i,n}(t) \cdot P_i, \quad t \in [0, 1] \tag{1}$$

Donde $B_{i,n}(t)$ son los **polinomios base de Bernstein**, definidos como:

$$B_{i,n}(t) = \binom{n}{i}(1-t)^{n-i}t^i \tag{2}$$

Aquí, $\binom{n}{i}$ representa el coeficiente binomial.

4.1.2. Algoritmo de De Casteljau (Implementación Recursiva)

Para la implementación computacional, especialmente para grados arbitrarios N, el cálculo directo de factoriales es ineficiente e inestable numéricamente. Se utilizó el **Algoritmo de De Casteljau**, que construye la curva mediante interpolaciones lineales recursivas.

Para un conjunto de puntos $P_0, ..., P_n$, el punto en la curva para un valor t se obtiene mediante la recurrencia:

$$P_i^{(k)} = (1-t)P_i^{(k-1)} + tP_{i+1}^{(k-1)} \tag{3}$$

Donde:

- k es el nivel de recursión (de 1 a n).
- $P_i^{(0)}$ son los puntos de control originales.
- $P_0^{(n)}$ es el punto final sobre la curva.

Justificación de elección: Este método fue seleccionado por su estabilidad numérica y su capacidad de generalización para N puntos sin cambiar el código base.

4.2. Curvas B-Spline (Basis Splines)

A diferencia de las curvas de Bézier, donde mover un solo punto de control afecta a **toda** la curva (control global), las B-Spline ofrecen **control local**. La curva se compone de segmentos polinómicos unidos con continuidad C^2 (continuidad de curvatura).

4.2.1. Variante Implementada: B-Spline Cúbica Uniforme

Se implementó la variante **Uniforme Cúbica**, donde el vector de nodos es equidistante. Esto simplifica las funciones base a polinomios precalculados que dependen solo de 4 puntos de control locales $(P_i, P_{i+1}, P_{i+2}, P_{i+3})$ para cada segmento.

Las funciones de mezcla (*Blending Functions*) en su forma matricial para un segmento $t \in [0, 1]$ son:

$$P(t) = \frac{1}{6} \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} P_i \\ P_{i+1} \\ P_{i+2} \\ P_{i+3} \end{bmatrix} \tag{4}$$

Esta formulación matricial permite un cálculo extremadamente rápido y garantiza que la curva sea suave y continua.

5. Implementación y Arquitectura

Para satisfacer los **Requerimientos No Funcionales** de modularidad y mantenimiento, se implementó el **Patrón de Diseño Strategy**.

5.1. Estructura Modular

1. **Interfaz `IEstrategiaCurva`:** Define el contrato que todos los algoritmos deben cumplir. Contiene métodos como `CalcularPuntos` y propiedades como `MinimoPuntosRequeridos`.
2. **Clases Concretas:**
   - `CurvaBezier.cs`: Implementa la lógica recursiva de De Casteljau.
   - `CurvaBSpline.cs`: Implementa la lógica de interpolación cúbica matricial.
3. **Contexto (`Form1.cs`):** La interfaz gráfica no conoce los detalles matemáticos, delegando el cálculo al algoritmo seleccionado en tiempo de ejecución.

5.2. Manejo de Errores y Validaciones

Se implementaron defensas robustas en el código:

- **Validación de Cantidad de Puntos:** El sistema impide dibujar una B-Spline con menos de 4 puntos.
- **Límites de Pantalla:** Se restringe el arrastre de nodos al área visible.
- **Manejo de Excepciones:** Bloques `try-catch` en eventos de renderizado.

6. Desarrollo y Análisis Comparativo

Durante las pruebas de ejecución se observaron los siguientes comportamientos característicos:

Característica	**Curva de Bézier**	**Curva B-Spline (Uniforme)**
Paso por puntos	Pasa por el primer (P_0) y último punto (P_n).	Generalmente no pasa por los puntos de control (es aproximante).
Control	**Global:** Mover un punto afecta toda la curva.	**Local:** Mover un punto solo afecta a los segmentos vecinos.
Complejidad	Polinómica de grado N.	Cúbica (Grado 3) constante.
Suavidad	Depende del grado, puede oscilar.	Continuidad C^2 garantizada.

Tabla 1: Comparativa técnica de algoritmos implementados

7. Conclusiones

1. **Cumplimiento Funcional:** Se logró desarrollar una aplicación robusta que implementa correctamente la matemática detrás de las curvas paramétricas. El algoritmo de De Casteljau demostró ser una solución estable para curvas de Bézier.

2. **Modularidad Exitosa:** La implementación del patrón *Strategy* desacopló la lógica matemática de la interfaz de usuario, facilitando el mantenimiento futuro.

3. **Experiencia de Usuario:** El uso de GDI+ con la técnica de doble buffer (`DoubleBuffered = true`) fue crucial para eliminar el parpadeo durante la interacción.

4. **Validación Matemática:** Se comprobó empíricamente la diferencia entre el control local (B-Spline) y el control global (Bézier), validando la teoría expuesta.

8. Anexos

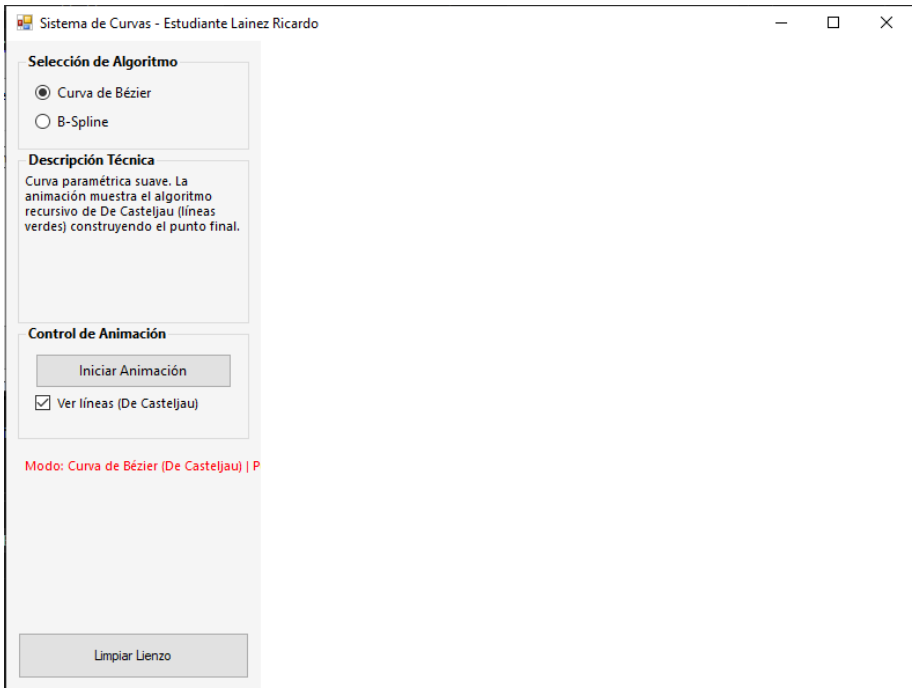

Figura 1: Interfaz del proyecto Curvas de Bezier, B-spline.

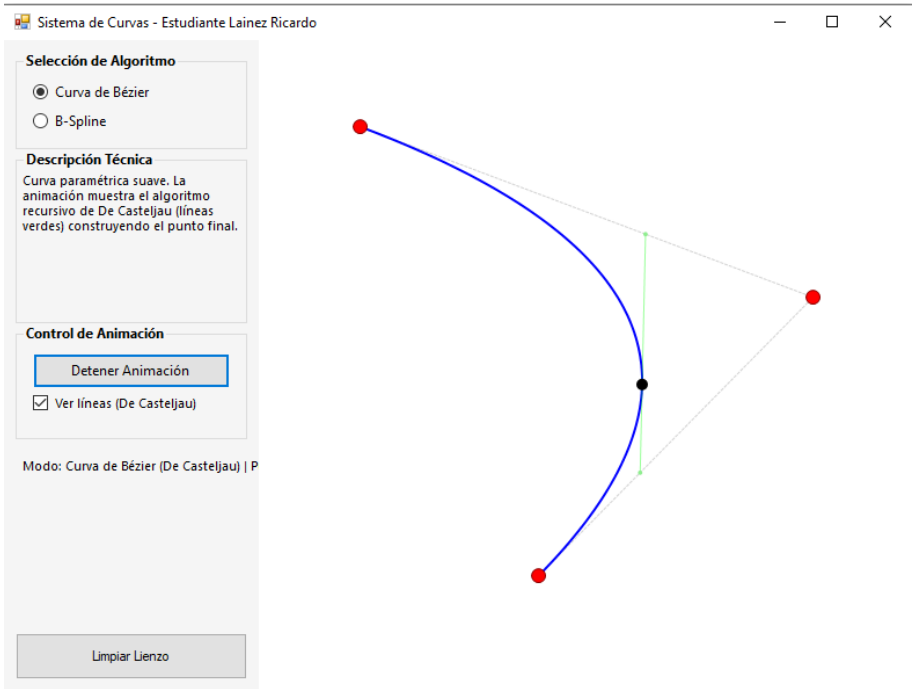

Figura 2: Ejecución de Bezier de grado 3.

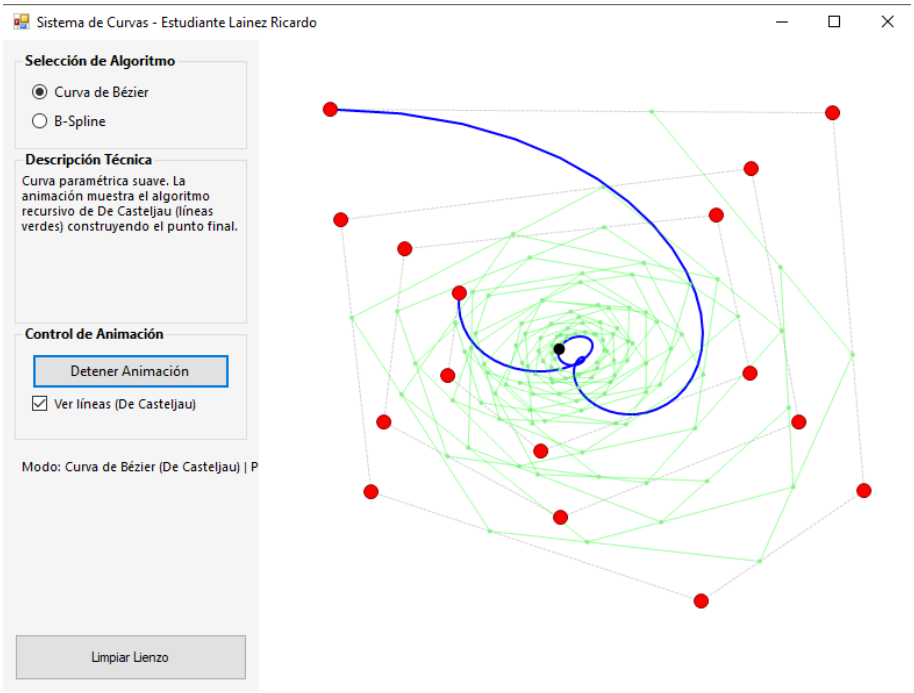

Figura 3: Ejecución de Bezier con grado n.

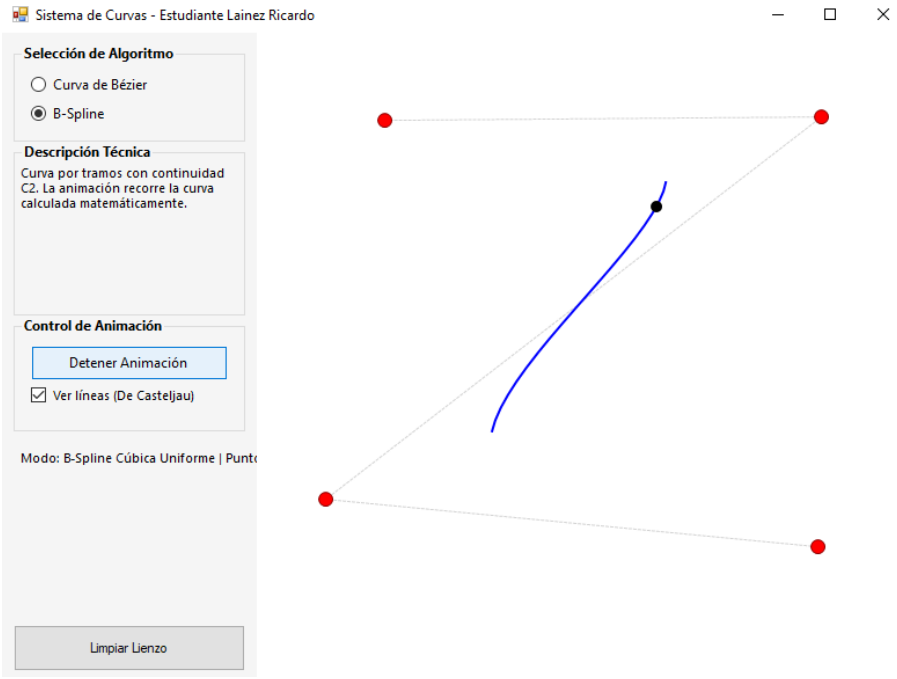

Figura 4: Ejecución de B-spline con 4 puntos.

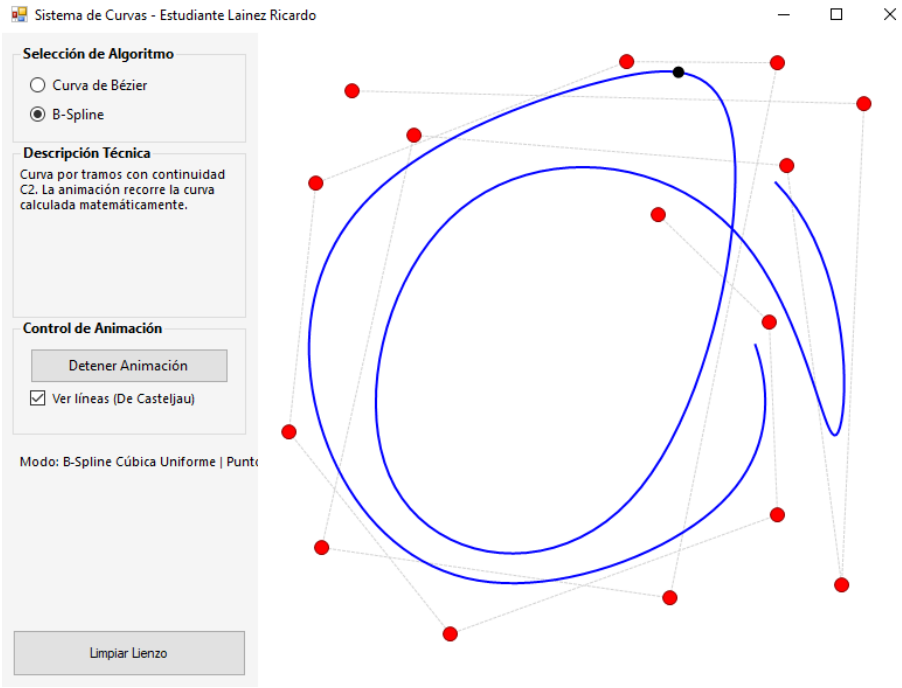

Figura 5: Ejecución de B-spline con n puntos.

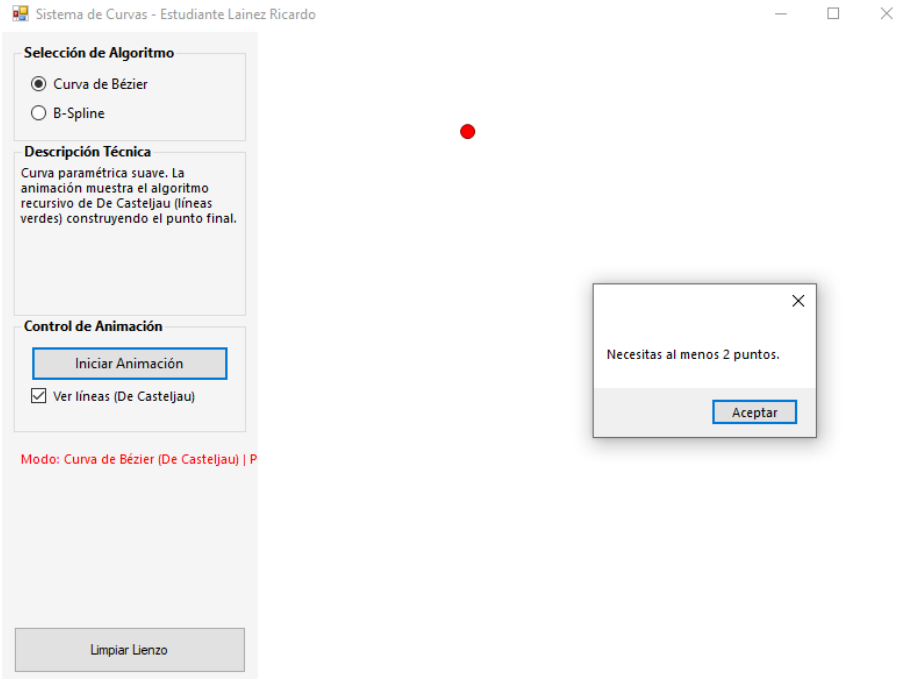

Figura 6: Validación de bezier.

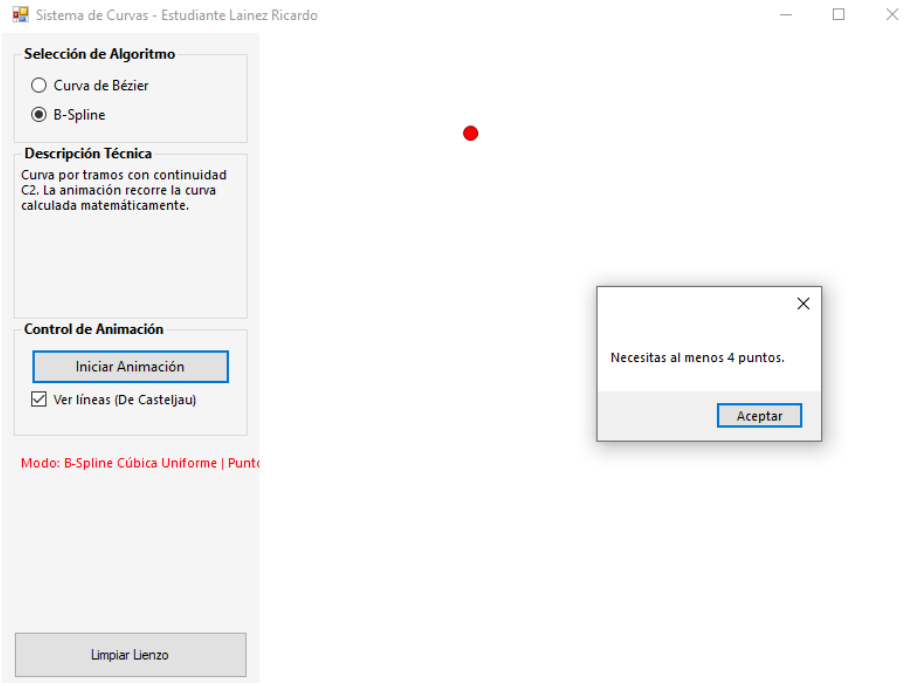

Figura 7: Validación de B-spline.