Universidad de las Ciencias Informáticas

Facultad 5 Entornos Virtuales

MÓDULO PARA GENERAR EFECTOS DE HUELLAS EN ENTORNOS DE REALIDAD VIRTUAL

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autores: Alejandro Catalá Aguirre.

Aldo Lara González.

Tutor: Ing. Yanoski R. Camacho Román.

Ciudad de la Habana

Junio, 2008

“El triunfo es de los que se sacrifican”

José Martí.

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los____días del mes de____ del año _____.

_____
Firma del autor
(Alejandro Catalá Aguirre)

_____
Firma del autor
(Aldo Lara González)

_____
Firma del Tutor
(Ing. Yanoski R. Camacho Román)

Datos de Contacto

Nombre y Apellidos: Yanoski R. Camacho Román

Edad: 28 años

Ciudadanía: Cubano

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: Ingeniero en Informática

Categoría Docente: Profesor Instructor

E-mail: rcamacho@uci.cu

Graduado de la CUJAE, con cinco años de experiencia en el tema de la Gráfica Computacional, y líder de un proyecto de Realidad Virtual en la Universidad de las Ciencias Informáticas.

Agradecimientos

A todos los que han contribuido de una forma u otra con la realización de este proyecto, reciban nuestro más sincero agradecimiento, en particular a nuestro tutor Ing. Yanoski Camacho. A la Revolución y la UCI por darnos la oportunidad de formar parte de esta tropa del futuro y formarnos como los profesionales que somos.

De Alejandro

A mis padres por el cariño, apoyo y la atención infinita que me han brindado a lo largo de toda mi vida. A mi hermano por su incondicionalidad. A mis compañeros de mil batallas por haber compartido conmigo este tiempo. Y a todos los que han estado a lo largo de estos 5 años, que también contribuyeron a que este sueño se hiciera realidad. Para todos ellos mis eternos AGRADECIMIENTOS.

De Aldo

Después de este largo trayecto llega la culminación de los estudios universitarios. En este momento tan importante en mi vida, quiero agradecer a todos los que han hecho posible que llegara este día. Especialmente a mis padres, a toda mi familia y amigos por todo el apoyo brindado a lo largo mi carrera para lograr hacer realidad este sueño.

Dedicatoria

A mis padres Bertha y Jorge, quienes son los principales responsables de que este día tan importante haya llegado en mi vida.

A mis hermanos Jorgito y Thalía.

A la memoria de mi abuela Bolín.

A toda mi familia.

Alejandro.

A mis padres Norbelis y Aldo por su infinita dedicación y siempre estar presentes.

A mis hermanos Alejandro y Delia.

A la memoria mi abuela que desde donde esté se sentirá orgullosa de mí.

A mi segunda madre Idania que siempre estuvo para ayudarme.

A mi prima Nurita por sus constantes llamadas.

A toda mi familia y amigos que me han ayudado durante todo este tiempo.

Aldo.

Resumen

En la actualidad los efectos visuales y más específico los efectos como frenado de autos, rastro de humo, entre otros, han tenido una gran propulsión en el mundo del gráfico por computadora, debido al desarrollo ascendente que existe hoy en día en cuanto a juegos tridimensionales y simuladores como son los de conducción, aviación, tiro, por lo que se hace necesario lograr este tipo de efectos.

En el caso del presente trabajo se realiza un módulo que genera efectos de huellas, pero a partir de un conjunto de puntos de control que definen la forma de la huella. Para alcanzar esta meta, se estudian métodos de generación de curvas y superficies, que permitan aproximar lo mejor posible los efectos, se plantean ventajas y desventajas según sus características, lo que unido a las necesidades del trabajo influirán en la selección del método a utilizar para el proyecto.

El módulo resultante de esta investigación, permitirá que con simples modificaciones pueda ser acoplado a la herramienta de desarrollo de aplicaciones de Sistemas Realidad Virtual, SceneToolkit, y de esta manera suprimir la ausencia en dicha herramienta de efectos como los mencionados anteriormente.

Palabras claves:

Módulo, efecto, puntos de control, curvas y superficies

Tabla de Contenido

Introducción

El avance de la informática en los últimos tiempos, ha propiciado la creación y masificación de un nuevo término: "**Realidad Virtual**" (RV). Iniciada en los programas de entrenamiento militares, simuladores de vuelo, centros de investigación y académicos, ha pasado a los programas más variados en el ámbito profesional y doméstico. Hoy se empieza a aplicar en la medicina, arquitectura, educación, juegos entre otros campos.

El término "Realidad Virtual" suele asociarse a casi todo aquello que tiene que ver con imágenes en tres dimensiones generadas por ordenador y con la interacción de los usuarios con este ambiente gráfico.

A finales de los 80, los gráficos generados por computador entraron en una nueva época. Además de que las imágenes tridimensionales comenzaran a reemplazar a las bidimensionales, también comenzó a surgir la necesidad de un espacio de trabajo totalmente interactivo generado a través de la tecnología. El objetivo de la RV es crear una experiencia que haga sentir al usuario que se encuentra inmerso en un mundo virtual, aparentemente real; para ello, se sirve de gráficos tridimensionales así como del sonido que envuelve las escenas mostradas.

La aplicación de la RV es una alternativa para aprender, adquirir habilidades y entretener, a una mayor velocidad, en casi cualquier campo de acción, en cualquier lugar, a un costo menor y con mínimo riesgo. Es por ello que la RV cobra cada vez más importancia, por su capacidad de minimizar los gastos y daños que pueda ocasionar algún tipo de actividad humana, por ejemplo, en el campo militar, se entrenan soldados y pilotos mediante simuladores, en la astronomía los astronautas tienen la posibilidad de volar sobre la superficie simulada de un planeta desconocido y experimentar la sensación que tendrían si estuvieran allí.

Cuba, un país con escaso desarrollo en Sistemas de Realidad Virtual (SRV), ha comenzado a dar sus primeros pasos en esta dirección. La creación de la Universidad de las Ciencias Informáticas, trajo consigo la creación también de un polo de RV dentro de la misma, en el cual existe un proyecto que se dedica al desarrollo de una herramienta(SceneToolKit) para uso de los demás proyectos que están relacionados con el desarrollo de SRV; dicha herramienta posee diferentes módulos para distintas

ramas, como son sonido, animación de personajes, así como un conjunto de funcionalidades útiles para el trabajo matemático, entre otros, pero carece de funcionalidades para generar efectos de huellas en Entornos de RV. La ausencia de efectos como marca de frenado de autos e impacto de disparos en la herramienta, disminuye las posibilidades de sobresalir en el mercado mundial a las aplicaciones que necesiten de este tipo de efecto especial; por lo señalado anteriormente el **problema** a resolver por este trabajo es:

¿Cómo generar efectos de huellas en Entornos de Realidad Virtual?

El **objeto de estudio** del presente proyecto es la generación de efectos visuales en Entornos de Realidad Virtual, y como **campo de acción,** lo relacionado con los procesos de generación de efectos de huellas para Entornos de RV.

Para darle solución al problema de este proyecto se propone como **objetivo general**, desarrollar un módulo de clases para la generación de efectos de huellas en Entornos de Realidad Virtual.

Para satisfacer el objetivo propuesto se definen un grupo de tareas, que se pueden resumir en las siguientes:

- Investigar características básicas sobre efectos visuales.
- Seleccionar las herramientas a utilizar.
- Estudiar los algoritmos y técnicas ya existentes para la generación de efectos de huellas en Entornos de RV.
- Desarrollar soluciones técnicas para alcanzar los objetivos propuestos.
- Implementar un módulo de clases que de solución al problema planteado.

Como resultado de este trabajo, se pretende obtener un módulo de clases que genere efectos de huellas en Entornos de RV, que con pocos cambios pueda ser insertado a la herramienta en desarrollo

(SceneToolKit); lo que provocará un incremento en las funcionalidades y potencialidades de la herramienta, además de que resolverá la necesidad que tiene de generar efectos de huellas en Entornos de RV.

El contenido de este trabajo se encuentra estructurado de la siguiente manera:

En un primer capítulo, “Fundamentación Teórica”, se hace un análisis bibliográfico donde se investigan las técnicas, algoritmos y tendencias actuales para la generación de curvas y superficies a partir de puntos de control. En el capítulo 2, “Soluciones Técnicas”, se exponen las características del método escogido para darle solución al problema del trabajo, además de las herramientas de desarrollo usadas. En el capítulo 3 “Descripción de la Solución Propuesta”, se crea el modelo del dominio, se hace la captura de requisitos y se crea el modelo de casos de uso del sistema, lo que proporciona una visión más exacta del sistema. Por último en el capitulo 4 “Diseño e Implementación del Sistema”, se presenta el diagrama de clases de clases de diseño y de secuencia, así como los diagramas de despliegue, de componentes y los estándares de programación, también se muestran los resultados obtenidos. Finalmente, se ofrece un glosario de términos para ayudar a la comprensión del lenguaje técnico utilizado a lo largo del trabajo.

Capítulo 1: Fundamentación Teórica

Introducción

La representación virtual no es exclusiva de los ordenadores ni del lenguaje digital. Las primeras experiencias parten de los efectos especiales en el cine y la televisión y posteriormente en los videojuegos. Mantener e incrementar un sistema óptimo y lo suficientemente real ha sido y será el principal objetivo de los investigadores desde años atrás pues es y será una demanda presente en el campo de la realidad virtual y de los efectos especiales en la industria de los videos juegos. Obtener una simulación óptima de los diferentes tipos de efectos especiales es algo que hoy en día se continúa investigando rigurosamente. La visualización de dichos efectos incrementa el realismo de los entornos virtuales, convenciendo al usuario de suspender sus dudas y de sumergirse más en la simulación.

Los efectos de huellas, como son las marcas de frenado de autos, marca de disparos y rastro de humo, son efectos que se usan con frecuencia en el mundo de la RV, la generación de este tipo de efectos, puede ser difícil de lograr.

En el presente capítulo se profundizará en los diferentes métodos de generación existentes para la creación de este tipo de efectos. A través de la representación y diseño de curvas y superficies.

1.1 Representación de curvas y superficies

La curvas y superficies poseen una representación matemática precisa, y flexible. No obstante, y salvo raras excepciones, no es factible, en un sistema computarizado, representar una curva, o superficie, mediante una ecuación, debido fundamentalmente a la necesidad de editar la representación. Se comenzará explicando su representación matemática, para justificar la representación computacional que se hace en sistemas computarizados. [1]

A nivel matemático, las curvas y superficies se pueden representar como ecuaciones de varias formas, atendiendo a como aparezcan las distintas variables involucradas: [1]

Ecuaciones explícitas, en las que aparece de forma explícita una de las variables en función de las otras dos. Estas expresiones tienen alguna de las formas siguientes:

en 3D z = f(x,y) (superficie)
ó y = f1(x), z = f2(x) (curva en el espacio)

en 2D y = f(x) (curva en el plano)

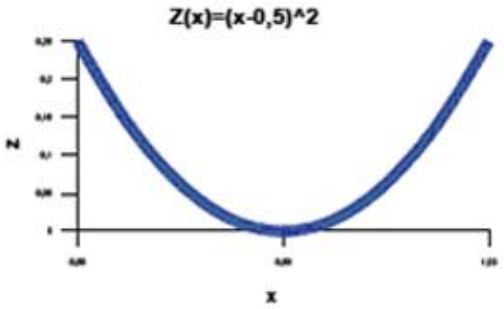

Figura 1a Curva explícita

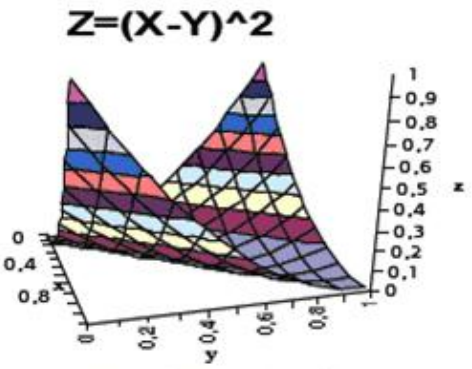

Figura 1b Superficie explícita

Figura 1 Curva y Superficie Explícita

Estas expresiones se suelen utilizar para curvas (figura 1a) y superficies univaluadas (figura 1b), como las que se suelen obtener como resultado de procesos de medida experimental (presión en puntos de

la superficie de la tierra, o elevación de un camino). La evaluación de la superficie es muy fácil. Sin embargo es difícil utilizarlas para casos generales, dado que las superficies que nos encontramos en la realidad, y que queremos modelar no serán, normalmente, univaluadas.

Ecuaciones implícitas, expresadas como una ecuación de las variables igualada a cero. En ellas no aparece ninguna variable despejada, por lo que su evaluación puede ser compleja. La expresión tendrá una de las formas siguientes:

f(x,y,z) = 0
f(x,y) = 0

según el elemento que se defina en el espacio o en el plano, e independientemente de la disensión de este. Un caso notable de curvas y superficies que se pueden expresar usando ecuaciones implícitas son las cuádricas. Por ejemplo un círculo (figura 2) de radio unidad se puede expresar como:

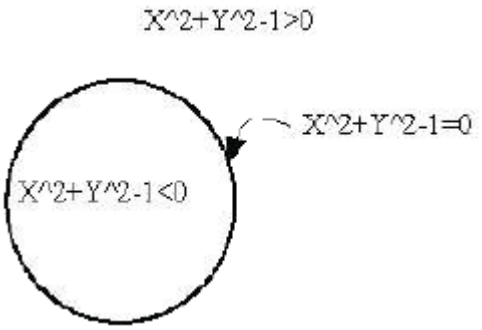

Figura 2 Curva implícita

$y^2 + x^2 - 1 = 0$

Las ecuaciones implícitas poseen la ventaja de ser orientables, es decir, es factible determinar hacia que lado de la superficie, o curva, se encuentra un punto sustituyendo sus coordenadas en la ecuación. Por ejemplo, el punto (0,0) está a la izquierda del círculo, el punto (1,0) está sobre el círculo, y el (2,2) está a la derecha.

Por contra, las ecuaciones implícitas pueden ser difíciles de evaluar. A partir de una expresión explícita se puede obtener una implícita. Obviamente, el recíproco no es siempre cierto.

Ecuaciones paramétricas, la curva o superficie se describe en base a un conjunto de parámetros que la recorren, como un conjunto de ecuaciones que permiten obtener cada una de las coordenadas a medida que el parámetro evoluciona sobre el elemento. Una curva se expresa usando un parámetro, **u**, que toma valores en un intervalo predeterminado, y dos o tres ecuaciones (según se defina, en el plano o el espacio). Así, una curva en el plano se define mediante el par de ecuaciones:

$x = f_1(u)$
$y = f_2(u)$ $u \in [u1,u2]$

El círculo anterior (figura 2) se puede expresar mediante las ecuaciones paramétricas:

$x = \cos(2\pi u)$
$y = \text{sen}(2\pi u)$ $u \in [0,1]$

Una superficie se describe en base a dos parámetros, (u, v), que describen dos direcciones ortogonales sobre ella, y a tres ecuaciones que expresan las coordenadas en base a estos parámetros:

$x = f_1(u,v)$,
$y = f_2(u,v)$,
$z = f_3(u,v)$ $u \ [u_1,u_2]$, $v \in [u_1,u_2]$

Modificando los valores de los parámetros se recorre la superficie, obteniéndose las coordenadas de los puntos que están sobre ella. Las expresiones paramétricas presentan la ventaja de ser flexibles, aunque no son orientables. Por otra parte, cualquier expresión explícita se puede poner, trivialmente, en forma paramétrica.

En aplicaciones de diseño se suelen utilizar expresiones paramétricas por su generalidad. No obstante, no es suficiente con la utilización de una expresión paramétrica. La necesidad de editar la curva, o superficie, obligaría a modificar, tanto los intervalos de variación de los parámetros como las funciones que describen las coordenadas (f_1, f_2 y f_3). Esto último no es factible, en un caso general, a menos que se fije la forma de estas funciones, haciendo que su modificación implique el cambio de un número reducido de parámetros cuyo efecto en la forma de la curva sea predecible. Una forma de conseguir esto es utilizar funciones polinómicas: [1]

$X(u) = X_0 + X_{1 \cdot u} + X_{2 \cdot u2} + ...$

El conjunto de curvas que se pueden generar depende del **grado** de los polinomios usados. En cualquier caso, la especificación de una curva se hace con una secuencia finita de parámetros. No

obstante, eso no es suficiente para el diseño de curvas, se necesita, además, que los parámetros sean relevantes, es decir, que se pueda intuir el valor de los parámetros a partir de la forma de la curva. La relación entre la forma de la curva y los coeficientes del polinomio. [1]

A continuación se verá como establecer la relación entre parámetros relevantes para el usuario y los coeficientes del polinomio [1] con un ejemplo. Utilizando polinomios de grado uno, esto es:

$X(u) = X_0 + X_{1 \cdot u}$

$Y(u) = Y_0 + Y_{1 \cdot u}$ $\quad u \in [0,1]$

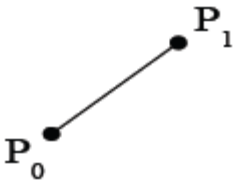

Figura 3 Línea generada por polinomios de grado1

La curva generada será la recta que pasa por los puntos (X_0, Y_0) y (X_0+X_1, Y_0+Y_1). Si se considera como intervalo de variación del parámetro el [0,1], la curva generada será la línea delimitada por estos puntos (ver figura 3). El cálculo de los coeficientes puede realizarse como sigue:

$(X_0, Y_0) = P_0$

$(X_0+X_1, Y_0+Y_1) = P_1 \quad => \quad (X_1, Y_1) = P_1 - P_0$

donde P_1 y P_0 son los puntos extremos. Generalizando, un método satisfactorio para generar curvas y superficies es definirlas a partir de un conjunto finito de puntos de control. Esto equivale a poner las expresiones anteriores en la forma:

$X(u) = \sum i=1^n x_i \cdot F_i(u)$

donde x_i es la coordenada x del i-ésimo punto de control, y las F_i son funciones de forma, que expresan la dependencia con el parámetro. En el ejemplo anterior, $X(u) = X_0 + X_{1 \cdot u} = P_0.x + (P_1.x - P_0.x) \cdot u = P_0.x \cdot (1-u) + P_1.x \cdot u$, luego las funciones de forma son:

$F_0(u) = 1-u$ y $F_1(u) = u$.

1.1.1 Proceso de diseño

En el ciclo de diseño de una curva, o superficie, solo se trabaja con los puntos de control, que son en sí la representación de la curva, o superficie, en el modelo. Por tanto, en el proceso de diseño se seleccionan y modifican los puntos de control, hasta que se obtenga la curva o superficie deseada [1]. El proceso puede comenzar con valores predeterminados asignados a los puntos de control.

En este proceso, el dibujo de la curva debe realizarse repetidas veces, al menos cada vez que se mueve un punto, e idealmente mientras se mueven los puntos con el dispositivo de entrada (durante el ciclo de realimentación). Por este motivo, cuando la máquina no es suficientemente potente, la visualización durante la edición se simplifica, aproximando la superficie como una malla de curvas. [1]

1.2 Métodos de diseño de curvas

La mayor parte de los métodos de diseño de curvas y superficies se basan en la utilización de puntos de control a partir de los cuales se define como un promedio de los puntos de control: [1]

$$P(u)=\sum_{i=1}^{n} P_i \cdot B_i(u) \qquad \textbf{(1)}$$

donde P_i son los puntos de control y $B_i(u)$ son funciones de forma, dadas en forma paramétrica.

Existen diversos métodos de diseño de curvas, con diferentes características. Entre estas, cabe destacar el carácter del método (que puede ser local o global) y el comportamiento respecto a los puntos de control (interpolante o no). Un método tiene carácter **local** cuando la modificación de un punto de control afecta solamente a la forma de la curva, o superficie, en las proximidades del punto de control. Por el contrario, en un método **global**, la modificación de un punto de control afecta a toda la curva, o superficie. Es más fácil editar una curva o superficie utilizando un método local, ya que permite ajustar la forma de la curva trozo a trozo. [1]

Un método **interpola** a los puntos de control cuando el elemento generado (curva o superficie) pasa por ellos.

Generalmente es necesario usar polinomios continuos a trozos, para conseguir métodos de diseño local, lo que influye en el grado de continuidad de la curva. El grado de continuidad indica el número de veces que se puede derivar su ecuación obteniendo una función continua. Geométricamente esto está relacionado con la continuidad del elemento. Una curva con continuidad C^0 es continua, pero su pendiente no. Si la continuidad es C^1 la curva y su pendiente son continuas, las primeras derivadas (tangentes) de las curvas en los puntos de unión son iguales. Si la continuidad es C^2 las primeras y segundas derivadas de los segmentos de curva son iguales en la intersección. En la figura 4 se puede apreciar cada una de las continuidades.

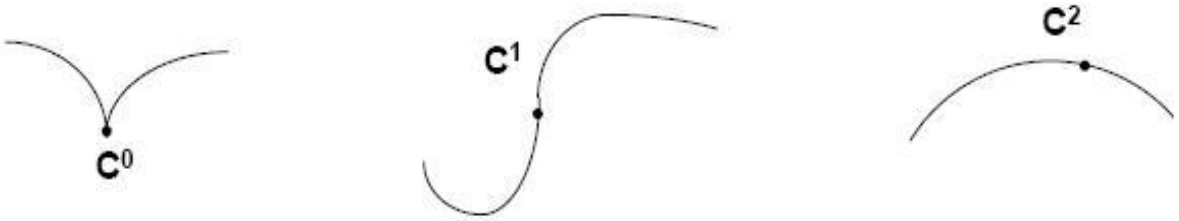

Figura 4 Continuidad de las curvas

La continuidad depende de la forma en que está parametrizada la curva. Para que la curva sea continua en un punto, sus vectores tangentes a izquierda y derecha deben coincidir. No obstante, la magnitud del vector tangente no influye en la apreciación que podemos hacer de continuidad en el punto. Por este motivo, se suele hablar de continuidad geométrica, denotada con G, en lugar de continuidad matemática. Una curva tiene continuidad geométrica en un punto si las tangentes a izquierda y derecha tienen la misma dirección, independientemente de la magnitud del vector tangente. Por este motivo, la continuidad matemática implica continuidad geométrica (salvo el caso especial en que el vector tangente es nulo). Normalmente se requiere como mínimo continuidad G1 y con frecuencia G2. [1]

En este epígrafe se profundizará en las curvas de Bézier, los Splines, los B-splines y los B-splines no uniformes racionales (**NURBS**). Las curvas de Bézier destacan por su simplicidad de formulación, tanto en la forma de Bézier usando polinomios de Bernstein, como en el planteamiento de De Casteljau. Los splines tienen una utilidad muy limitada en diseño, aunque son una buena opción cuando se quiere un método que interpole los puntos de control. [1]

Las curvas B-splines a diferencia de las curvas anteriores, permiten controlar el grado de continuidad. Por otro lado, es posible obtener el método de Bézier como caso particular de B-spline. Independientemente del método usado, la descripción de la curva también se puede realizar usando notación matricial, separando la dependencia con el parámetro de la dependencia con los puntos de control. [1]

1.2.1 Curvas de Bézier

Las curvas de Bézier son un tipo de curvas cúbicas paramétricas definidas por Pierre Bézier. Cada segmento de este tipo de curvas se especifica mediante 4 puntos (P1, P4, P2, P3). Dos de esos puntos son los puntos de inicio (P1) y final (P4) del segmento, mientras que los otros 2 puntos (P2, P3) especifican indirectamente los vectores tangentes al segmento en los puntos inicial y final del segmento. En la figura 5 se observa la representación de un segmento de curva de Bèzier, correspondiéndose los puntos 0, 1, 2 y 3 a P1, P2, P3 y P4 respectivamente. [2]

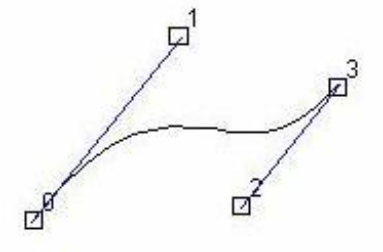

Figura 5 Curva de Bézier.

Matemáticamente se expresa:

$$\boldsymbol{P}(u)=\sum_{i=0}^{n} \boldsymbol{P}_i B_i^n(u) \qquad u\in[0,1] \tag{2}$$

Donde $B_i^n(u)$ es la función base, o de forma; para el cálculo de las funciones base existen varios métodos entre ellos: polinomios de Bernstein, y la forma matricial.

La figura 6a muestra las funciones de forma para n=2, y la figura 6b para n=3. Se puede observar que todas las funciones de forma son nulas en los extremos, salvo la primera y la última que valen uno

para u=0 o u=1. Esto implica que en los puntos extremos, es decir, al principio y final de la curva solo influya uno de los puntos de control, y que en los extremos la curva pase por el primer o último punto de control.

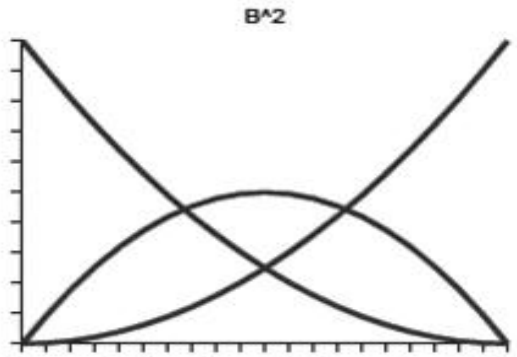

Figura 6a Funciones de forma de grado 2

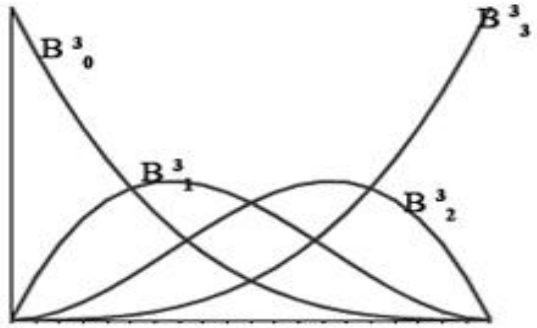

Figura 6b Funciones de forma de grado 3

Figura 6 Funciones de forma

Para ningún otro valor de **u**, hay funciones de forma con valor igual a uno. Esto implica que la curva no pase por ningún otro punto de control. [1]

Para cualquier valor de **u** la suma de las funciones de forma es igual a uno. Es decir, se puede ver las funciones de forma como pesos en una media ponderada de los puntos de control. Como consecuencia de esto, la curva estará siempre dentro de la envolvente convexa de los puntos de control (mínimo polígono convexo que contiene a los puntos de control). [1]

Se puede observar también que cada una de las funciones de forma presenta un máximo, que se presenta en valores de **u** crecientes con el índice de la función. Esto hace que la influencia de cada punto de control sea mayor para un valor de **u** que crece con el índice del punto, por lo que la curva sigue la forma de la poligonal, pero de forma más suave.

Es fácil comprobar (calculando las pendientes en los extremos) que la dirección de salida en el primer punto de control coincide con la de la recta que pasa por el segundo punto de control, e igualmente la dirección de llegada al último punto es la de la recta que lo une al penúltimo. En resumen, la curva tiene las siguientes propiedades: [1]

- La curva es interior a la frontera convexa.
- Pasa por P_0 (u=0) y por Pn (u=1), pero no por los intermedios.
- La dirección de salida de estos puntos está determinada por P_1 y P_{n-1}.
- La curva sigue la forma de la poligonal.
- La continuidad es C^{∞}.
- El grado del polinomio está determinado por el número de puntos de control.
- La modificación de un punto de control afecta a toda la curva.

Interpolación con curvas de Bézier:

Se puede interpolar puntos mediante las splines de Catmull-Rom, que son una sucesión de curvas de Bézier de tercer grado, unidas con continuidad paramétrica C^1 [3]. Las splines de Catmull-Rom, interpolan una lista de m+1 puntos P_i, a excepción de los puntos inicial P_0 y final P_m como se muestra en la figura.

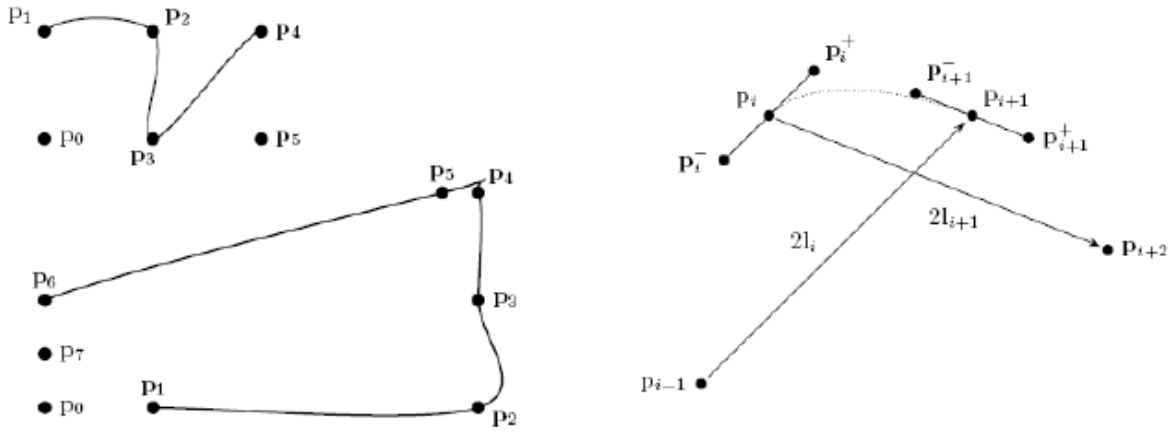

Figura 7 Interpolación con curvas de Bézier.

Entre cada par de puntos intermedios P_i y P_{i+1} se define una curva de Bézier mediante puntos de control agregados que garanticen la continuidad paramétrica. La idea es hacer que la derivada en cada punto sea la "velocidad media" que se requiere para ir del punto anterior al posterior. (Por eso no se interpolan los extremos). El parámetro se hace variar en forma continua entre 0 y m, tomando el valor **i** cuando la curva pasa por P_i, es decir que $P_{(i)} = P_i$. En los puntos dados:

$$l_i = \frac{dP(i)}{du} = \frac{P_{i+1} - P_{i-1}}{(i+1)-(i-1)} = \frac{P_{i+1} - P_{i-1}}{2}$$

Para lograr esa derivada, se ubican dos nuevos puntos de control: $\mathbf{P_i^-}$ y $\mathbf{P_i^+}$ definidos mediante:

$$P_i^- = P_i - \frac{l_i}{3} \qquad P_i^+ = P_i + \frac{l_i}{3}$$

Habiendo asignado la misma velocidad media a ambos tramos, es posible que para un tramo muy corto la curva "no tenga tiempo de frenar" (overshoot) y genere un rulo. Se ve un ejemplo entre P_4 y P_5 en la figura 7. Cuando hay tramos de distinta longitud (y por lo tanto, en general) es preferible utilizar un factor de corrección para la velocidad asignada en cada tramo (que suele ser la longitud del tramo). Hay varias formas de hacerlo y varios nombres para el resultado, el más conocido es el de Overhauser. Como se puede notar, en aquellos puntos tienen control local, el movimiento de un punto interpolante afecta a lo sumo a cuatro tramos de la curva. [3]

1.2.1.1 Algoritmo de Casteljau

La manera tradicional de trazar las curvas de Bézier está basada en el algoritmo de Casteljau. En este apartado, se describe dicho algoritmo para los casos de generación de una curva que pase por dos puntos y se acerque a un tercero. A partir de estos dos casos se generaliza el método dando las ecuaciones que modelan matemáticamente el comportamiento del algoritmo de Casteljau. [4]

A continuación se explica la generación de una curva que pasa por dos puntos y "tiende a" un tercero. Sean p_0 y p_2 los puntos por los que debe de pasar la curva y sea p_1 el punto que "tira" de ella, colocados en la forma que se indica en la figura 8:

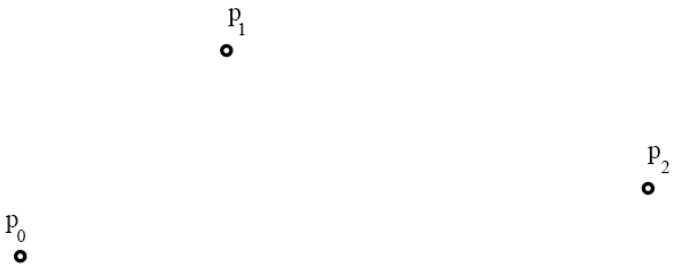

Figura 8 Tres puntos de control de una curva spline

uniendo P_0 con P_1 y P_1 con P_2 y subdividiendo esas rectas a razón t, 1-t, se crean unos nuevos puntos p^1 :

$$\mathbf{p}_0^1 = (1\text{-}t)\,\mathbf{p}_0 + t\,\mathbf{p}_1$$
$$\mathbf{p}_1^1 = (1\text{-}t)\,\mathbf{p}_1 + t\,\mathbf{p}_2 \qquad \textbf{(3)}$$

y repitiendo el proceso con los nuevos puntos p^1:

$$\mathbf{p}_0^2 = (1\text{-}t)\,\mathbf{p}_0^1 + t\,\mathbf{p}_1^1 \qquad \textbf{(4)}$$

Este punto p^2 pertenece a la curva buscada. Una representación gráfica de lo anterior se expresa en la figura 9.

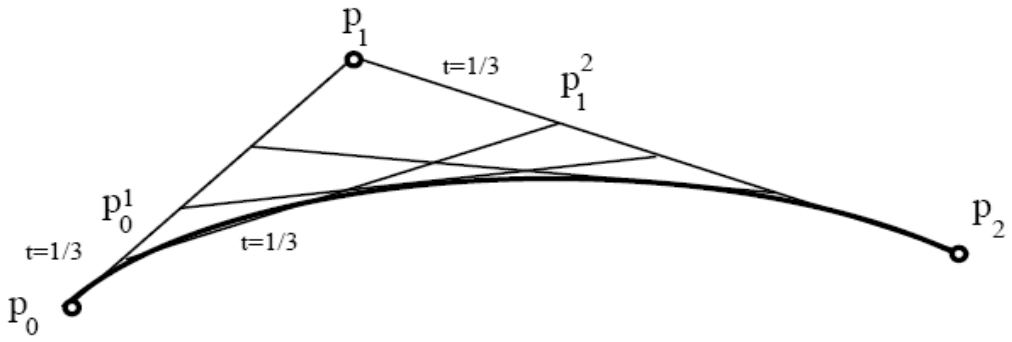

Figura 9 Curva spline de grado 2

en la que se ha hecho el proceso para t=1/3, t=1/2 y t=2/3 dejando simplemente marcado en la figura el caso para t=1/3. Insertando las dos primeras ecuaciones en la tercera se obtiene la ecuación paramétrica de la curva:

$$\mathbf{c}\ (t) = (1\text{-}t)^2\ \mathbf{p}_0 + 2t\ (1\text{-}t)\ \mathbf{p}_1 + t^2\ \mathbf{p}_2 \qquad \textbf{(5)}$$

Esto es una expresión cuadrática en t. Por lo tanto, c(t) traza una parábola al variar t entre menos infinito e infinito. Para t = 0 la curva pasa por el punto p_0, para t = 1 pasa por el punto p_1 propiedad que se conoce como interpolación en los extremos de la curva. Para t entre 0 y 1 la curva recorre el tramo entre estos puntos. Dado que la curva que se ha realizado es un polinomio de grado dos, se dice que la curva spline realizada es de grado 2. También se conoce como curva de Bézier de orden 3, pues son 3 los puntos de control implicados en la realización de la misma. [4]

Las ventajas del algoritmo de de Casteljau son su sencillez y el hecho de que involucra tan sólo sumas, ya que todos los términos son positivos. [5]

1.2.1.2 Polinomios de Bernstein

En el apartado anterior, se ha visto como mediante el algoritmo de Casteljau se puede definir una curva de comportamiento suave. También se ha llegado a una ecuación paramétrica que expresa el algoritmo para los casos lineal y cuadrático. Esta ecuación paramétrica se conoce como la forma de Bernstein de una curva de Bézier. [4]

En las ecuaciones 4 y 5 se puede ver que los términos que aparecen multiplicando los diferentes puntos de control, coinciden con los resultantes de aplicar el teorema del binomio a la suma [t + (1-t)]. Esto es:

$$[\mathbf{t} + (\mathbf{1} - \mathbf{t})]^2 = \sum_{\mathbf{i}=\mathbf{0}}^{\mathbf{n}} \binom{\mathbf{n}}{\mathbf{i}} \mathbf{t}^{\mathbf{i}} (\mathbf{1} - \mathbf{t})^{\mathbf{n}-\mathbf{i}} = \sum_{\mathbf{i}=\mathbf{1}}^{\mathbf{n}} \mathbf{B}_{\mathbf{i}}^{\mathbf{n}} (\mathbf{t}) \qquad \textbf{(6)}$$

en donde cada término de la sumatoria es un polinomio de Bernstein.

$$\mathbf{B}_{\mathbf{i}}^{\mathbf{n}} (\mathbf{t}) = \binom{\mathbf{n}}{\mathbf{i}} \mathbf{t}^{\mathbf{i}} (\mathbf{1} - \mathbf{t})^{\mathbf{n}-\mathbf{i}} \qquad \textbf{(7)}$$

De este modo, es posible expresar la ecuación paramétrica de la curva como:

$$C_n(t) = \sum_{i=0}^{n} p_i \, B_i^n(t) \qquad (8)$$

lo que se conoce como forma de Bernstein de una curva de Bézier.

Otra propiedad de las curvas de Bézier es su control pseudo local. Es debido a que el polinomio de Bernstein $B_i^n(t)$ tiene un único máximo en el intervalo $0<t<1$ localizado en $t = i/n$. Por lo tanto, moviendo el punto de control p_i, la curva estará afectada sobre todo en la zona en la que el parámetro vale $t = i/n$, aunque el efecto alcanzará a toda la curva, pues el polinomio afecta a toda ella. [4]

1.2.1.3 Forma matricial

A la hora de programar una ecuación de la forma de las curvas de Bézier en la que aparecen productos y sumas mezclados, es más fácil si se consigue expresar la ecuación en forma de un producto de vectores. Se desarrolla inicialmente la curva de grado 2 para luego pasar a curvas generales de grado n. [4]

Sea C(t) una curva de Bézier de grado 2 como la de la ecuación 10. Agrupando términos se llega a la expresión:

$$C(t) = (p_0 - 2p_0 + 2p_1 p_0 - 2p_1 + p_2) * \begin{pmatrix} 1 \\ t \\ t^2 \end{pmatrix} \qquad (9)$$

En general, resulta un vector creado como combinación lineal de los puntos de control por otro vector formado por potencias crecientes del parámetro t. Descomponiendo el primer vector, queda la ecuación matricial buscada ecuación 10. [4]

$$C(t) = (P_0 P_1 P_2) * \begin{pmatrix} 1 & -2 & 1 \\ 0 & 2 & -2 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} 1 \\ t \\ t^2 \end{pmatrix} \qquad (10)$$

En general, una curva no racional de Bézier de grado n puede ser expresada como:

$$\mathbf{C(t) = (p_0 \ldots p_n) * N * \begin{pmatrix} t^0 \\ \ldots \\ t^n \end{pmatrix}} \quad \textbf{(11)}$$

en donde los coeficientes de la matriz N están generados por:

$$\mathbf{n_{ij} = -(1)^{j-i} \binom{n}{j} * \binom{j}{i}} \quad \textbf{(12)}$$

1.2.2 Spline

Los splines son una familia de curvas que permiten definir segmentos de curva con gran precisión y trazado muy suave. Son de gran utilidad en aplicaciones para el diseño de superficies. Spline en gráficos por computador y modelado geométrico hace referencia a la representación paramétrica por trozos de geometrías, con un nivel específico de continuidad.

El grado de una curva de Bézier determina el número de puntos que contiene el polígono de control. Debido a esto, cuando se necesita aumentar el número de puntos para tener más flexibilidad en la definición de la curva, se debe aumentar el grado. A efectos prácticos y de coste computacional, grados excesivamente altos (n > 10) no son aconsejables. La extensión natural de estas curvas para obtener otras más flexibles, es la de las curvas polinómicas a trozos. Se trata de definir una nueva curva que sea la unión de varias curvas polinómicas simples (como las curvas de Bézier) y que cumplan una serie de requisitos de continuidad en la unión entre cada dos de estas curvas polinómicas. Esta nueva curva se llamará **spline**. [6]

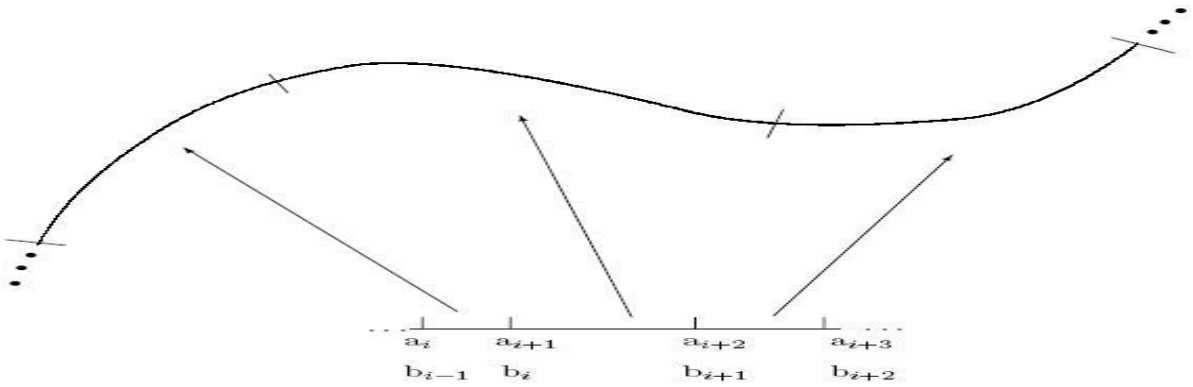

Figura 10 Curva Spline.

Si se tienen L curvas de Bézier unidas de manera consecutiva y cada una de ellas definida en el intervalo [0,1], tal como se aprecia en la figura 10, es posible realizar una transformación local del parámetro tal que queden definidas en $[a_i,b_i]$. Así obtenemos una curva global como unión de todas ellas y cuyo parámetro estará definido en [a,b].

Propiedades

La forma geométrica de la curva queda perfectamente definida con los polígonos de control de los segmentos de Bézier. La continuidad de la curva variará en función de la parametrización escogida. Con una misma forma geométrica se pueden obtener curvas con diferentes grados de continuidad en función del tipo de parametrización escogida.[6]

1.2.2.1 Splines de interpolación cúbicos

Este es uno de los tipos de curva más populares, ya que a la potencia de las curvas Splines para modelar curvas se une la posibilidad de interpolación para facilitar su guiado y mantienen la continuidad C' por toda la curva. La curva de tipo spline cúbica consiste en partir la curva en trozos de curva cúbica paramétrica que une cada pareja de puntos, con la condición de que sea continua hasta la segunda derivada, es decir, continua respecto a posición, tangente y curvatura.

Para la representación de este tipo de curva uno de los usados es el algoritmo de Catmull-Rom spline interpola un juego de puntos de tal manera que la dirección tangente a cada punto es paralela al segmento de la línea que conecta los dos puntos vecinos. [7.1]

La aplicación consecutiva a cada uno de los trozos de la curva permite calcular todos los vectores tangentes de los puntos intermedios mediante un sistema de ecuaciones lineales. Es decir, si se tienen **m** puntos, y por tanto m-1 trozos de interpolación, existen m-2 puntos intermedios, por lo que se obtienen m-2 ecuaciones iguales a la anterior. [8]

Esto hace que falten dos condiciones para dejar definida de forma unívoca la ecuación, y las que se utilizan normalmente son una de las siguientes: [8]

- Vectores tangentes conocidos en los puntos extremos.
- Derivadas segundas en los puntos extremos iguales a cero; este último tipo es conocido como spline cúbico natural.

1.2.3 Funciones base B-spline

El orden de las funciones base en la expansión polinomial de Bernstein utilizada por las curvas de Bézier depende del número de puntos de control del modelo. Debido a ello, se necesitan funciones base de grado (n−1) para pasar una curva polinomial de Bézier a través de n puntos. Sin embargo, el cálculo de funciones de orden elevado es ineficiente y numéricamente inestable. Este tipo de curvas tienen problemas para ajustarse a algunas funciones complejas, además el control que se tiene sobre la representación no es suficientemente local. [9]

La solución a este problema es utilizar funciones base polinomiales por partes conocidas como splines. Este tipo de funciones existen solo en una parte del intervalo paramétrico y dependen de algunos puntos de control, lo cual permite tener un control local de la representación. Adicionalmente, el orden de estas funciones puede ser escogido según sea conveniente. Las funciones base splines se definen de la siguiente manera: [9]

Sea $U = \{u_0, \cdots u_m\}$ una secuencia no decreciente de números reales, i,e., $u_i \leq u_i+1$, $i = 0, \ldots, m-1$. Donde u son llamados **knots** (nodos o nudos), y U es el vector de nodos. La función base de grado p, denotada por $N_{i,p}$ se define como:

$$\mathbf{N_{i,0}(u)} = \begin{cases} \mathbf{1} & \mathbf{si\ u_i \leq t < u_i + 1} \\ \mathbf{0} & \text{en caso contrario} \end{cases}$$

$$\mathbf{N_{i,p}(u)} = \frac{\mathbf{u - u_i}}{\mathbf{u_{i+p} - u_i}} \mathbf{N_{i,p-1}(u)} + \frac{\mathbf{u_{i+p+1} - u}}{\mathbf{u_{i+p+1} - u_{i+1}}} \mathbf{N_{i+1,p-1}(u)} \quad \textbf{(13)}$$

Esta técnica de evaluación de la curva requiere menos operaciones debido a que la recursividad afecta solamente a los términos escalares. En la figura 11 se aprecian las funciones base de orden 3 generadas a partir del vector de nodos U = {0 0 0 0 0, 25 0, 5 0, 75 1 1 1 1}.[9]

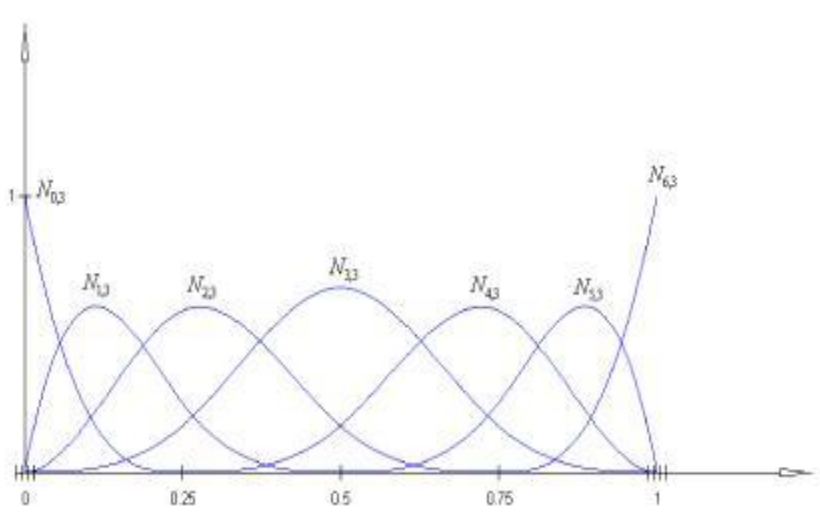

Figura 11 Funciones base splines

1.2.4 B-spline

Existen varios tipos de curvas B-splines, a continuación se abordarán algunas de ellas.[6]

Las curvas B-splines surgen a partir de una combinación de las funciones base Splines y los puntos de control a aproximar. Es definida por su **grado**, su **polígono de control** y su secuencia de **knots** que no son más que los puntos de enlaces entre los distintos trozos de la curva. Como una B-spline es una curva formada por diversos segmentos polinómicos, el grado de la línea seria el mismo grado que tendrá cada uno de los segmentos.

Una curva B-spline de grado p está definida por:

$$\mathbf{C(u)} = \sum_{\mathbf{i=0}}^{\mathbf{n}} \mathbf{N_{i,p}(u)P_i} \qquad \textbf{(14)}$$

Donde P_i son los puntos de control, y $N_{i,p}(u)$ son las funciones base B-splines de grado p. En la figura 12 se muestra una curva B-spline.

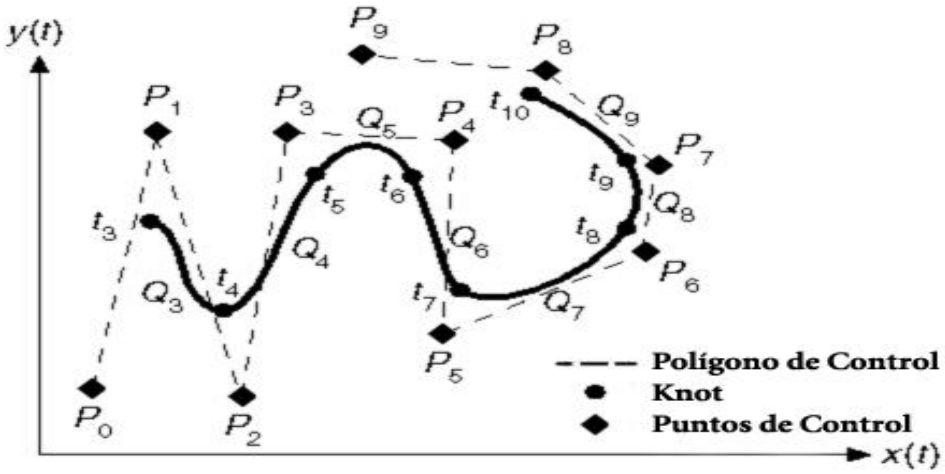

Figura 12 Curva B-spline

El **polígono de control** es un conjunto de puntos en el espacio donde algunos de ellos pueden estar repetidos. La curva pasaría por el primer y por el último punto y se acercará, de manera suave, a todos los otros puntos.

La secuencia de **knots** normalmente se definen en el intervalo [0,1] y, por tanto, el primero valdrá 0 y el último 1. Si un mismo valor está repetido **r** veces, se dice que tiene una multiplicidad **r**. Diferentes multiplicidades implicarán diferentes propiedades en relación al punto de control que corresponde al knot. En concreto, una multiplicidad igual al orden supondrá que la curva interpole al punto correspondiente. El primer knot de la lista y el último tendrán multiplicidad igual al orden para asegurar la interpolación de los puntos extremos.

El número de puntos y el de *knots* deben cumplir que:

$$N_k = N_p + o \qquad \text{con:} \quad \left| \begin{array}{l} N_k\text{=Número de } knots \\ N_p\text{=Número de puntos} \\ o\text{=orden} \end{array} \right.$$

Para evaluar una curva de este tipo se pueden usar varios algoritmos:

Inserción de nodos. Algoritmo de Boehm.

La inserción de knots se utiliza para "mejorar" una curva que se define primero en forma grosera y se va refinando secuencialmente hasta adoptar la forma deseada. El polígono de control se reemplaza localmente con uno más "refinado". Para un valor arbitrario del parámetro habrá que hacer las cuentas de la interpolación lineal. Los nuevos puntos de control pueden luego moverse libremente para reajustar la curva.

Inserción múltiple: Algoritmo de De Boor.

El algoritmo de De Boor se utiliza para calcular el punto variable de la curva y trazarla. Consiste en encontrar para un dado valor del parámetro **t** el punto P(t,t,t) y el método equivale a insertar n veces el nudo **t**.

1.2.4.1 B-spline uniforme no racionales

Cada trozo de la curva se puede ajustar para que el parámetro t esté situado en [0,1], suele hacerse para que tenga un parámetro continuado. Esto significa que llamando t_i al límite superior del parámetro en el trozo de curva i-ésimo, se puede poner que para cada trozo se cumple $t_i \leq t \leq t_i+1$.

El término uniforme hace referencia a que los nudos están separados mediante intervalos iguales del parámetro t, mientras que el término no racional se utiliza para distinguir este tipo de curva de aquellos en que las funciones paramétricas se definen mediante la relación entre dos polinomios.

Existe una continuidad C^0, C^1 y C^2 en todos los nudos de unión de cada tramo de la curva, la continuidad que nos ofrece este tipo de curva es muy interesante, pero sacrificando el control de por donde queremos que pase la curva.

Si entre los puntos de control que aportamos para dibujar la curva repetimos posiciones de forma sucesiva, hacemos que la curva pase más cerca de dichos puntos, e incluso podemos hacer que pase por ellos. Todo esto a costa de perder grados de continuidad.

1.2.4.2 B-spline no uniformes y no racionales

Se diferencian de las curvas anteriores en que el intervalo del parámetro t entre los nudos sucesivos no necesita ser uniforme. Esto implica que las funciones correspondientes a cada tramo de la curva no tienen porque ser las mismas.

Las ventajas que tienen este tipo de curvas respecto a las uniformes son:

- Puede modificarse el orden de continuidad para nudos concretos.
- Si se reduce la continuidad a C^0 la curva pasa por el punto de control, pero sin que eso implique que el tramo de curva sea una línea recta.
- Los puntos de comienzo y fin de la curva pueden interpolarse con facilidad.
- Pueden añadirse nudos y puntos de control para modificar el aspecto de la curva.

La notación de este tipo de curva difiere de la anterior en que vamos a introducir una secuencia de valores de nudo que serán t_0, ..., t_{m+4}, es decir, que tendremos 4 nudos más que puntos de control se hayan definido. Los nudos van a ser los puntos de enlace de cada tramo de curva con el siguiente.

La única restricción que tiene la secuencia de nudos es que siempre sea no decreciente, lo que permite que si tengan valores iguales. Cuando esto ocurra se produce una multiplicidad del nudo. Esto significa que si existe un nudo de multiplicidad 2 en el que $t_i = t_i+1$ el tramo de curva Q_i se reduce a un punto. La multiplicidad de nudos reduce el orden de continuidad de la curva.

No existe un único conjunto de funciones paramétricas que nos definan los diferentes tramos de la curva, sino que, al depender cada función de los intervalos entre los nudos, se van a definir de manera

recursiva en función de las funciones de los tramos anteriores. Para cuarto orden (es decir, funciones polinómicas cúbicas) la recurrencia de funciones sería:

$$B_{i,1}(t) = \begin{cases} 1, t_1 \leq t < t_{1+1} \\ 0, resto\ de\ casos \end{cases}$$

$$B_{i,2}(t) = \frac{t - t_i}{t_{i+1} - t_i} B_{i,1}(t) + \frac{t_{i+2} - t}{t_{i+2} - t_{i+1}} B_{i+1,1}(t)$$

$$B_{i,3}(t) = \frac{t - t_i}{t_{i+2} - t_i} B_{i,2}(t) + \frac{t_{i+3} - t}{t_{i+3} - t_{i+1}} B_{i+1,2}(t)$$

$$B_{i,4}(t) = \frac{t - t_i}{t_{i+3} - t_i} B_{i,3}(t) + \frac{t_{i+4} - t}{t_{i+4} - t_{i+1}} B_{i+1,3}(t)$$

que se corresponden con las funciones de orden 1 a 4 correspondientes al punto i-ésimo.

1.2.4.3 B-spline no uniformes racionales (NURBS)

Las NURBS, B-splines racionales no uniformes, son representaciones matemáticas de geometría en 3D capaces de describir cualquier forma con precisión, desde simples líneas en 2D, círculos, arcos o curvas, hasta los más complejos sólidos o superficies orgánicas de forma libre en 3D. La figura 13 muestra la representación de una curva NURBS.

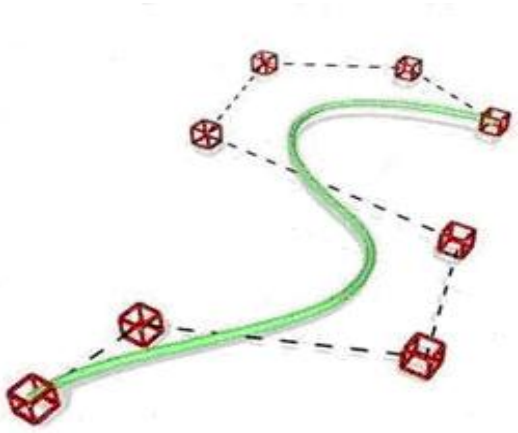

Figura 13 Curva NURBS

Una curva NURBS se define mediante cuatro elementos: **grado**, **puntos de control**, **nodos (knot)** y **regla de cálculo**.

El **grado** es un número. Este número normalmente es 1, 2, 3 o 5 pero puede ser cualquier número entero positivo. Las líneas y polilíneas NURBS son grado 1, los círculos son grado 2 y la mayoría de las formas libres son grado 3 o 5.

Los **puntos de control** son una lista de puntos de grado+1 como mínimo. Una de las maneras más sencillas de cambiar la forma de una curva NURBS es mover los puntos de control.

Los **nodos** son una lista de números de grado+N-1, donde N es el número de puntos de control. Esta lista de números de nodos debe cumplir varias condiciones técnicas. El modo estándar para asegurar que las condiciones técnicas se cumplan es requerir que el número se mantenga igual o aumente a medida que vaya bajando en la lista y limitar el número de valores duplicados a que no sea superior al grado. Los valores duplicados del nodo en la mitad de la lista del nodo hacen que una curva de NURBS sea menos suave.

La **regla de cálculo** de una curva utiliza una fórmula matemática que coge un número y asigna un punto. La regla de cálculo NURBS es una fórmula que comprende el grado, los puntos de control y los nodos. En la fórmula hay lo que se llama funciones bases B-spline.

Las expresiones para evaluar una NURBS serán equivalentes a las de evaluación de las B-splines, añadiendo los pesos Wi obtenemos la expresión de la línea NURBS en función de la base B-spline.

$$\vec{\mathbf{S}}(\mathbf{u}) = \frac{\sum_{\mathbf{i}=\mathbf{1}}^{\mathbf{n}} \mathbf{W_i}\, \vec{\mathbf{P_i}}\, \mathbf{N_i^n}}{\sum_{\mathbf{i}=\mathbf{1}}^{\mathbf{n}} \mathbf{W_i}\, \mathbf{N_i^n}} \qquad \textbf{(15)}$$

El cambio de los pesos de los puntos de control de una NURBS afectará, en general, a la forma de la curva. En concreto, el incremento del peso de un punto tenderá a atraer la curva hacia ese punto, en la figura 14 se aprecia la diferencia entre un movimiento de un punto de control y un incremento del peso para el mismo punto.

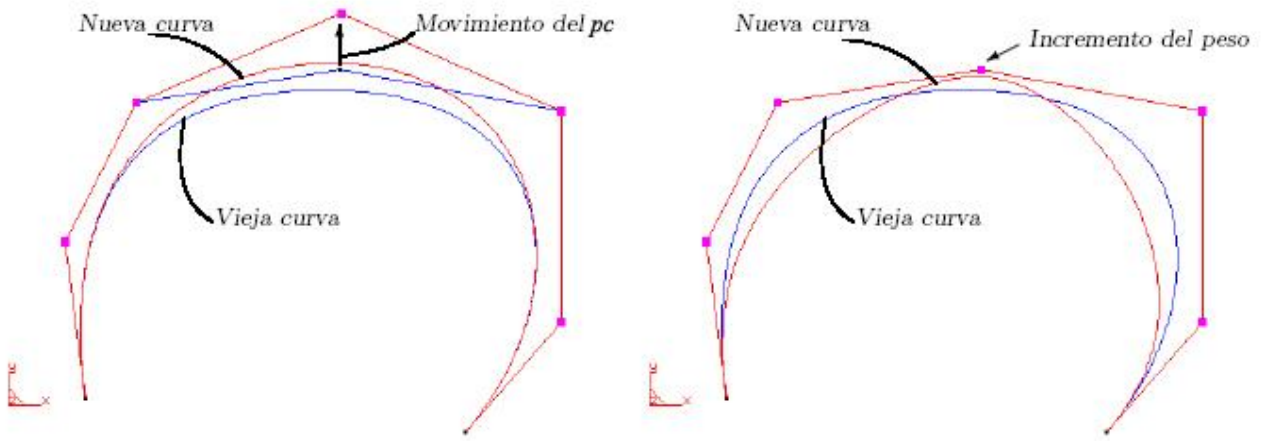

Figura 14 Influencia de los pesos en la curva NURBS

Si todos los pesos tienen el mismo valor, la forma de la curva es equivalente a la curva no racional. Esto limita el interés de los pesos para modelar porque, si se varían varios pesos de varios puntos, la curva tiende a recuperar su forma original. Se parte de la restricción de que todos los pesos deben ser positivos para evitar singularidades. [6]

1.3 Diseño de superficies

Una superficie se describe matemáticamente usando dos parámetros, que establecen un sistema de coordenadas sobre ella, permitiendo recorrerla. La superficie se puede definir directamente a partir de una malla de puntos de control, al igual que una curva, con la única diferencia de que los puntos de control forma una distribución bidimensional y que las funciones de forma dependen de dos parámetros, Bij(u,v).[1]

Alternativamente, es posible definir una superficie a partir de una o varias curvas. Por otra parte, es utilizado también las mallas de triángulos como una aproximación a la superficie. En esta sección se aborda la representación y construcción de superficies, a partir de curvas.

1.3.1 Superficies Bézier

Los métodos más comunes de representación de superficies en el modelamiento geométrico son las ecuaciones implícitas y las funciones paramétricas [10]. La ecuación implícita de una curva en el plano xy tiene la forma f(x, y) = 0. Esta ecuación describe una relación implícita entre la cordenadas x y y de los

$$C(u) = (x(u), y(u)) \qquad a \leq u \leq b \tag{16}$$

puntos sobre la curva. En la forma paramétrica, cada una de las coordenadas de un punto sobre la curva es representada como una función explícita de un parámetro independiente.

Así, C(u) es una función valorada por la variable independiente **u** en el intervalo paramétrico arbitrario [a,b], el cual usualmente está normalizado entre [0,1]. De igual forma una superficie definida por una ecuación implícita f(x, y, z) = 0 tiene una representación paramétrica de la forma:

$$S(u, v) = \big(x(u, v), y(u, v), z(u, v)\big) \tag{17}$$

De las ecuaciones 16 y 17 se puede inferir que el proceso de parametrización implica un cambio dimensional de $R^n \rightarrow R^{n-k}$.

La parametrización planteada por Bézier [11], se basa en los polinomios de Bernstein (ver subepígrafe 1.2.1.2) de grado **n,** para crear curvas y superficies paramétricas que dependen de unos puntos de control. Una curva de Bézier de grado **n** tiene la forma:

$$\mathbf{C(u)} = \sum_{\mathbf{i=0}}^{\mathbf{n}} \mathbf{B_{i,n}(u)P_i} \tag{18}$$

En donde $B_{i,n}(u)$ son las funciones base. Estas tienen la forma característica de los polinomios de Bernstein de grado **n** (ver ecuación 7).

Las superficies de Bézier se definen como:

$$S(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_{i,n}(u)\, B_{j,m}(v)\, P_{i,j} \tag{19}$$

De las ecuaciones 7 y 19, se puede apreciar que el grado de las funciones base que generan las curvas y superficies de Bézier depende del número de puntos de control con los cuales se genera la representación. Esta forma paramétrica presenta problemas de cálculo en las funciones base cuando hay demasiados puntos de control (ecuación 7). Las curvas y superficies de Bézier se pueden extender a una forma racional en donde cada punto de control P_i tiene asociado un peso:

$$S(u,v) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{m} B_{i,n}(u)\, B_{j,m}(v)\, w_{i,j}\, P_{i,j}}{\sum_{i=0}^{n} \sum_{j=0}^{m} B_{i,n}(u)\, B_{j,m}(v)\, w_{i,j}} \tag{20}$$

Las B-splines presentan una solución a este problema, haciendo que las funciones base existan solo en una parte del intervalo paramétrico. [9]

1.3.2 Superficies B-spline

Una superficie B-spline se obtiene al combinar linealmente una red bidireccional de puntos de control $P_{i,j}$ y los productos de las funciones base Splines $N_{i,p}(u)$ y $N_{j,q}(v)$ definidas en los vectores nodo U y V respectivamente. [9]

$$S(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u)\, N_{j,q}(v)\, P_{i,j} \tag{21}$$

Las superficies B-splines conforman una representación unificada que tiene todas las propiedades de continuidad y derivación de las funciones base Splines en ambas direcciones paramétricas. En las superficies B-splines se genera un producto tensor de las funciones base bivariadas que presenta

capacidad de ajuste local y alta continuidad. El producto tensor de las funciones base $N_{i,p}(u)$ $N_{j,q}(v)$ con orden p en la dirección u y q en la dirección v está definido por los vectores de nodos normalizados

$$U = \{\underbrace{0,\ldots,0}_{p+1}, u_{p+1},\ldots,u_{n-1},\underbrace{1,\ldots,1}_{p+1}\}$$
$$V = \{\underbrace{0,\ldots,0}_{q+1}, v_{q+1},\ldots,v_{m-1},\underbrace{1,\ldots,1}_{q+1}\}$$

para una red bidireccional de puntos de control $P_{i,j}$ con n puntos de control en la dirección u, y m en la dirección v.

De forma análoga a las curvas B-splines, las superficies tienen propiedades de control local dado que $N_{i,p}(u)$ $N_{j,q}(v) = 0$, si (u, v) están por fuera del rectángulo [ui, ui+p+1)×[vj , vj+q+1). La continuidad y las derivadas de S(u, v) provienen del producto tensor de sus funciones base $N_{i,p}(u)$ $N_{j,q}(v)$. En particular, S(u, v) es p − k (q − k) veces diferenciable en la dirección u (v) para un nodo u (v) de multiplicidad k. El producto tensor de las funciones base interpola las 4 esquinas de la red bidireccional de puntos de control $P_{i,j}$ en una superficie B-spline. [9]

1.3.3 Superficies NURBS

Realizando la extensión de las expresiones de las líneas NURBS en dos direcciones, obtenemos las superficies NURBS. Estas son el producto tensorial de dos líneas. [9]

Las superficies quedarán perfectamente definidas mediante el grado y la lista de **knots** en cada una de las direcciones **u** y **v**. Tanto el grado como el número de **knots**, no tienen que coincidir para **u** y **v**. Los puntos de control serán una cuadrícula en el espacio que en general tendrá diferente número de puntos según cada una de las direcciones. [9]

Una superficie NURBS S(u,v) de grado (p, q) es una función racional bivariada de la forma quedando su expresión general como:

$$\vec{S}(u,v) = \frac{\sum_i \sum_j W_{ij} \vec{P}_{ij} N_i^m(u) N_j^n(u)}{\sum_i \sum_j W_{ij} N_i^m(u) N_j^n(u)} \quad \textbf{(22)}$$

donde {Wij} i,j representan los valores escalares de los pesos asociados a los puntos de control y {Pij} i=0,...,n;j=0,...,m es la malla de puntos de control , con vectores de nodos U y V definidos.

La figura 15 muestra una superficie Nurbs:

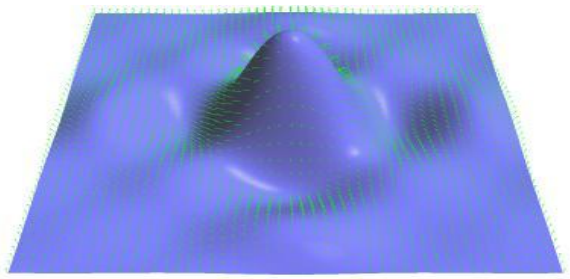

Figura 15 Superficie NURBS

La mayoría de las operaciones a realizar sobre estas superficies, incluida la evaluación, pueden realizarse primero sobre un sentido, obtener como resultado una línea y aplicar la operación sobre el otro sentido. De esta manera se reduce el problema de operar sobre una superficie al problema de operar sobre un conjunto de líneas.[9]

1.4 Texturizado de superficies

El mapeado de texturas es una técnica gráfica que consiste en aplicar una serie de dibujos o plantillas a las caras de un objeto tridimensional. Por ejemplo, pegando la foto del rostro de una persona en una esfera plana, la habremos convertido en una cabeza. Lo mismo ocurre si se aplica la textura de la madera a un cilindro: obtendremos un tronco bastante real. [12]

Para que estas texturas se ajusten debidamente al objeto donde se pegan, se suele aplicar una corrección de perspectiva o textura inversa, que estira o comprime la textura según su posición, evitando cualquier anomalía y pérdida de realidad. Básicamente se trata de buscar el píxel de la pantalla que se corresponde con cada píxel de la textura. [12]

Las coordenadas de la textura son asignadas a los vértices del objeto en cuestión, según diversos procedimientos, de tal forma que se identifique cada uno de los píxels de dicho objeto durante el render con uno de los texels de la textura para sustituir, o alterar, alguna de las características de superficie del pixel original, como el color, el brillo o la transparencia, por aquellas que indique la propia textura. En función del tamaño visible del objeto, la resolución de render y la de la propia textura, podrá resultar que un pixel del objeto se corresponda con varios texels, promediando los valores de éstos, o que un texel haga lo propio con varios píxels, aplicando entonces sus valores a todos los píxels afectados. [12]

- 'Pegar' una textura sobre un objeto consiste en definir una función unívoca (función de mapeo o mapping) que a cada punto del espacio ocupado por la superficie o sólido le hace corresponder un punto en el espacio donde está definida la textura. [13]
- Los elementos del espacio de textura se denominan texels. [13]

En la figura 16 se muestra lo explicado anteriormente:

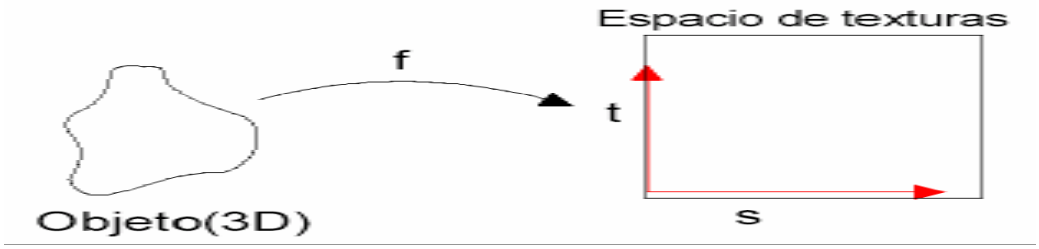

Figura 16 Mapping

Las características y ventajas del mapeado de texturas son obvias, ya que permite aumentar enormemente la complejidad visual de un modelo sin necesidad de aumentar su complejidad geométrica. En cualquier escena que se precie, la iluminación y las texturas suelen tener mayor importancia que la propia geometría que sustenta los modelos. [12]

Existen diversos tipos de mapeado pero el presente trabajo se centrará en el **Mapeado Coordenadas UV:**

El mapeado **UV** es una manera de mapear texturas de tipo imagen sobre modelos tridimensionales. Se puede usar para aplicar texturas a formas arbitrarias y complejas como cabezas humanas o animales. A menudo estas texturas son imágenes pintadas o dibujadas, creadas con programas como Photoshop, o cualquier otro. [12]

Las dimensiones del espacio de textura están normalizadas, de manera que todas sus coordenadas varían entre 0 y 1 como se muestra en la figura. [13]

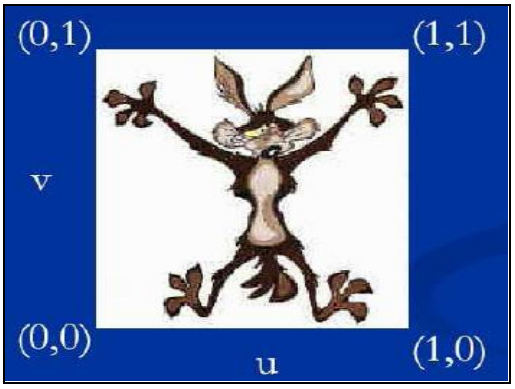

Figura 17 Coordenadas textura

Se realiza una asignación punto a punto. Es decir, a cada primitiva se le asignan coordenadas **uv** entre 0 y 1 a cada uno de sus puntos, definiendo un área de correspondencia entre el modelo y la textura. Estas áreas no son necesariamente iguales, sino que la textura se expande o contrae para cubrir la primitiva (ver figura 18). En el interior de la primitiva se realiza una interpolación de la textura.

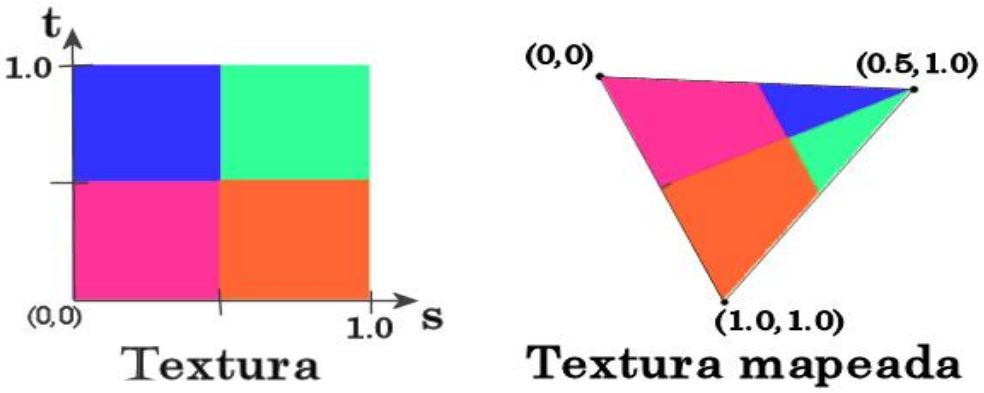

Figura 18 Coordenadas textura

Posiblemente se acerca más a la realidad que cualquier otro, puesto que la textura está "anclada" a las coordenadas **UV** correspondientes (para entendernos, las coordenadas **UV** son como la longitud y la latitud en una esfera, un valor determinado por dos números que indica la posición exacta de un punto en el espacio). Sólo tiene que colocar la textura, y retorcer, estirar o doblar la figura, para que la textura aplicada se acomode a los nuevos valores (ver figura 18). [13]

Capítulo 2: Soluciones Técnicas

Introducción

En el presente capítulo se proponen las soluciones técnicas para generar efectos de huellas en Entornos de Realidad Virtual, y soluciones especificas a partir de los métodos estudiados en el capitulo anterior, también se abordará sobre la selección del método más apropiado según los objetivos planteados y las necesidades de este trabajo.

2.1 Generación de puntos de una curva

En el capítulo anterior se estudiaron varias de las técnicas más utilizadas para la generación de curvas y superficies a partir de un conjunto de puntos de control. En el presente trabajo de diploma para generar los puntos de una curva que aproxima a los puntos de control, se hace uso de la curva B-spline.

Las curvas B-splines surgen a partir de una combinación de las funciones base Splines y los puntos de control a aproximar. Es definida por su grado, su polígono de control y su secuencia de knots. A partir de las curvas B-splines se puede obtener una generalización de las funciones de Bézier, ya que si el número de puntos de control es igual al grado y los valores de la secuencia de knot están entre 0 y 1, entonces es una curva de Bézier.

El grado de la curva B-spline se puede ajustar independientemente del número de puntos de control; a diferencia de lo que ocurre con las curvas de Bézier, que a pesar de ser una herramienta muy potente en el diseño de curvas, tiene esa limitación, si la curva que se pretende modelar tiene una forma compleja, se necesitan demasiados puntos de control para modelarla, lo que hace que el grado de la curva se eleve demasiado, que a efectos prácticos y de costo computacional, no es aconsejable; otra ventaja de la curva B-spline es que fácilmente se puede obtener la continuidad requerida.

Una solución para controlar el importante factor grado en la modelación de curvas, es la de las curvas polinómicas a trozos, como es el caso de la curva B-spline. Se trata de definir una nueva curva que sea la unión de varias curvas polinómicas simples (como suelen ser las curvas de Bézier) y que cumplan una serie de requisitos de continuidad en la unión entre cada dos de estas curvas polinómicas, por lo que se dispone de un control local, lo cual permite un ajuste más fino.

El polinomio que se utilizará para definir los trozos de las curva B-spline será de grado 3, este polinomio es una función continua que posee primera y segunda derivada continua, condición que asegura que se cumplan las importantes restricciones de continuidad y suavidad en los puntos de unión entre un tramo de curva y otro. Las funciones tendrán la forma:

$$x = x(t) = a_x t^3 + b_x t^2 + c_x t + d_x$$
$$y = y(t) = a_y t^3 + b_y t^2 + c_y t + d_y$$
$$z = z(t) = a_z t^3 + b_z t^2 + c_z t + d_z$$
$$t \in [0,1]$$

Por las razones anteriormente expuestas es que se escogió el algoritmo de la curva B-spline para generar los puntos de la curva a partir de un conjunto finito de puntos de control.

2.2 Generación de superficie

La generación de superficie se trabajará a partir de dos curvas. Luego de tener las curvas se generará una malla de triángulos como una aproximación a dicha superficie. En esta sección se explica la representación y construcción de superficies, a partir de curvas.

La superficie que se generará será a partir de dos listas de puntos (L_1 y L_2) que conforman dos curvas a una distancia **h** una de la otra.

Al algoritmo que se desarrollará se le pasará como parámetros la lista de **puntos de control** (puntos por donde se desea que pase la superficie), y un ancho, que sería el ancho de la huella.

Para generar las listas de puntos (L_1 y L_2) se trasladan los **puntos de control** dados inicialmente, la distancia **h** dividida por 2 en el sentido positivo del eje Y y en el sentido negativo del mismo eje (esto puede variar según conveniencia del programador), por lo que se obtienen dos nuevas listas de **puntos de control**. Con las cuales se trabaja para obtener las listas de puntos (L_1 y L_2) que conforman las dos curvas.

Para calcular cada punto de una curva se determinan las funciones bases o de forma (ver subepígrafe 1.2.3), luego se hace una sumatoria de la multiplicación de los puntos de control por su respectiva función base, este cálculo se realiza mediante la forma matricial, de esta manera se obtiene un nuevo valor para cada una de las coordenadas X, Y y Z, las cuales conforman el punto que será adicionado a la lista.

Luego de haberse obtenido las dos listas de puntos de las curvas (L_1 y L_2), se concebirá un algoritmo, el cual tiene como entrada dichas listas y como salida una lista de triángulos, donde cada tres puntos de la lista se forma un triángulo, y de esta forma se genera la superficie o malla deseada (ver figura 19).

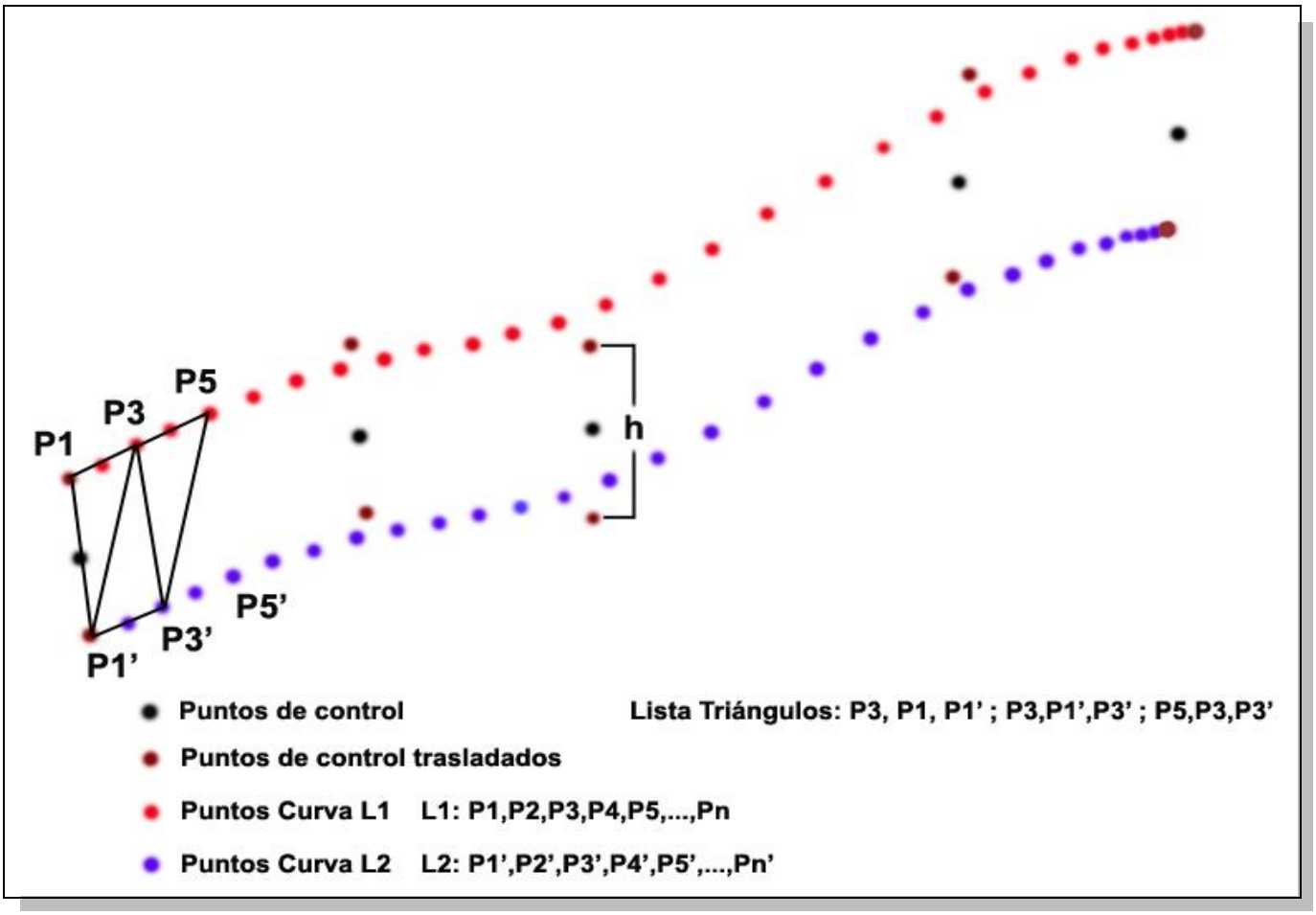

Figura 19 Proceso de creación de la superficie

2.3 Metodología

La metodología de desarrollo utilizada en la realización de este trabajo de diploma es: Rational Unified Process (RUP, siglas en inglés), la cual es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas Orientados a Objetos.

Posee tres características fundamentales las cuales influyeron en la selección de esta metodología y que la hacen una metodología robusta y poderosa.

- **Dirigido por casos de uso:** los casos de uso reflejan lo que los usuarios futuros necesitan y desean**,** constituyen la guía fundamental establecida para las actividades a realizar durante todo el proceso de desarrollo del sistema.

- **Centrado en la arquitectura:** se refiere a que se incluye los aspectos estáticos y dinámicos más significativos del sistema. Además recoge una serie de factores como la plataforma en la cual funciona el software, los bloques de construcción reutilizables que se tienen, consideraciones de implementación, sistemas heredados y requisitos no funcionales.

- **Iterativo e Incremental:** RUP divide el proyecto en fases de desarrollo, propone además que cada una de ellas se desarrolle en iteraciones, las cuales aportan un incremento en el proceso de desarrollo y terminan con el cumplimiento del punto de control trazado en la fase.

RUP deberá adaptarse a las características propias del proyecto u organización. El tamaño del mismo, así como su tipo o las regulaciones que lo condicionen, influirán en su diseño específico. También se deberá tener en cuenta el alcance del proyecto.

2.4 Herramientas de desarrollo y lenguaje utilizado

En el desarrollo del módulo propuesto se utilizó el Visual Studio 2003 y para crear la documentación de esta, el Rational Rose 2003 y como lenguaje el C++.

2.4.1 Visual Studio 2003

Microsoft Visual Studio 2003 es un software desarrollado por la compañía Microsoft, entre las características fundamentales que presenta se encuentran algunas como, su capacidad de crear aplicaciones de alto rendimiento rápidamente, disminuir los costos de funcionamiento de tecnologías de la información y además se integra con una amplia gama de aplicaciones, sistemas y dispositivos, la plataforma sobre la que se ha desarrollará este trabajo es el entorno de programación Visual C++, debido a que la programación correspondiente a la parte gráfica y el modelo matemático se realizan en C++, además de ser un lenguaje con posibilidad de desarrollo en plataforma libre.

2.4.2 Rational Rose

Es una herramienta de desarrollo del software para el Modelado Visual. Rational tiene gran ventaja para el equipo de desarrollo ya que los unifica a través del modelamiento el cual está basado en el Unified Modeling Language (UML). El UML es la notación estándar para arquitectura de software. Esto significa que con Rational Rose, todo el equipo puede comunicarse con un lenguaje y una herramienta. Rational Rose domina el mercado de herramientas para el análisis, modelamiento, diseño y construcción orientado a objetos. Esta herramienta permite detallar, analizar, diseñar el sistema antes de codificarlo. Tiene varias características que lo distinguen como son el de chequear la sintaxis UML, mantiene la consistencia de los modelos del sistema del software, genera la documentación automáticamente, la generación de código a partir de los modelos, permite la ingeniería inversa (crear modelo a partir de código) entre otras.

2.4.3 C++

Existen diversos lenguajes que se utilizan para desarrollar aplicaciones gráficas profesionales en 3D entre ellos se ecuentran: Lenguaje Ensamblador, C y C++, por ser los que con más velocidad ejecutan el código (menor costo de ejecución del programa). A éstos se ha unido recientemente el Java como una opción para el desarrollo de este tipo de aplicaciones. Es decisión de los directivos del proyecto "Herramienta de desarrollo para sistemas de Realidad Virtual", implementar este proyecto mediante el lenguaje C++, que ha estado en su línea de trabajo con magníficos resultados. Es el lenguaje en que se tiene mayor experiencia por parte del equipo de desarrolladores del proyecto. Si se estudian las características de este lenguaje, se puede apreciar lo acertado de la elección, dado que C++ es un lenguaje de programación de propósito general, especialmente indicado para la programación de sistemas por su flexibilidad y su potencia. Siendo uno de los más utilizados en las comunidades de desarrolladores de software, incluyendo la programación gráfica.

Capítulo 3: Descripción de la Solución Propuesta

Introducción

En este capítulo se obtiene una visión más específica del sistema que se va a desarrollar, partiendo fundamentalmente de las soluciones técnicas. Se describen las reglas del negocio y se tiene un acercamiento al modelo del sistema a través del modelo del dominio. Se especifican los requerimientos funcionales y no funcionales, lo que va a permitir hacer una concepción general del módulo que se va a desarrollar, y se identifican los casos de usos del sistema.

3.1 Reglas del negocio

Para que la superficie se genere de manera correcta, es necesario que los puntos de control no estén muy distantes uno del otro, o sea no exista una diferencia brusca, entre las respectivas coordenadas de cada punto, la aplicación no se detiene, pero no se obtendría el resultado esperado, además el nivel de realismo disminuiría.

Para esta versión el fichero de textura a cargar debe ser de extensión BMP, sus dimensiones deben ser múltiplos de potencias de 2, Ej.: 8, 16, 32, 64, 128, 256, 512, etc; en caso de no existir el fichero en el camino indicado, ser de otro formato, estar corrupto o no tener el modelo textura no constituye un error, lo que sucederá es que la malla se creará sin textura.

Se recomienda no utilizar como entrada muchos puntos de control, es decir tratar de que sean los necesarios, ya que esto aumentaría la cantidad de cálculo que tiene que realizar el ordenador, lo que a su vez repercutiría en el tiempo de ejecución.

3.2 Modelo del dominio

Para el modelamiento del sistema, se desarrolla el modelo del dominio. Con el objetivo de comprender y describir los conceptos más importantes dentro del contexto del sistema (ver figura 20).

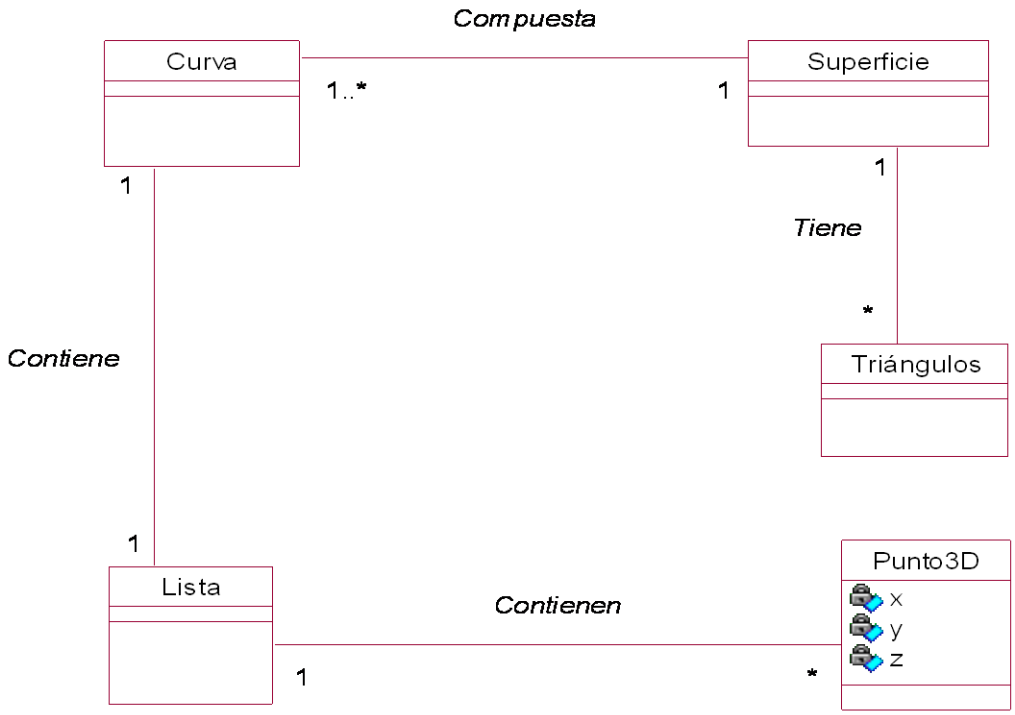

Figura 20 Diagrama del Modelo del Dominio

3.3 Glosario de términos del modelo del dominio

Curva: Encargado de recibir la lista de los puntos de control y realizar el cálculo de las funciones bases para luego obtener los puntos por donde pasará la curva.

Superficie: Encargado de recibir la lista de los puntos de las curvas generadas y construir la malla.

Punto3D: Estructura para la ubicación de los puntos de control en el espacio 3D.

Triángulos: Recibe la lista de los puntos de la malla y realiza la triangulación.

Lista: Estructura para almacenar la lista de los puntos de la curva.

3.4 Captura de requisitos

A continuación se expondrán las condiciones que se deben alcanzar para resolver nuestro problema, mediante los requisitos funcionales y no funcionales del sistema.

3.4.1 Requisitos funcionales

1. Crear punto.
2. Adicionar punto.
3. Eliminar punto.
4. Crear matriz de puntos de control a partir de la lista de puntos de control.
5. Obtener punto de la matriz de puntos de control.
6. Trasladar los puntos de control una distancia h/2 (dada h) en el sentido positivo del eje escogido.
7. Trasladar los puntos de control una distancia h/2 (dada h) en el sentido negativo del eje escogido.
8. Calcular las funciones bases B-spline.
9. Calcular las coordenadas de los puntos de la curva.
   9.1 Sumatoria de los puntos de control multiplicadas por sus respectivas funciones bases.
10. Generar lista de puntos de la primera curva.
11. Generar lista de puntos de la segunda curva.
12. Obtener una lista de triángulos a partir de las dos listas de puntos generadas.
    12.1 Combinar dichas listas y obtener una nueva lista de puntos donde cada tres puntos se forma un triángulo.

13. Generar malla.
14. Eliminar malla.
15. Texturizar.

3.4.2 Requisitos no funcionales

- **Usabilidad:** Los futuros usuarios del sistema serán programadores con conocimientos básicos con respecto al tema de gráficos por computadoras.

- **Rendimiento:** Debe ser una aplicación en tiempo real donde tenga nivel de velocidad de procesamiento de los cálculos, tiempo de respuesta y recuperación.

- **Soporte:** Una versión inicial deberá ser compatible con la plataforma Windows, pero debe estar preparado para que con rápidas modificaciones pueda migrar para Linux.

- **Software:** Sistema operativo Windows.

- **Hardware:** Compatibilidad con tarjetas gráficas de la familia NVIDIA, y tarjeta de video de más de 128 megabyte de RAM.

- **Diseño e implementación:** Debe trabajar con la biblioteca gráfica de OpenGL y además de emplear los Estándares de codificación del proyecto “Herramienta de desarrollo para Sistemas de Realidad Virtual”. Se harán llamadas a dichas bibliotecas desde Lenguaje C++. Se regirá por la filosofía de Programación Orientada a Objetos.

3.5 Modelo de casos de uso del sistema

Después de quedar definidos los requisitos funcionales y no funcionales. Para satisfacer dichos requisitos se definen los procesos que responden a dichas funcionalidades, definiendo los actores que interactúan con el sistema y los casos de uso que dan cumplimiento a estos requerimientos.

3.5.1 Definición de los actores del sistema

Tabla 1"Justificación de los actores"

Actores	Justificación
STK	Es el que se beneficiará con las funcionalidades que brinda el módulo de clases, a grosso modo: establecer los puntos de control y generar la superficie deseada.

3.5.2 Casos de uso del sistema

1. Generar Curva.
2. Generar Superficie.
3. Gestionar Punto.

3.5.3 Diagrama de casos de uso

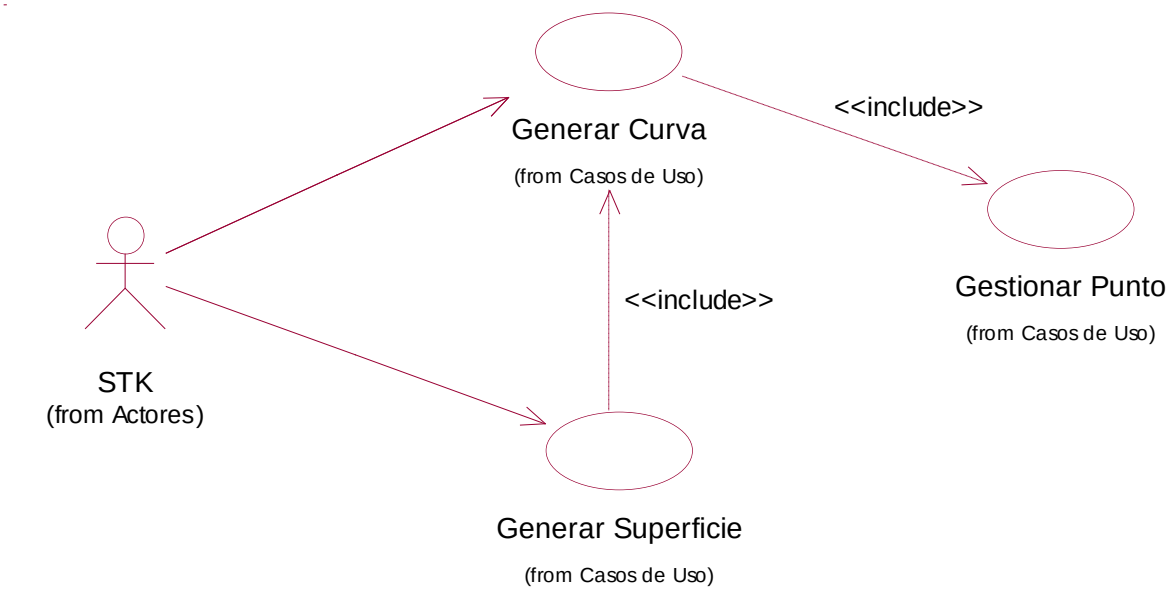

Figura 21 Diagrama de Casos de Uso

3.5.4 Especificación de los casos de usos en formato expandido

La especificación de los casos de uso en formato expandido constituye una descripción textual de los mismos que explica de manera detallada la realización de cada caso de uso y sirve como complemento para un mejor entendimiento de los diagramas.

Tabla 2 CU "Generar Curva"

Nombre del CU	Generar Curva
Actores	STK
Propósito	Generar una lista de puntos a partir de los puntos de control.
Resumen:	El caso de uso comienza cuando el actor introduce la lista de los puntos de control, a partir de los cuales se calculan las coordenadas de los puntos de la curva; que no es más que la sumatoria de cada coordenada de los puntos de control multiplicadas por sus respectivas funciones bases.
Referencias	R1, R2, R4, R5, R6, R7, R8, R9, CU3(*include*)

Precondiciones	Debe estar creada la lista de puntos de control.
Curso normal de los eventos:	
Acción del actor	**Respuesta del sistema**
1- Introduce la lista de puntos de control 3D.	
	2- Se calcula la cantidad **N** de puntos de control de la lista.
	3- Se reserva memoria para **N** filas para la matriz de puntos de control.
	4- Se crea una matriz de **N** filas por 3 columnas.
	5- Se recorre la lista de puntos de control y se introducen las coordenadas **x,y,z** de cada punto de control en la matriz creada anteriormente.
	6- Se determina el valor del parámetro **t** de la función B-spline, 0 < **t** < 1.
	7- Se calculan las funciones bases B-spline según el parámetro **t**.
	8- Se calculan las coordenadas **x**, **y**, **z** de cada uno de los puntos de la curva.
	9- Se crea un nuevo punto con los valores de **x**, **y**, **z** calculados anteriormente.
	10- Se adiciona el punto a una lista.
	11- Se retorna la lista.
Poscondiciones	Generada la lista de los puntos que componen la curva.

Tabla 3 CU "Generar Superficie"

Nombre del CU	Generar Superficie
Actores	STK
Propósito	Generar una malla a partir de dos curvas.
Resumen:	Se inicia cuando el actor solicita crear una superficie pasándole la lista de puntos de control y el ancho de la huella **h**, a partir de los cuales se generará una malla de triángulos que se aproxime a la superficie deseada.
Referencias	R10, R11, R12, R13, R14, CU1(*include*).
Precondiciones	Deben estar creadas 2 curvas.
Curso normal de los eventos:	
Acción del actor	**Respuesta del sistema**
1- Solicita crear una superficie pasándole la lista de puntos de control y el ancho de la huella **h.**	
	2- Se trasladan los puntos de control la distancia **h**/2 en el sentido positivo del eje conveniente.
	3- Se trasladan los puntos de control la distancia **h**/2 en el sentido negativo del eje seleccionado.
	4- Se crean dos nuevas listas de puntos de control con los puntos trasladados.
	5- Se genera para cada una de las listas de puntos de control creadas, una lista de puntos de la curva.
	6- Se recorren ambas listas de puntos de las curvas, donde se seleccionan puntos, con los que se crea una nueva lista de puntos, en dicha lista cada tres puntos se forma un triángulo, de esta manera queda

	generada la malla que aproxima a la superfice deseada.
Poscondiciones	Superfice Creada

Tabla 4 CU "Gestionar Punto"

Nombre del CU	Gestionar Punto
Actores	STK
Propósito	Permitir las operaciones necesarias con los puntos.
Resumen:	Se inicia cuando el actor solicita crear, adicionar, trasladar o eliminar punto, y finaliza cuando el sistema aplica al punto deseado el requerimiento solicitado por el actor.
Referencias	R1,R2,R3,R5,R6,R7
Precondiciones	Debe existir el punto.
Curso normal de los eventos:	
Acción del actor	**Respuesta del sistema**
1- Solicita crear, adicionar, trasladar o eliminar punto.	
	2- Según la funcionalidad solicitada por el actor se ejecuta alguna de las siguientes acciones: a) Se decide crear punto, ir la sección "Crear Punto". b) Se decide adicionar punto, ir la sección "Adicionar Punto".

	c) Se decide trasladar punto, ir la sección "Trasladar Punto". d) Se decide eliminar punto, ir la sección "Eliminar Punto".
Sección: Crear Punto	
Acciones del Actor	**Respuesta del Sistema**
1- Solicita crear un punto.	
	2- Se crea un punto con las coordenadas **x**,**y**,**z**.
Sección: Adicionar Punto	
Acciones del Actor	**Respuesta del Sistema**
1- Solicita adicionar un punto.	
	2- Se adiciona el punto a una lista.
Sección: Trasladar Punto	
Acciones del Actor	**Respuesta del Sistema**
1- Solicita trasladar un punto.	
	2- Se selecciona el eje donde se trasladaran los puntos.
	3- Se suma una distancia dada a la coordenada correspondiente al eje escogido
Sección: Eliminar Punto	
Acciones del Actor	**Respuesta del Sistema**
1- Solicita eliminar un punto.	
	2- Se busca el punto deseado.
	3- Se elimina el punto.
Poscondiciones	Punto Gestionado.

Capítulo 4: Diseño e Implementación del Sistema

Introducción

La primera parte del capítulo encierra el diagrama de clases del sistema, propuesto como resultado del refinamiento de las etapas anteriores. A continuación se describen cada una de las clases del diseño y se presentan los diagramas de secuencia de los casos de uso.

También se da a conocer los estándares de codificación. Se realiza el paso del diseño de clases a la creación de componentes físicos, que se traducen en ficheros .h y .cpp correspondiente a la implementación en C++, los cuales se muestran a través del diagrama de componentes. Además se exponen los resultados obtenidos.

4.1 Diagrama de clases del diseño

En el diagrama de diseño que se muestra en la figura 22 se representan las clases de diseño y sus relaciones para dar solución al problema de este trabajo. La nomenclatura utilizada para el diagrama de clases, se explica en el epígrafe “Estándares de codificación” de este capítulo.

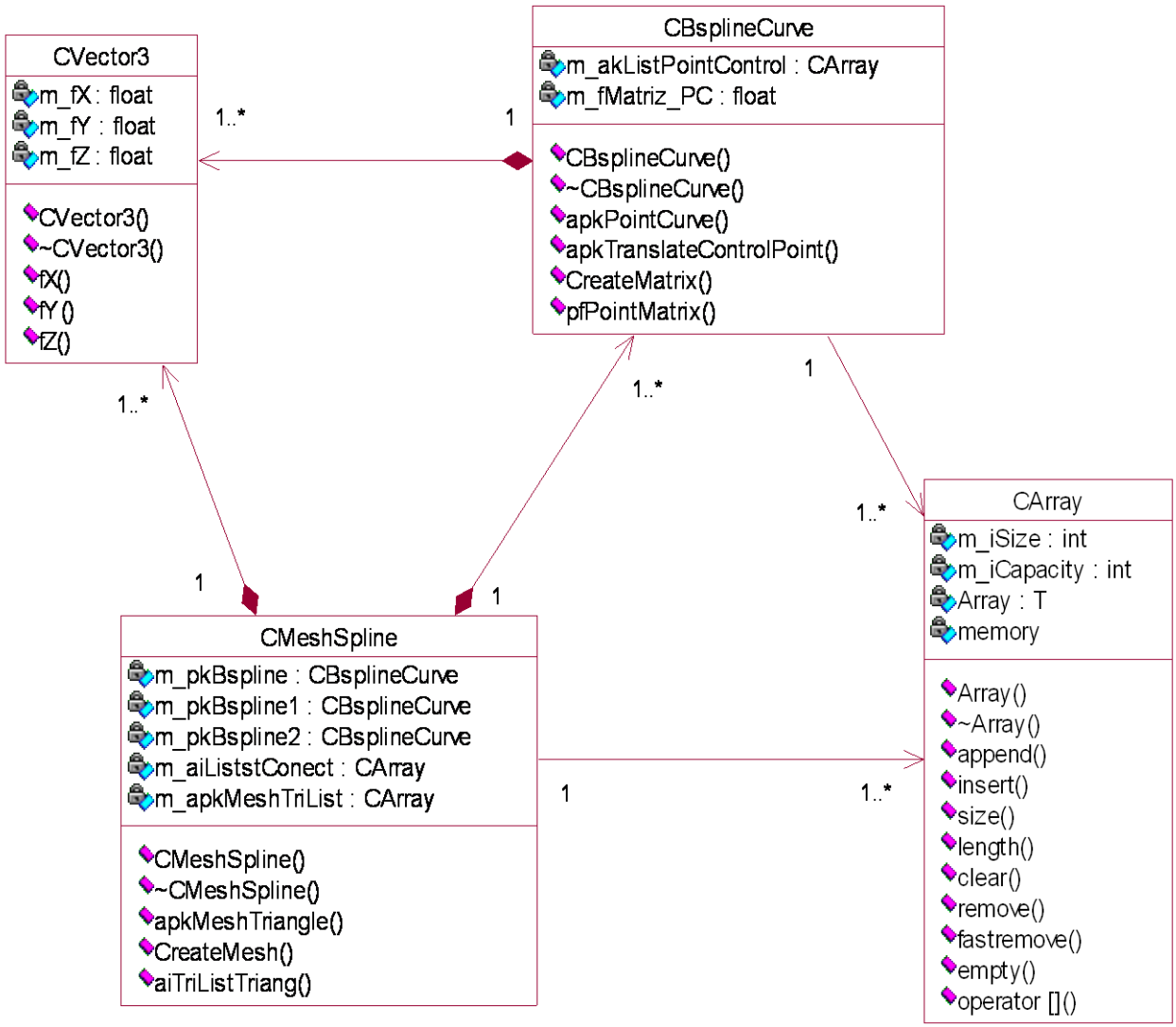

Figura 22 Diagrama de clases del Diseño

4.2 Descripción de las clases del diseño

La descripción de las clases de diseño describe los atributos y métodos que tiene una clase, brindando la información necesaria para lograr un completo entendimiento de la misma. De esta manera el sistema queda listo para ser implementado.

Tabla 5 "Descripción de la clase CMeshSpline"

Nombre:	CMeshSpline	
Tipo de clase:	Controladora	
Atributo		**Tipo**
m_pkBspline		CBsplineCurve*
m_pkBspline1		CBsplineCurve*
m_pkBspline2		CBsplineCurve*
m_apkMeshTriList		Array<CVector3*>
m_aiListstConect		Array<CVector3*>
Para cada responsabilidad:		
Nombre:	apkMeshTriangle	
Descripción:	Función que se encarga, pasándole la lista de puntos de control y el ancho de la huella, de devolver la lista de los vértices de los triángulos con los que se construirá la malla.	
Nombre:	CreateMesh	
Descripción:	Función que se encarga, pasándole dos listas de vértices, de crear la lista de los puntos de los triángulos de la malla.	
Nombre:	aiTriListTriang	
Descripción:	Función que se encarga de devolver una lista de enteros donde cada 3 indican un triángulo.	

Tabla 6 "Descripción de la clase CBsplineCurve"

Nombre:	CBsplineCurve	
Tipo de clase:	Entidad	
Atributo		**Tipo**
m_apkListPointControl		Array<CVector3*>
m_aafMatrix_CP		float **
Para cada responsabilidad:		
Nombre:	apkPointCurve	
Descripción:	Función que se encarga de devolver la lista de los puntos de la curva.	
Nombre:	apkTranslateControlPoint	
Descripción:	Función que se encarga, pasándole el ancho de la huella, de devolver la lista de los puntos de la curva trasladados en el eje **Y**.	
Nombre:	CreateMatrix	
Descripción:	Función que se encarga, de crear la matriz de los puntos de los puntos de control.	
Nombre:	pfPointMatrix	
Descripción:	Función que se encarga, pasándole una posición, de devolver la matriz en esa posición.	

Tabla 7 "Descripción de la clase CVector3"

Nombre:	CVector3	
Tipo de clase:	Entidad	
Atributo		**Tipo**
m_fX		float
m_fY		float
m_fZ		float
Para cada responsabilidad:		
Nombre:	`fX()`	
Descripción:	Función que se encarga, de devolver la coordenada X del punto de control.	
Nombre:	`fY()`	
Descripción:	Función que se encarga, de devolver la coordenada Y del punto de control.	
Nombre:	`fZ()`	
Descripción:	Función que se encarga, de devolver la coordenada Z del punto de control.	

4.3 Diagramas de secuencia

En el presente epígrafe se muestran los diagramas de secuencia correspondientes a cada uno de los escenarios de los casos de uso, mostrando las interacciones entre objetos, ordenadas en secuencia temporal.

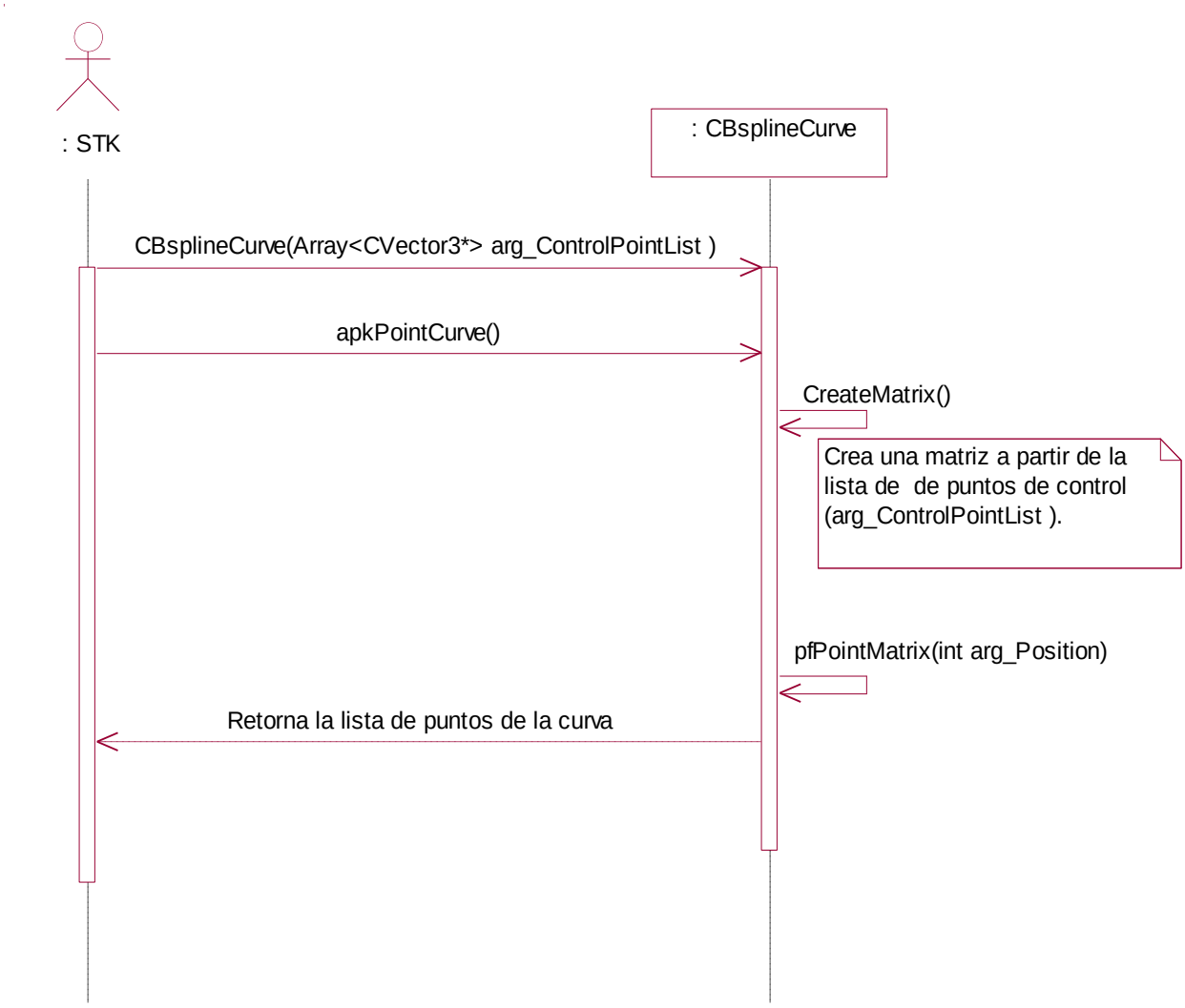

Figura 23 Diagrama de Secuencia "Generar Curva"

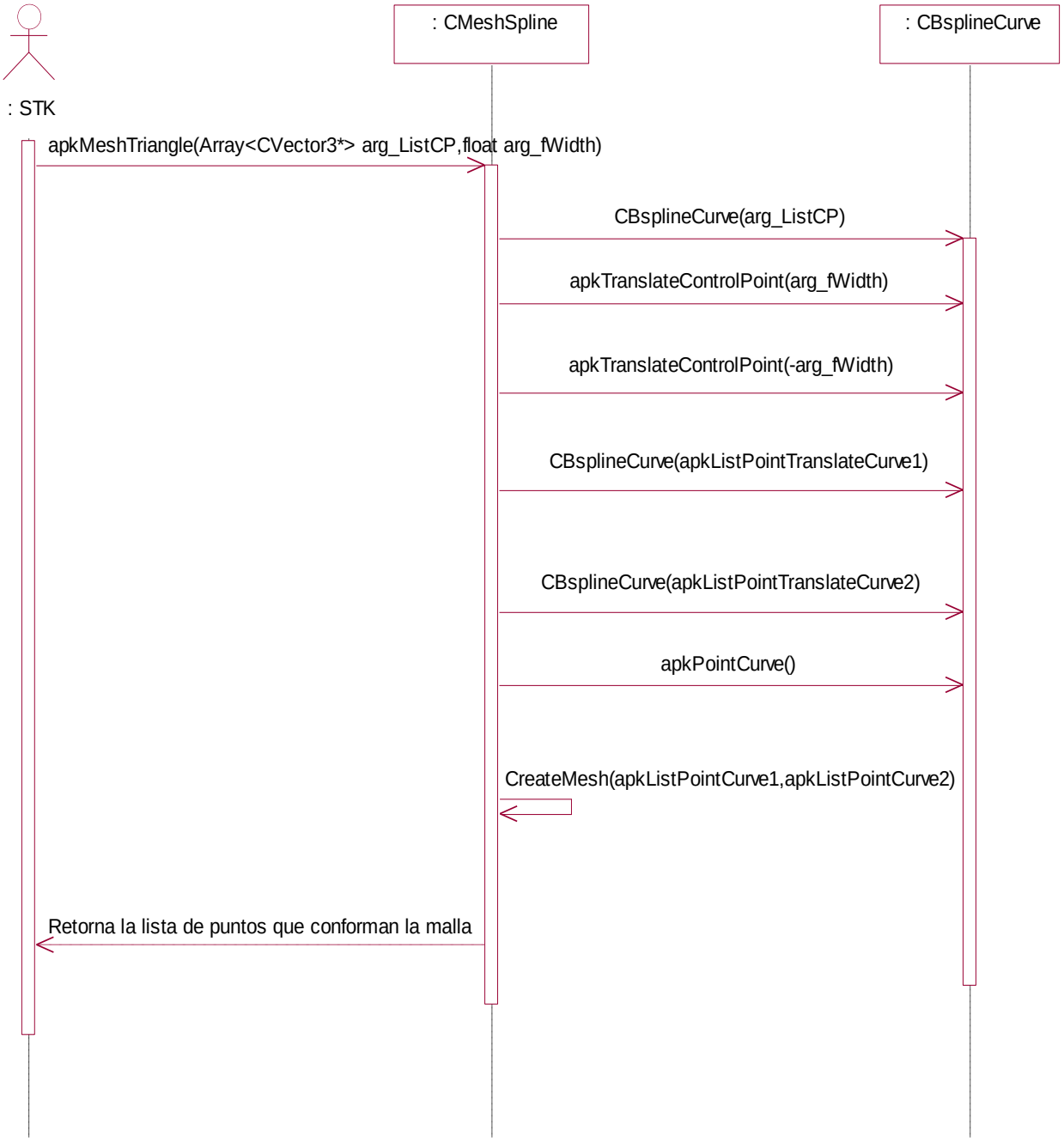

Figura 24 Diagrama de Secuencia "Generar Superficie"

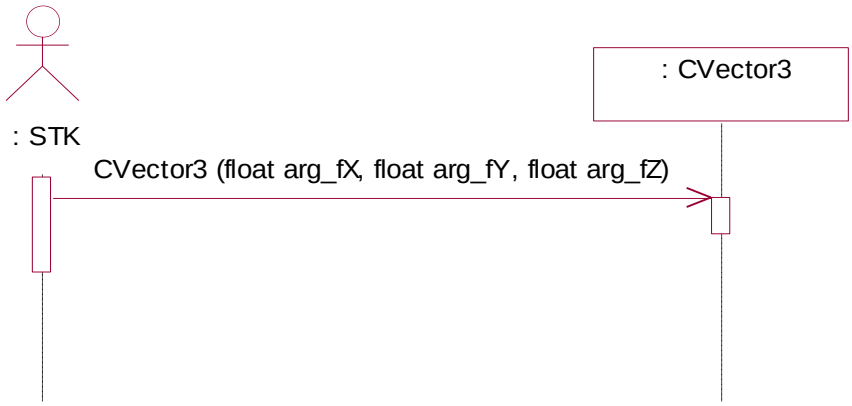

Figura 25 Diagrama de Secuencia "Crear Punto"

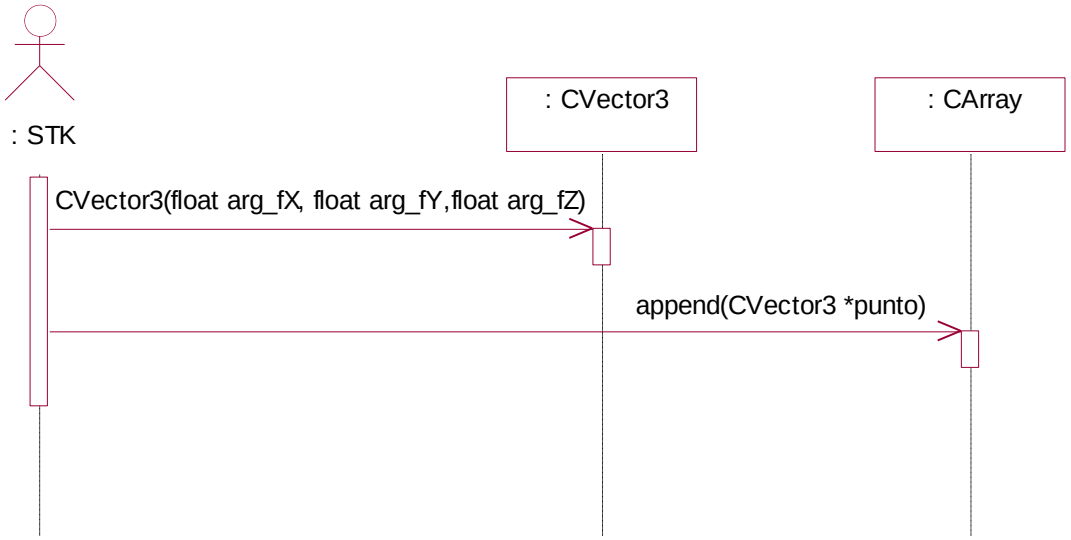

Figura 26 Diagrama de Secuencia "Adicionar Punto"

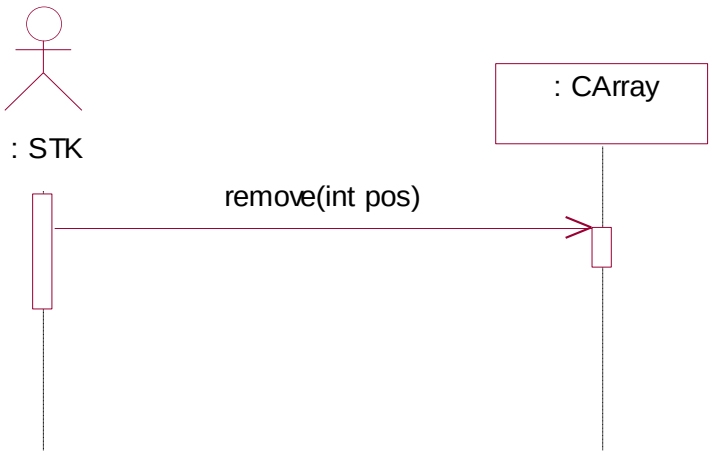

Figura 27 Diagrama de Secuencia "Eliminar Punto"

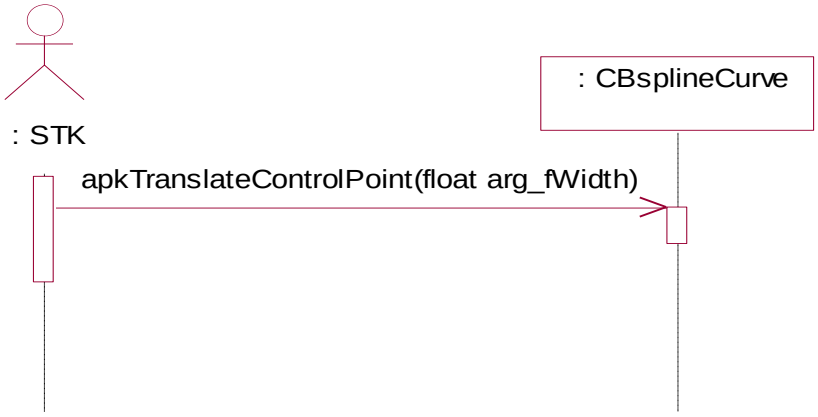

Figura 28 Diagrama de Secuencia "Trasladar Punto"

4.4 Estándares de codificación

El código del módulo sigue algunos estándares propuestos por el grupo de desarrollo (respetando los estándares de codificación para C++). Está programado en inglés, debido que las palabras son simples, no se acentúan y es un idioma muy difundido en el mundo informático.

El conocimiento de los estándares seguidos para el desarrollo de la misma permitirá un mayor entendimiento del código.

Nombre de los ficheros:

Se nombrarán los ficheros .h y .cpp de la siguiente manera:

STKNameOfUnits.cpp

Se usará **GT** para identificar el nombre de la herramienta (en definición!!)

Constantes:

Las constantes se nombrarán con mayúsculas, utilizándose el "_" para separar las palabras: MY_CONST_ZERO = 0;

Tipos de datos:

Los tipos se nombrarán siguiendo el siguiente patrón:

Enumerados: enum **E**MyEnum {**ME**_VALUE, **ME**_OTHER_VALUE};

Indicando con "**E**" que es de tipo enumerado. Nótese que las primeras letras de las constantes de enumerados son las iniciales del nombre del enumerado. Véase otro ejemplo:

enum **E**Node**T**ype {**NT**_GEOMETRYNODE,…};

Estructuras: struct **S**MyStruct {…};

Indicando con "**S**" que es una estructura. Las variables miembros de la estructura se nombrarán igual que en las clases, leer más adelante.

Clases: class **C**ClassName;

Indicando con "**C**" que es una clase

Declaración de variables:

Los nombres de las variables comenzarán con un identificado del tipo de dato al que correspondan, como se muestra a continuación. En el caso de que sean variables miembros de una clase, se le antepondrá el identificador "m_" (en minúscula), si son globales se les antepondrá la letra "g", y en caso de ser argumentos de algún método, se les antepondrá el prefijo "arg_".

Tipos simples:

bool **b**VarName;

int **i**Name;

unsigned int **ui**Name;

float **f**Name;

char **c**Name;

char* **ac**Name; // arreglo de caracteres

char* **pc**Name; // puntero a un char

char** **aac**Name; // bidimensional

char** **apc**Name; // arreglo de punteros

bool **m_b**MemberVarName; //variable miembro

char **g**cGlobalVarName; //variable global, no se le antepone “”

short **s**Name;

Instancias de tipos creados:

EMyEnumerated **e**Name;

SMyStructure **k**Name;

CClassName **k**ObjectName;

CClassName* **pk**Name; //puntero a objeto

CClassName* **ak**Name; //arreglo de objetos

CClassName* **ak**Name; // variable miembro de clase

IMyInterface* **pI**Name; //puntero interfaces

Métodos

En el caso de los métodos, se les antepondrá el identificador del tipo de dato de devolución, y en caso de no tenerlo (void), no se les antepondrá nada. Solamente los constructores y destructores comenzarán con “”.

En el caso de los argumentos se les antepone el prefijo “arg_”

Constructor y destructor:

CClassName (bool **arg**_bVarName, float& **arg**_fVarName);

~**C**ClassName ();

Funciones:

bool bFunction1 (…);

int* **pi**Function2 (…);

CClassName* **pk**Function3 (…);

Procedimientos:

void Procedure4 (…);

Métodos de acceso a miembros

Los métodos de acceso a los miembros de las clases no se nombrarán “Gets” y “Sets”, sino como los demás métodos, pero **con el nombre** de la variable a la que se accede y sin “m_”:

int **i**MyVar; //variable

Obtención del valor:

```
int iMyVar();
{
  return iMyVar;
}
```

Establecimiento del valor:

```
void MyVar(char* arg_iMyVar)

{

  iMyVar = arg_iMyVar;

}
```

Obtención y* establecimiento *del valor:

```
int& iMyVar();

{

   return iMyVar;

}
```

4.5 Diagrama de despliegue

El diagrama de despliegue muestra las relaciones físicas entre los componentes *hardware* y *software* en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes *software* (procesos y objetos que se ejecutan en ellos). En el caso de este trabajo el componente que utilizará en tiempo de compilación será una computadora (figura 29).

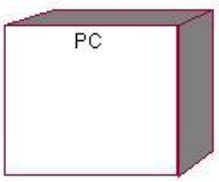

Figura 29 Diagrama de despliegue

4.6 Diagrama de componentes

El diagrama de componentes describe cómo se implementan los elementos del modelo de diseño, por ejemplo las clases de diseño, en términos de archivos de código fuente y ejecutables, el diagrama de componentes para el presente trabajo se muestra a continuación.

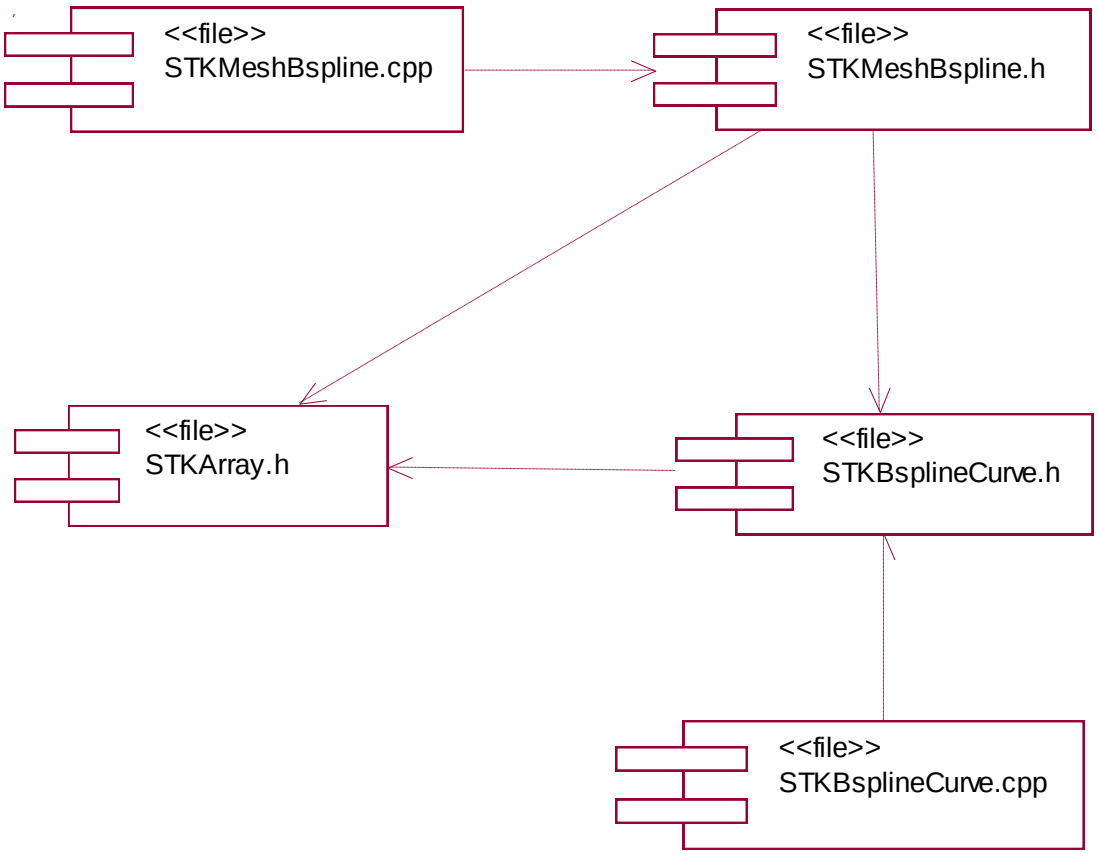

Figura 30 Diagrama de Componentes

4.7 Resultados

Como culminación del trabajo de diploma, se realizó un DEMO para demostrar y comprobar que del algoritmo implementado se obtenían los resultados esperados, y de esta manera se daba cumplimiento al objetivo propuesto en el mismo. A continución se muestran las imágenes obtenidas durante la ejecución del DEMO.

En la primera imagen(figura 31) se puede observar la ubicación de los puntos de control, que definen la forma de la superficie que se desea generar.

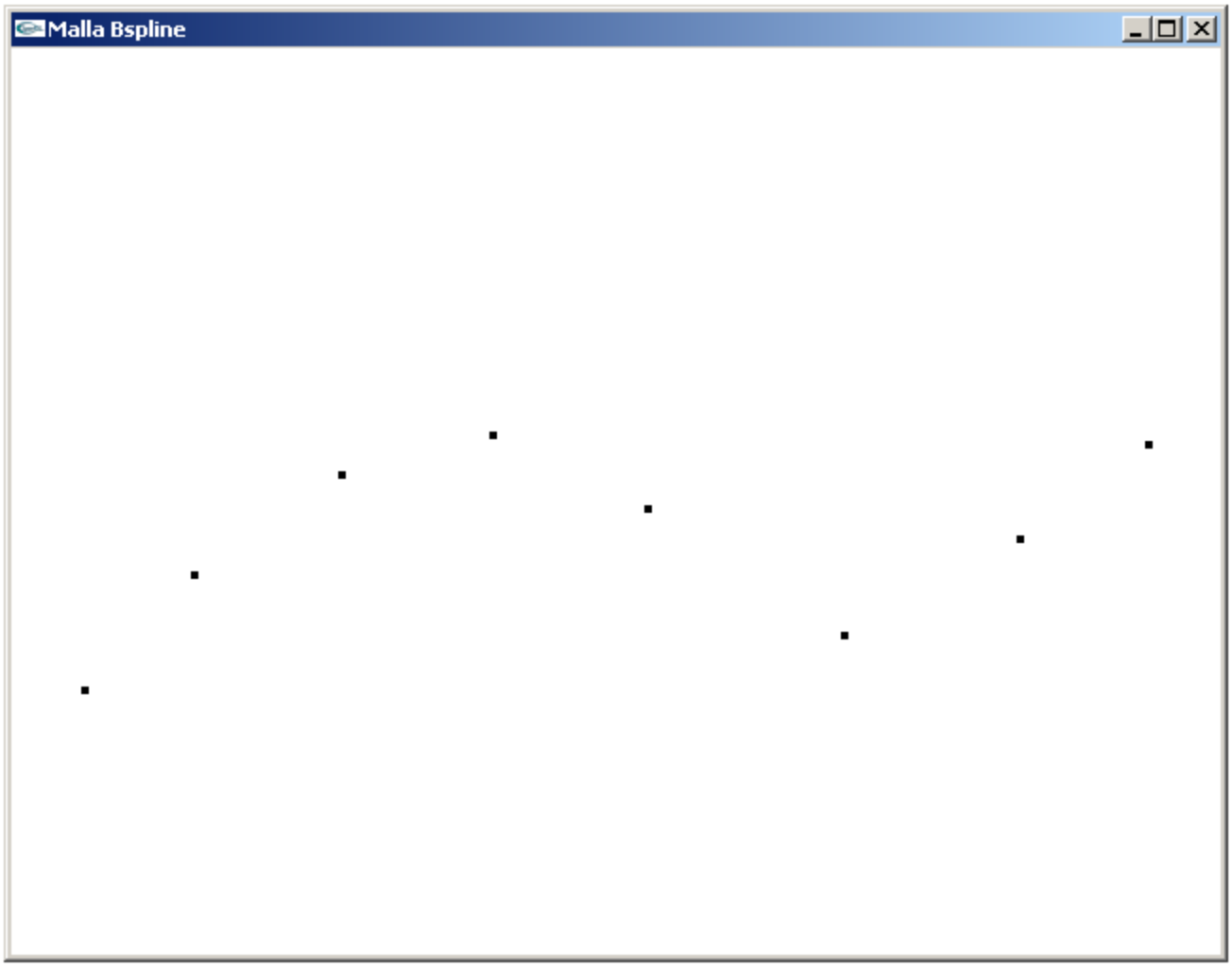

Figura 31 Puntos de control

En la figura 32 se muestra la malla de triángulos que define la forma de la huella deseada, pudiéndose comprobar que se obtuvo el resultado esperado.

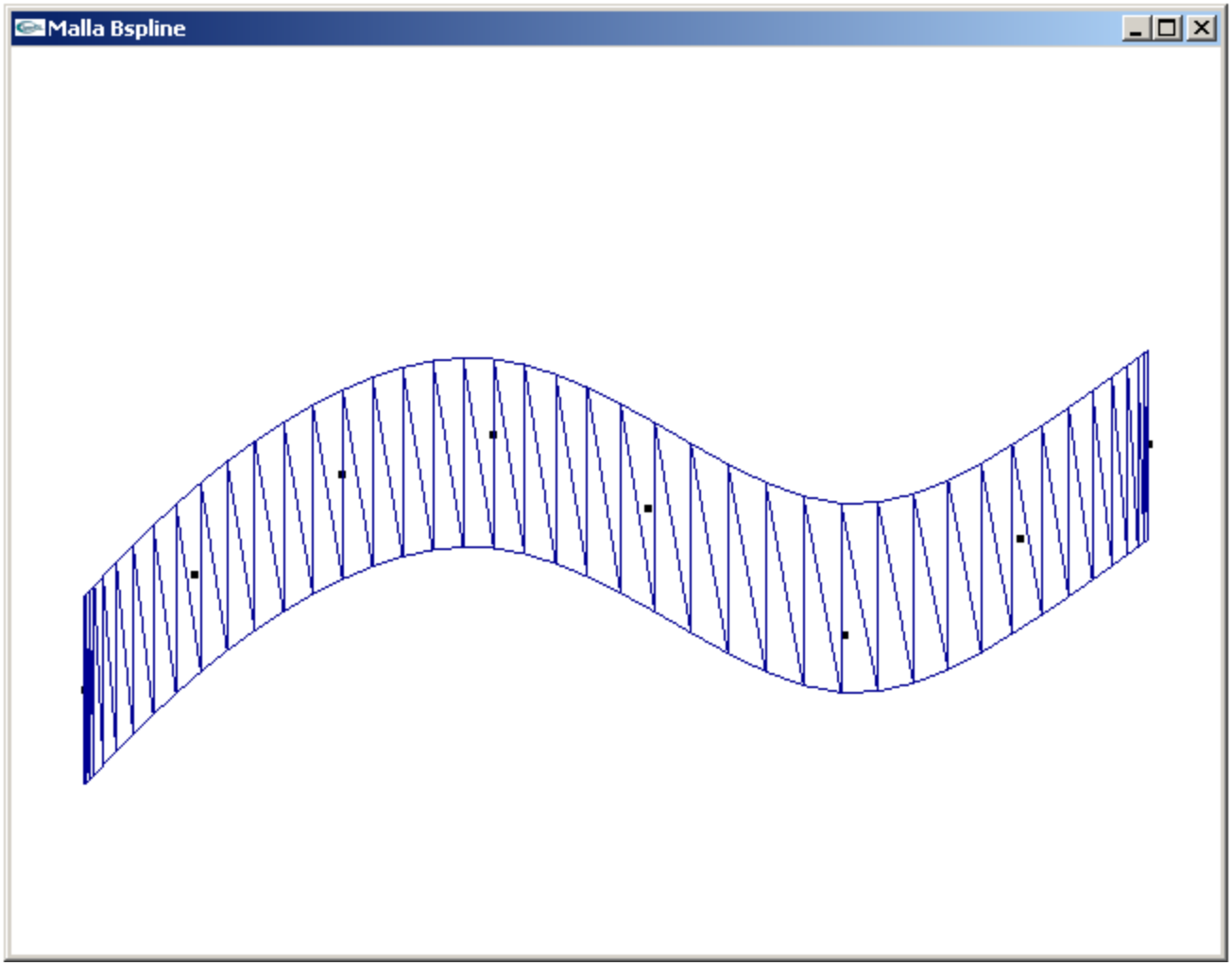

Figura 32 Malla B-spline

Por último se muestra la malla texturizada, mediante el mapeado **uv**, demostrando que la forma en que están organizados los puntos de la malla permite fácilmente poder hacer la asignación de los puntos de textura a los puntos de la malla, de manera que esta quede continua y perfectamente acoplada a la malla correspondiente.

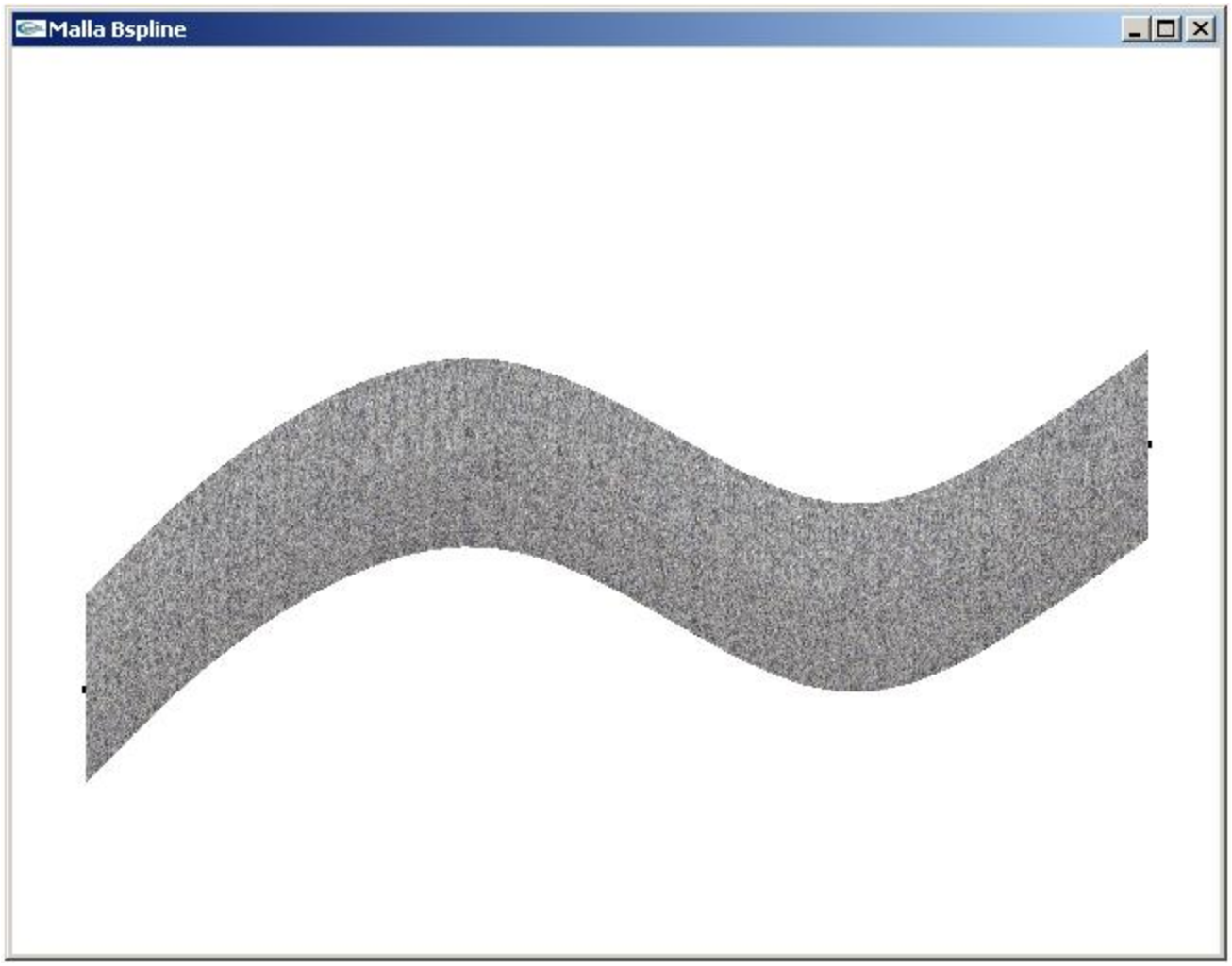

Figura 33 Malla texturizada

Conclusiones

Para dar cumplimiento a los objetivos de este proyecto, en correspondencia con las exigencias del mismo, fue necesario realizar un estudio de los métodos y técnicas para la generación de efectos de huellas en Entornos de Realidad Virtual, específicamente, lo relacionado con curvas y superficies que están definidas por un conjunto de puntos en el espacio. De esta manera se logró diseñar e implementar un módulo de clases mediante el cual se logra la generación de huellas con una adecuada precisión, logrando así una solución factible al problema planteado en el trabajo.

Después se realizó el proceso de Ingeniería del Software utilizando el Proceso Unificado del Software (RUP) como metodología de desarrollo, con los artefactos de RUP, se llevó a cabo la propuesta y consolidación de la solución. El módulo obtenido está concebido para poder adaptarlo fácilmente a la herramienta en desarrollo (SceneToolKit) sin que sufra su estructura, así como la incorporación de nuevas funcionalidades, además aunque la solución esté orientada para aplicaciones en Entornos de Realidad Virtual puede ser usada perfectamente en cualquier tipo de aplicación.

Recomendaciones

Se recomiendan los siguientes aspectos al trabajo:

- Realizar las modificaciones necesarias para lograr que el módulo sea totalmente acoplable a la herramienta en desarrollo (SceneTollKit).
- Usar el módulo en otras aplicaciones para otras funcionalidades que no sea necesariamente la Realidad Virtual, como pueden ser aplicaciones de diseño asistido por computadoras.
- Incorporar otros tipos de efectos de huellas como es impacto de disparo en paredes.

Referencias Bibliográficas

[1]. **Torres, J.C.** *Universidad de Granada.* [En línea] [Citado el: 9 de enero de 2008.] http://lsi.ugr.es/~jctorres/cad0405/teoria/resumenTema3.pdf.

[2]. **Bengoa Moreno, Alberto, y otros.** [En línea] [Citado el: 16 de abril de 2008.] http://sabia.tic.udc.es/gc/teoria/Curvas/HTML/TrabajoGC.htm.

[3]. **Calvo, Néstor.** Curvas y Superficies para modelado geométrico. *Computación Gráfica. FICH-UNL.* [En línea] [Citado el: 3 de diciembre de 2007.] http://cimec.org.ar/twiki/pub/Cimec/ComputacionGrafica/curvas.doc.pdf.

[4]. [En línea] [Citado el: 12 de marzo de 2008.] http://wgpi.tsc.uvigo.es/~xulio/Web-TMM/Clase8Splines/c4%20cur%20sp2.pdf.

[5]. **Jambrina, Leonardo Fernández.** [En línea] [Citado el: 9 de febrero de 2008.] http://debin.etsin.upm.es/~leonardo/tema2.htm.

[6]. **Ribó, Ramón.** Descripción matemática de las NURBS. [En línea] 2 de abril de 2003. [Citado el: 9 de enero de 2008.] ftp://gid.cimne.upc.es/pub/ramsan/DescripcionDeLasNURBS.pdf.

[7]. **Lengyel, Eric.** *Mathematics for 3D Game Programming and Computer Graphics (2 Edition).* s.l. : Charles Rives Media.

[7.1] Capítulo 15: Curves and Surfaces.

[8]. [En línea] [Citado el: 9 de enero de 2008.] http://exp-grafica.uma.es/Profesores/www-jrad/document/sist-avanzados/funcDAO3.pdf.

[9]. **Morales, Ernesto Cuartas.** *Universidad Nacional de Colombia.* [En línea] 2004. [Citado el: 11 de febrero de 2008.] http://pci.unalmzl.edu.co/Tesis/ernesto.pdf.

[10]. **Wayne, T.** *The NURBS Book(2nd edition).* s.l. : Springer-Verlag, New York, 1997.

[11]. **Gerald, Farin.** *Curves and Surfaces for CAGD(5 edition).* s.l. : Morgan Kaufmann Publishers, 2002.

[12]. **Sanz Sanfructuoso, Daniel y Carrasco de Pedro, Oscar.** [En línea] [Citado el: 20 de enero de 2008.] http://gsii.usal.es/~corchado/igrafica/descargas/temas/Tema11.pdf.

[13]. **Calvo, Néstor, y otros.** *Computación Gráfica. FICH-UNL.* [En línea] 11 de octubre de 2007. [Citado el: 20 de enero de 2008.] http://www.cimec.org.ar/twiki/pub/Cimec/ComputacionGrafica/texturas.ppt.pdf.

Bibliografía Consultada

1. **Remolar, Inmaculada.** Modelado geométrico II. *http://graficos.uji.es.* [En línea] [Citado el: 9 de enero de 2008.] http://graficos.uji.es/grafica/Documentos/8%20ModeladoII.pdf.

2. **Osorio, Juan D., Prieto, Flavio A. y Osorio, Gustavo A.** *Universidad Nacional de Colombia.* [En línea] 2004. [Citado el: 9 de enero de 2008.] http://redalyc.uaemex.mx/redalyc/pdf/496/49614207.pdf.

3. *Parametric Curves And Surfaces.* [Online] [Cited: marzo 3, 2008.] http://www.robthebloke.org/opengl_programming.html#3.

4. **Delgado Olmos, Ángel H. y Márquez García, M. Luisa.** Recorte y agregación de superficies clásicas. Tratamiento y aplicaciones. *Universidad de Granada. España.* [En línea] [Citado el: 4 de diciembre de 2007.] http://w3.iec.csic.es/ursi/articulos_gandia_2005/articulos/RA/280.pdf.

5. **Cortés Parejo, José y Cordero Valle, Juan Manuel.** *CURVAS Y SUPERFICIES PARA MODELADO GEOMÉTRICO.* s.l. : Ra-ma, 2002. ISBN: 8478975314.

6. **Prautzsch, Hartmut, Boehm, Wolfgang and Paluszny, Marco.** *Métodos de Bézier y B-splines.* s.l. : Universitätsverlag Karlsruhe, 2005. ISBN: 3-937300-47-3.

7. **García Paredes, Ignacio y De Cáceres García, José Luis.** [En línea] 2003. [Citado el: 8 de octubre de 2007.] http://gsii.usal.es/~corchado/igrafica/apuntes2002-2003/tema3/Presentacion/Graficos%203D.ppt.

8. **Recopilación de Autores.** *Game Programming Gems I.* s.l. : Charles River Media, 2000.
   8.1 Capitulo 11: Curves and Surfaces. pág.: 575

Glosario de Términos

A:

Aproximación: La curva se acerca a los puntos de control.

******

C:

Control global: La modificación de un punto de control solo afecta a las zonas de la curva o superficie próximas al punto modificado.

Control local: La modificación de un punto control afecta a toda la curva o superficie.

******

G:

Grado: Es un número. Este número normalmente es 1, 2, 3 o 5 pero puede ser cualquier número entero positivo.

******

I:

Interpolación: La curva pasa por los puntos de control.

******

K:

knots: Son los puntos de enlace entre los distintos trozos de la curva se les suele llamar nudos o nodos.

******

M:

Malla: Forma de representar un modelo a partir de polígonos. Colección de vértices, aristas y polígonos conectados de forma que cada arista es compartida como máximo por dos polígonos.

Matriz: Arreglo de elementos.

******

O:

Orden: Es igual a uno más el grado de los polinomios que generan la curva (en el caso de curvas de Bézier el orden coincide con el número de puntos de control necesarios para generar la curva).

P:

Píxel: Abreviatura de “picture element”. Es la menor unidad de información de una imagen digital.

Polígono convexo: Frontera del polígono convexo que encierra el conjunto de puntos de control.

Programación Orientada a Objetos: POO es una filosofía de programación que se basa en la utilización de objetos. El objetivo de la POO es "imponer" una serie de normas de desarrollo que aseguren y faciliten la mantenibilidad y reusabilidad del código.

Puntos de control: Conjunto de puntos que indican la forma general de la curva.

******

R:

Realidad Virtual: Simulación de un medio ambiente real o imaginario que se puede experimentar visualmente en tres dimensiones. La realidad virtual puede además proporcionar una experiencia interactiva de percepción táctil, sonora y de movimiento.

******

S:

Sistema de Realidad Virtual: Sistema informático interactivo que ofrece una percepción sensorial al usuario de un mundo tridimensional sintético que suplanta al real.

T:

Texel: Del inglés, texture element. Unidad mínima de una textura aplicada a una superficie.

Textura: Imagen que sirve de "piel" a los modelos en un mundo virtual.

Texturizar: Aplicar texturas.

******

V:

Virtual: Término utilizado para hacer referencia a algo que no tiene existencia física o real, solo aparente.

Índice de Figuras y Tablas

Figuras

Tablas