Gráficos por Computador: Prácticas de Laboratorio

Práctica 2. Modelado Geométrico

El objetivo de la práctica es la construcción de clases que permitan dibujar en pantalla curvas y superficies de Bezier de grado 3.

Práctica 2.1 Visualización de curvas cúbicas de Bezier (0.5 puntos)

Objetivo:
Desarrollar y validar clases de objetos gráficos elementales como las curvas de Bezier de grado 3 y practicar el muestreo de la curva mediante el método de las diferencias avanzadas.

Descripción:
Se quieren implementar las clases que permitan obtener instancias de curvas de Bezier de grado 3 utilizando las clases básicas de la práctica 1. La clase debe proveer métodos para el cálculo de los puntos de la curva y sus tangentes.

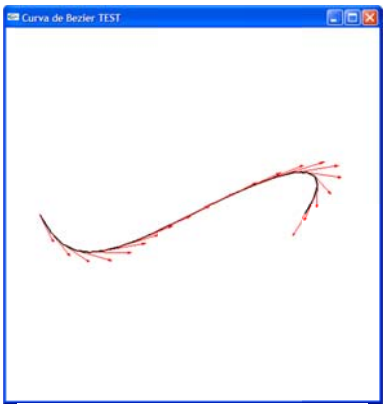
La clase *CurvaBezier* está definida en el fichero suministrado *CurvaBezier.h* . La implementación de clase *CurvaBezier.cpp* debe completarla el alumno siendo obligatorio el uso del método de **diferencias avanzadas** para la función miembro *getPoints( )* . Como se puede observar mirando el código, se han definido como matrices la característica de Bezier y la de coeficientes, de manera que en la implementación se pueda hacer uso de la clase *Matriz* de la práctica anterior.

El cálculo de la tangente en cada punto se puede realizar por el método que se prefiera incluida la evaluación directa de la función Q'(u).

El alumno debe construir, asimismo, un programa de visualización que haga uso de la clase anterior.

Proceso a seguir:
1. Análisis del código suministrado
2. Revisión de los conceptos teóricos sobre curvas paramétricas y, en especial, de Bezier
3. Construcción de los métodos *setC(), tangent(), getPoints() y getTangents()*
4. Generación de *VerCurvaSimple.exe*
5. Construcción de *Ver2Curvas.cpp* y generación de *Ver2Curvas.exe*
6. Ampliación de *Ver2Curvas* para alcanzar la calificación máxima

Puntuación y entregables:
La práctica puntúa 0,5 puntos.



Se obtienen 0,25 puntos si:
- Se construye correctamente la clase *CurvaBezier* según requisitos
- Se visualiza correctamente la curva test *VerCurvaSimple.cpp*
- Se construye *Ver2Curvas* que dibuje, al menos, dos curvas enlazadas con continuidad C^1. Cada curva será de un color diferente
- Se dibujan correctamente las tangentes en cada punto calculado

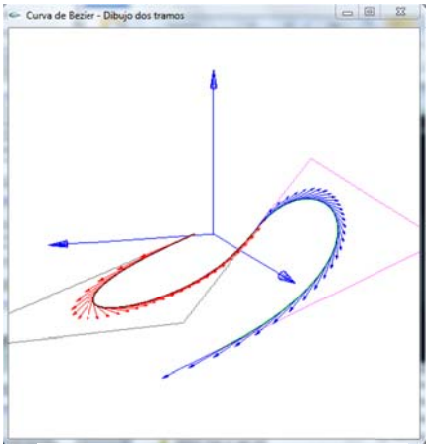
Se valorará para los 0,25 puntos restantes lo siguiente:
- Dibujo de los puntos de control, el polígono característico y los ejes de coordenadas
- Interacción mediante ratón para mover el dibujo (inspección)
- Animación basada en la transformación de los puntos de control usando *Algebra*
- Edición interactiva de los puntos de control

Entregables: ***VerCurvaSimple.exe***, ***Ver2Curvas.exe*** y todo el **código fuente** necesario para generarlos. Los ejecutables deben haberse generado en modo "Release" para Windows.

Sesiones: 1,5

Apoyo:

- *CurvaBezier.h* : Fichero de definición de la clase curva cúbica de Bezier que se suministra
- *CurvaBezier.cpp* : Fichero de implementación de la clase curva cúbica de Bezier. Se suministra incompleto
- *VerCurvaSimple.cpp* : Programa fuente de validación de la clase. El resultado correcto del programa se muestra en la figura 1
- Diferentes ejecutables como ejemplos
- Presentación de la práctica en *Practica2.pdf*



s

Práctica 2.2 Visualización superficies bicúbicas de Bezier (0.5 Puntos)

Objetivo:
Desarrollar y validar clases de objetos gráficos elementales como las superficies bicúbicas de Bezier practicando el muestreo de la forma mediante el método de las diferencias avanzadas.

Descripción:
Se quieren implementar la clase que permita obtener instancias de superficies de Bezier de grado 3 utilizando las clases básicas de la práctica 1. La clase *SuperficieBezier* está definida en el fichero *SuperficieBezier.h* que se provee. El método debe ser de nuevo el de **diferencias avanzadas**, esta vez para superficies. Además se deberán calcular, por simple evaluación de las derivadas parciales en (*u,v)*, las tangentes y la normal en cada punto calculado de la superficie.

Se debe construir un programa para la representación gráfica de una superficie a elegir.

Proceso a seguir:

1. Análisis del código suministrado
2. Revisión de los conceptos teóricos sobre superficies paramétricas y, en especial, de Bezier
3. Construcción de los métodos *setC(), utangent(), vtangent(), normal(), getPoints(), getNormals() y getTangents()*
4. Generación de *VerSuperficieAlambrico.exe*
5. Construcción de *VerSuperficie.cpp* y generación de *VerSuperficie.exe* para una superficie propia y visualización alámbrica
6. Ampliación de *VerSuperficie* para alcanzar la calificación máxima

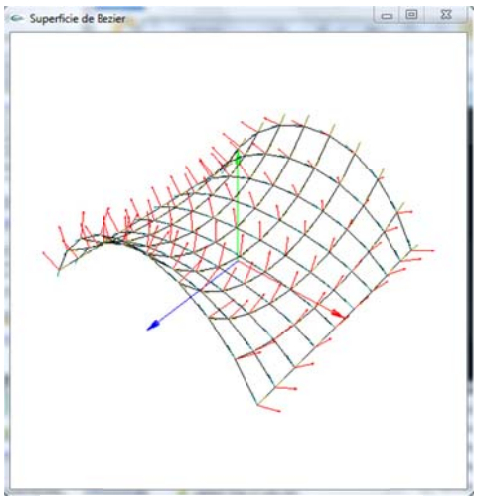
Puntuación y entregables:
La práctica puntúa 0,5 puntos.

Para la obtención de 0,25 puntos se requiere:

- La implementación correcta de la clase *SuperficieBezier*
- La representación gráfica correcta de *VerSuperficieAlambrico*
- El dibujo de las normales en cada punto de la superficie

Para los restantes 0,25 puntos se valorará:

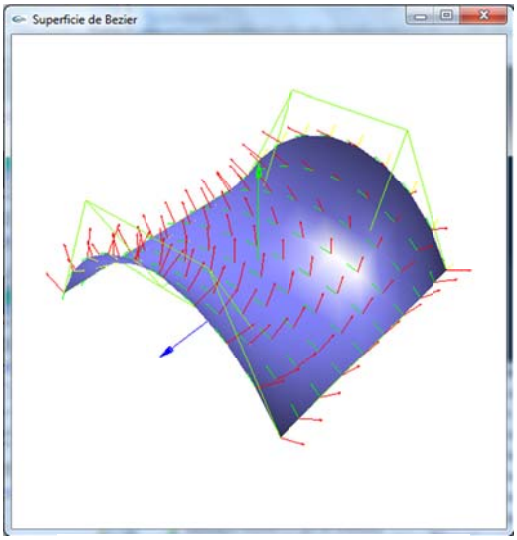
- El dibujo de la malla de control y los ejes sobre una superficie propia diferente
- La iluminación de la superficie como malla poligonal usando OpenGL
- La interactividad con la aplicación (inspección, luces, resolución de malla, etc.)
- Animación de los puntos de control usando transformaciones de *Algebra*



)

Entregables: ***VerSuperficieAlambrico.exe***, ***VerSuperficie.exe*** y todo el **código fuente** necesario para generarlos. Los ejecutables deben haberse generado en modo "Release" para Windows.

Sesiones: 1,5



Apoyo:

- *SuperficieBezier.h* : Definición de la clase superficie de Bezier. Se suministra
- *SuperficieBezier.cpp* : Implementación de la clase superficie de Bezier. Se suministra incompleto
- *VerSuperficieAlambrico.cpp* : Código de test que produce el resultado de la figura 3
- Diferentes ejecutables como ejemplos
- Presentación de la práctica en *Practica2.pdf*