UNIVERSIDAD NACIONAL DEL LITORAL
FACULTAD DE INGENIERÍA Y CIENCIAS HÍDRICAS

Tecnicatura en diseño
y programación de videojuegos

UNLVIRTUAL

MANIPULACION DE OBJETOS EN 2D
Unidad 6: Curvas

Docente
Walter Sotil

Tutores
Emmanuel Rojas Fredini, Cristian Yones

CONTENIDOS

1. Introducción

En esta unidad estudiaremos cómo se representan distintos tipos de curvas y su uso en videojuegos.

Las curvas pueden ser extremadamente útiles para muchas aplicaciones. Por ejemplo, para definir la trayectoria de objetos, como proyectiles, en un juego; para determinar una función que describa la velocidad de un auto, o cualquier otro atributo, así como para dibujar curvas en pantalla. Otros usos consisten en dibujar desde fuentes (TrueType Fonts), hasta representar cabellos humanos.

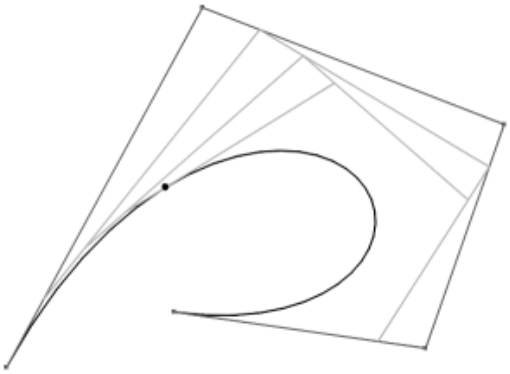


Figura 1. Ejemplo de curva

En esta unidad explicaremos qué es una curva y cómo se puede y nos conviene representarla. Luego explicaremos en detalle un tipo de curvas, conocidas como *curvas de Bézier,* y ahondaremos en su implementación y representación. Por último, explicaremos curvas más flexibles que las de Bézier, conocidas como *Splines.*

Curva
En computación gráfica y en CAGD (Computer Aided Geometric Design), una curva es un conjunto continuo y unidimensional de puntos.

Es muy difícil definir con precisión lo que es una curva, que depende -en general- del contexto.

En computación gráfica (CG) y en CAGD (Computer Aided Geometric Design), una curva es un conjunto continuo y unidimensional de puntos, es decir, que todas las coordenadas reales (x, y) varían como funciones continuas de un solo parámetro real, que llamaremos t.

Aquí sólo nos interesaremos por la representación gráfica de una sucesión de puntos y únicamente en casos simples, sin elementos raros.

2. Representación de curvas

La definición analítica de una curva puede hacerse de varios modos y se relaciona directamente con la forma de representarla gráficamente.

Explícita

Seguramente es la primera forma que se nos ocurriría para representar una curva. Se provee un mapeo o función directa de una coordenada hacia las otras. Por ejemplo, para una curva en 2D que posee coordenadas (x, y), podríamos escribir la curva como una función $y(x)$, o al revés, como $x(y)$. Esto nos ofrece la ventaja de evaluar muy fácilmente la posición de la curva, en caso de que tengamos la coordenada

Recordemos que si una función es de la forma $g(h)$, luego se dice que g es la variable dependiente y h es la variable independiente

independiente. Así, podríamos definir una curva de forma parabólica de la siguiente forma:

$$y(x) = x^2$$

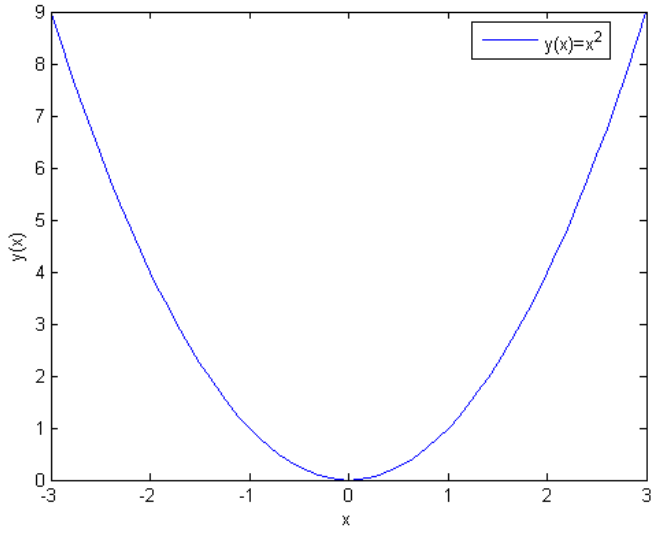


Figura 2. Curva parabólica

Para definir una curva en forma de línea, la ecuación podría ser $y(x) = 3x + 2$. Si restringimos la variación x a un cierto intervalo, podemos generar un segmento.

Definida de esta forma, una curva es muy fácil de graficar, ya que podemos variar sistemáticamente x en el intervalo y obtener los correspondientes valores de y evaluando directamente la función.

El problema de esta forma de representación es que no todas las curvas pueden representarse correctamente. Por ejemplo, para representar una recta vertical, no podríamos hacerlo en la forma $y(x)$, ya que -por definición de función matemática- para cada valor de x sólo puede haber un valor de y. En este caso, la representación debería ser $x(y)$, pero tendríamos el mismo problema que la línea vertical, si la línea a representar es horizontal. Representar un círculo no sería posible, ya que para cada x se tienen dos posibles valores de y.

Resumiendo, la representación explícita es la forma más directa posible de representación gráfica de una curva, pero tiene la desventaja de que es muy poco permisiva en las formas que permite representar.

Representación explícita
Es la forma más directa posible de representación gráfica de una curva, pero tiene la desventaja de que es muy poco permisiva en las formas que permite representar.

Implícita

Es la definición de una relación analítica entre las variables de una curva, pero sin la fórmula para deducir unas en función de otras. Esto significa que un lado de la ecuación tiene las dos componentes, igualado a 0 o a algún valor constante:

Curva en 2D: $\boldsymbol{f(x,y) = 0}$

Por ejemplo, para definir una circunferencia, podemos utilizar la ecuación $x^2 + y^2 = r^2$, donde r es el radio de la circunferencia. Los puntos de la curva serán todos aquellos puntos que satisfagan esta ecuación. Para curvas estándar (rectas, círculos, planos, toroides, etc.), este tipo de definición es más directa y permite visualizar y modificar sencillamente parámetros importantes y específicos de cada curva (radio en un círculo, distancia al origen en un plano, etc.).

Representación implícita
Es la definición de una relación analítica entre las variables de una curva, pero sin la fórmula para deducir unas en función de otras.

La desventaja reside en que, dada la ecuación, la curva puede ser difícil de graficar, ya que no se obtienen los puntos de la curva en forma directa. Por ejemplo, supongamos que no sabemos lo que representa la ecuación $x^2 + y^2 = 1$; para un dado x, digamos $x = 3$, despejamos y en función de x, obteniendo valores complejos ($y = \sqrt{1-9} = \sqrt{-8}$, y una raíz de un número negativo da como resultado un número complejo), sin representación en el espacio real.

Esta forma sólo se utiliza en los casos especiales en que las constantes tienen un significado geométrico.

Cada tipo de curva tiene una ecuación determinada que posee valores que la caracteriza. Por ejemplo, una circunferencia es caracterizada por su radio. Aquí no se enuncian formulas, ya que no son de utilidad para nosotros, sino sólo la forma en que se representa cada tipo de curva.

Paramétrica

Es la forma que utilizamos para definir matemáticamente las curvas. Los valores de cada coordenada se expresan como función de un parámetro independiente escalar, que no son coordenadas:

Curva en 2D:

$$\boldsymbol{x = f(t)}$$

$$\boldsymbol{y = g(t)}$$

Para una curva, estas funciones, dado un valor de t, nos dan un punto de la curva. Si imaginamos que dibujamos la curva con un lápiz en un papel, el parámetro t puede verse como el tiempo y las ecuaciones responden a la pregunta: ¿dónde está el lápiz en el tiempo t?

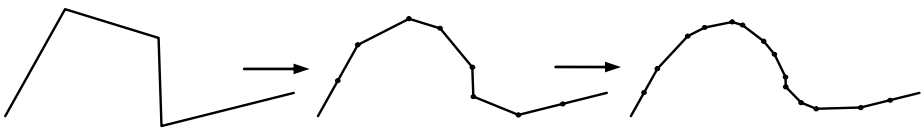
Representación paramétrica
Es la forma que utilizamos para definir matemáticamente las curvas.

Por ejemplo, una recta podría expresarse como $\boldsymbol{x = at}$, $\boldsymbol{y = bt}$ *(los valores que tomen* $\boldsymbol{a}$ *y* $\boldsymbol{b}$ *darán la inclinación de la recta)* y una circunferencia podría expresarse como $\boldsymbol{x = r\,cos(t)}$, $\boldsymbol{y = r\,sin(t)}$.

Esta forma es la preferida en CG por la facilidad con que se obtienen los puntos para graficar y por su generalidad (puede representar todo tipo de curvas).

Generativa o procedural

Esta forma es totalmente diferente a las anteriores, pues la curva es el resultado de un procedimiento o algoritmo; en general, la modificación de una curva simple de partida. Por ejemplo, dividir en tres cada tramo de una poligonal y reemplazar los dos tramos que convergen en un vértice original por el que une los dos nuevos vértices cercanos al original:



Representación generativa o procedural
Esta forma es totalmente diferente a las anteriores, pues la curva es el resultado de un procedimiento o algoritmo; en general, la modificación de una curva simple de partida.

¿Qué representación usaremos?

Por ser la forma más flexible para definir curvas, trabajaremos en la forma paramétrica, que representaremos en forma vectorial:

Curva en 2D: $P(t) = \{x(t),\ y(t)\}$

Estas funciones pueden ser de cualquier tipo; sin embargo, en general, se suelen usar polinomios.

El grado de un polinomio está dado por el exponente mayor que el mismo tenga.

Recordemos que un polinomio vectorial de grado n se puede escribir como:

$$P(t) = \bar{a}t^0 + \bar{b}t^1 + \bar{c}t^2 + \bar{d}t^3 + \cdots + \bar{\alpha}t^n$$

Los elementos $\bar{a}$, $\bar{b}$, etc. son vectores que en general serán puntos en 2D.

3. Curvas de Bézier: generalidades

Se denomina *curva de Bézier* a un método de definición de una curva de forma paramétrica en serie de potencias.

Curva de Bézier
Es un método de definición de una curva de forma paramétrica en serie de potencias.

El método consiste en definir algunos puntos de control, a partir de los cuales se calculan los puntos de la curva a través de unas funciones de peso, también conocidas como *funciones básicas* o *blending functions*. Los puntos de control de la curva determinan la forma de la misma. Dependiendo del orden de los puntos de control, la influencia sobre la curva es diferente:

- La curva siempre comenzará en el primer punto de control y terminará en el último. Es decir, pasa por ambos puntos, lo cual hace que la curva los interpole.
- Todos los puntos de control interiores, entre el primero y el último, *atraen* la curva hacia ellos como si se tratase de magnetos actuando sobre un metal, pero la curva no pasará por esos puntos, sino que sólo los aproximará.

Unicidad
Dada una curva *C* de grado *n*, hay uno y sólo un conjunto de *n+1* puntos de control que la definen como curva de Bézier.

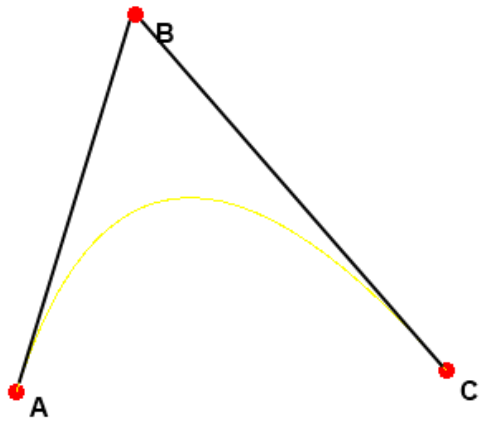


Figura 3. Curva de Bézier

En la *Figura 3* podemos ver una curva de Bézier en amarillo, formada por tres puntos de control: $\bar{A}$, $\bar{B}$ y $\bar{C}$ en color rojo. Podemos ver que la curva pasa por los puntos de control $\bar{A}$ y $\bar{C}$, que son los puntos de control inicial y final, respectivamente.

Por otro lado, el punto de control $\bar{B}$ *atrae* la curva hacia él, pero ésta no pasa por ese punto. Además, en la figura podemos ver la línea negra que une los puntos de control $\bar{A}$, $\bar{B}$ y $\bar{C}$, conocida como *polígono de control.*

Antes de ver cómo crear esta curva usando ecuaciones paramétricas, consideremos un método geométrico para evaluarla, esto es, para calcular los puntos de la curva.

Ante todo, recordemos cómo interpolábamos linealmente entre dos puntos.

La interpolación lineal entre dos puntos es una parametrización de una línea entre ambos. Si los mismos son $\bar{A}$ y $\bar{B}$ y nosotros queremos un punto $\bar{C}$ sobre la recta tal que esté a una proporción t entre los puntos $\bar{A}$ y $\bar{B}$, sabiendo que el vector que va desde $\bar{A}$ hasta $\bar{B}$ es $\bar{B} - \bar{A}$, entonces $\bar{C}$ estará a $t(\bar{B} - \bar{A})$ desde el punto $\bar{A}$, es decir a:

$$\bar{C} = \bar{A} + t(\bar{B} - \bar{A})$$

$\overline{\boldsymbol{C}} = (1-t)\overline{\boldsymbol{A}} + t\overline{\boldsymbol{B}}$

Donde t está en el rango $[0,1]$

El punto $\overline{\boldsymbol{C}}$ está dividiendo el segmento $\overline{\boldsymbol{AB}}$ en un *ratio* o proporción $t:(1-t)$. En la *Figura 4* podemos ver un segmento parametrizado. En este caso, se usó un $t = 0.4$, es decir, que el segmento $\overline{\boldsymbol{AB}}$ está dividido en un ratio $0.4:(1-0.4)$.

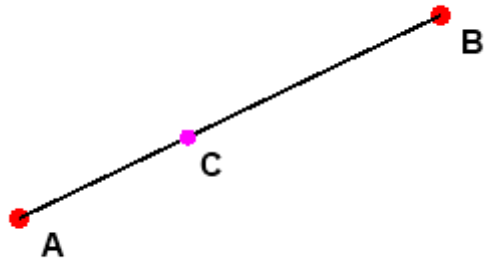


Figura 4. Línea ratio $\boldsymbol{t}:(\boldsymbol{1}-\boldsymbol{t})$

Ahora, retomando la explicación del método gráfico para dibujar la curva, *desarrollaremos una forma que, dado un valor del parámetro de la curva, por ejemplo un valor t dentro de* $[0,1]$*, el método nos dará el punto de la curva que corresponde.*

Lo primero que haremos es tomar el polígono de control, que estará formado por $n+1$ puntos de control, y a cada segmento del polígono lo dividiremos en un ratio $t:(1-t)$, siendo t el parámetro del punto que se desea calcular.

En la *Figura 5,* es posible observar el resultado de este proceso usando un $t = 0.5$, donde los puntos magenta son los recientemente creados. Cabe destacar que en el ejemplo, los puntos de la curva eran $n+1=3$, y que luego de este proceso se obtuvieron como resultado sólo dos puntos, disminuyéndose en uno la cantidad de puntos.

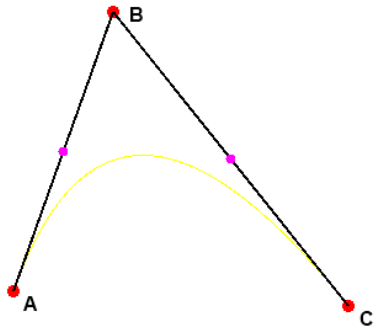


Figura 5. Primer paso

Luego, tomando los últimos puntos calculados, y procediendo del mismo modo en que lo hicimos al comienzo con los puntos de control, trazamos una recta entre cada par de nuevos puntos. A estos segmentos nuevos los dividimos en un ratio $t:(1-t)$, calculando así nuevos puntos. Como se observó en el primer paso, de un grupo de n puntos, ahora se obtuvieron $n-1$ puntos.

En la *Figura 6* podemos ver el resultado de este último paso, usando un $t = 0.5$ en la , que nos da como resultado sólo un punto.

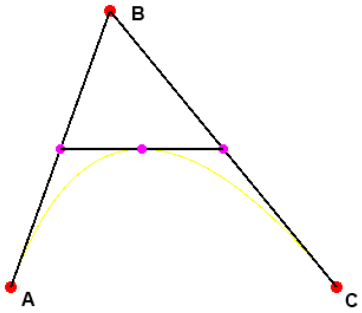


Figura 6. Segundo paso

En el ejemplo que constaba de una curva C con tres puntos de control, podemos ver que el último punto creado en el segundo paso es efectivamente un punto sobre la curva. Más específicamente, es el punto de la curva C que corresponde a $t = 0.5$.

De esta forma, con los pasos que se efectuaron, obtuvimos nuestro método para evaluar la curva para un valor del parámetro t. De modo que, si repetimos los pasos en el ejemplo usando distintos valores de t, obtendremos otros puntos de la curva, con los cuales podremos graficarla, hacer que un objeto la siga, o cualquier uso que le demos a la curva. En la Figura 7 se puede observar la curva de ejemplo evaluada por este método en distintos puntos.

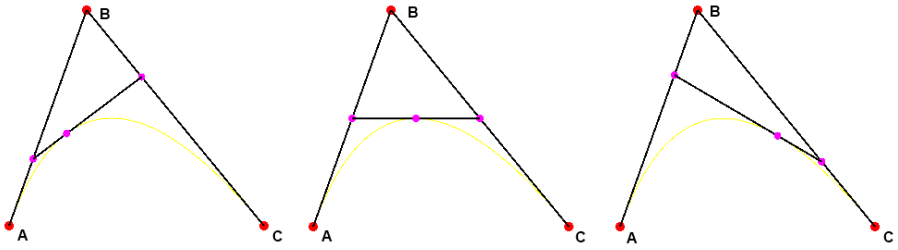


Figura 7. Curva C evaluada en distintos puntos

El algoritmo que acabamos de explicar se conoce como *algoritmo de De Casteljau,* el cual es numéricamente estable para la evaluación de curvas de Bézier (en realidad es un algoritmo para calcular polinomios en forma de Bernstein).

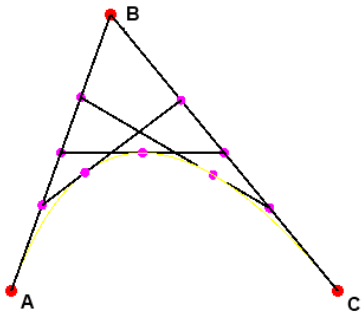


Figura 8. Curva evaluada en varios puntos

Recapitulemos, entonces, los pasos para evaluar la curva mediante el algoritmo de De Casteljau:

- Consideremos una curva de Bézier que tiene puntos de control $\overline{\boldsymbol{A}}$, $\overline{\boldsymbol{B}}$, $\overline{\boldsymbol{C}}$, etc. Conectando los puntos consecutivos, formamos el polígono de control.

- Dividimos cada segmento del polígono, usando siempre el ratio $t:(1-t)$ para el valor de t que se desee evaluar la curva, y conectamos los puntos nuevos que obtuvimos, formando un nuevo polígono. De esta manera, obtenemos un polígono que tiene un segmento menos que el anterior.

- Repetimos el paso anterior hasta que terminamos con un solo punto, que será el punto de la curva que corresponde al parámetro t.

4. Orden y grado de una curva de Bézier

Orden de una curva de Bézier
Es la cantidad de puntos de control que dicha curva posee.

Por razones históricas se llama *orden* de una curva de Bézier a la cantidad de puntos de control que dicha curva posee. Para dos puntos de control, la curva es de segundo orden.

Grado de una curva de Bézier
Es el grado del polinomio de la representación de dicha curva.

El *grado* de una curva es el grado del polinomio de su representación, que veremos más adelante. En estas curvas, el grado es igual a $\boldsymbol{orden-1}$. Entonces, la curva de orden 2 posee grado 1.

Resumiendo:

$$\boldsymbol{orden = n + 1} \qquad (\boldsymbol{orden = grado + 1})$$

Desde ahora en adelante, cuando usemos n nos estaremos refiriendo al grado, tal como acabamos de definirlo.

5. Nemotecnia

Nemotecnia
Se refiere a la forma $\overline{\boldsymbol{P}}_{\boldsymbol{i}}^{\boldsymbol{j}}$ de expresar la curva, donde el superíndice j nos dirá la iteración del algoritmo en la que se obtuvo el punto, en tanto que el subíndice i nos dará el número de punto.

Para realizar los cálculos del algoritmo de De Casteljau es interesante entender un diagrama donde se expresan todos los cálculos a realizar.

Antes de presentarlo, pensemos que la curva posee $n+1$ puntos de control. Luego, que cada punto que se utilice en el cálculo se representará como $\overline{\boldsymbol{P}}_{\boldsymbol{i}}^{\boldsymbol{j}}$ donde el superíndice j nos dirá la iteración del algoritmo en la que se obtuvo el punto, en tanto que el subíndice i nos dará el número de punto. Esto significa que los puntos de control que corresponderán a la iteración número 0 serán los puntos $\overline{\boldsymbol{P}}_{\boldsymbol{i}}^{\boldsymbol{0}}$, donde i tiene valores entre $[0, n]$.

En la próxima iteración del algoritmo, como vimos en esta iteración, deberemos tener n puntos, por ejemplo uno menos que en la iteración anterior; o sea que los puntos serán $\overline{\boldsymbol{P}}_{\boldsymbol{i}}^{\boldsymbol{1}}$, donde i tiene valores entre $[0, n-1]$,y así sucesivamente, hasta que en la iteración número n sólo tendremos un solo punto, el cual será $\overline{\boldsymbol{P}}_{\boldsymbol{0}}^{\boldsymbol{n}}$. Éste será el punto en la curva para el valor paramétrico t usado, es decir, el $C(t)$.

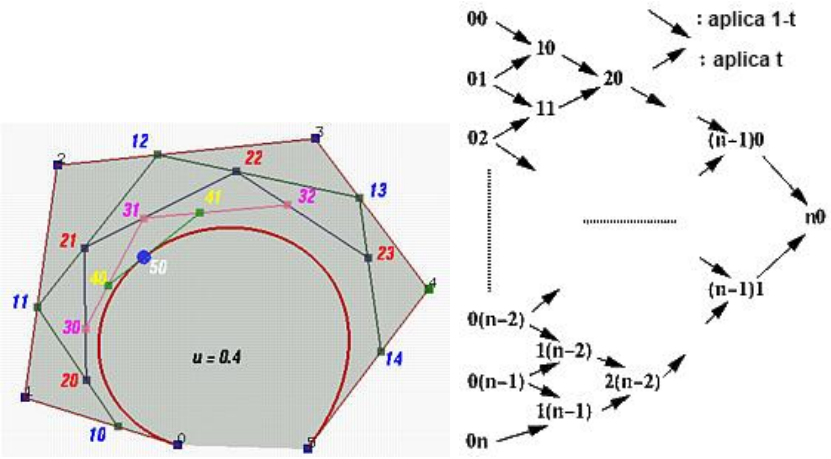


Figura 9. Diagrama De Casteljau

En la Figura 9 podemos ver un diagrama que sintetiza los pasos del algoritmo. Se resume la notación de puntos $\bar{P}_i^j$, poniendo sólo los índices de la siguiente forma: ji. Es decir que el número 20 representa el punto número 0(fila 0) correspondiente a la iteración número 2(columna 2).

Las flechas indican que al punto origen de la flecha se le asigna la operación señalada en la leyenda del diagrama (flecha hacia abajo $1-t$ y flecha hacia arriba t) y se suma con el resultado de la otra flecha que tiene el mismo destino para formar el nuevo punto.

Entonces, las flechas: $00 -> 10$ y $01 -> 10$ significan $\bar{P}_0^1 = (1-t)\bar{P}_0^0 + t\bar{P}_1^0$, generalizando tendremos $\bar{P}_i^{j+1} = (1-t)\bar{P}_i^j + t\bar{P}_{i+1}^j$. El proceso de cálculo será columna a columna. Por ejemplo, primero se calculan todos los puntos de la primera columna (que serán los puntos de control de la curva); luego se calculan los de la segunda columna; después, los de la tercera columna, llegando al punto final, que es el único punto de la última columna y el correspondiente a la curva.

6. Pero ¿cómo se implementa en código?

La implementación es simplemente resolver el diagrama que se presentó arriba. Supongamos que tenemos nuestro arreglo de puntos de control P con n+1 puntos, donde cada elemento del arreglo es un vector en 2D. Además, contamos con un arreglo Q con n+1 puntos, que usaremos para realizar los cálculos. Entonces, una función que, dado un valor para el parámetro t de la curva nos da el punto en la curva, puede implementarse como:

```
Vector2D EvaluarDeCasteljau(float t){
         for(int i=0; i <= n ; ++i)
                  Q[i] = P[i]; // primera columna del diagrama
         for(int k=1; k <= n ; ++k)//por cada columna siguiente del diagrama
                  for(int i=0; i <= (N-k); ++i)
                    Q[i] = (1 - t)*Q[i] + t*Q[i+1];//dividir en ratio t:(1-t)
         return Q[0];
}
```

7. Representación matemática de las curvas de Bézier

Como vimos anteriormente, las curvas de Bézier se representan de forma paramétrica. Dijimos también que dicha curva se compone de puntos de control y posee, además, funciones bases, sobre las cuales volveremos más adelante.

Así las cosas, lo conveniente sería expresar, en una función paramétrica de la forma de $C(t)$, la curva C en función del parámetro t, los puntos de control que la componen y las funciones bases. Esta forma nos permitiría evaluar puntos de la curva para cualquier parámetro t, evaluando sólo una función, en lugar de realizar el algoritmo de De Casteljau. Si bien esto parece lo ideal, más adelante veremos que promete más de lo que realmente puede ofrecer.

Desarrollemos, a modo de ejemplo, cómo sería la representación paramétrica de la forma que hicimos para introducir el algoritmo de De Casteljau. En la siguiente figura podemos ver que en una curva de Bézier que tiene tres puntos de control, $\overline{\boldsymbol{A}}$, $\overline{\boldsymbol{B}}$ y $\overline{\boldsymbol{C}}$, se establecieron dos puntos mediante la división en ratio $t:(1-t)$ $\overline{\boldsymbol{Q_1}}$ y $\overline{\boldsymbol{Q_2}}$.

Allí también se pueden observar las expresiones paramétricas de esos dos puntos. Luego, siguiendo el proceso para calcular un punto en la curva, calculamos otro punto $\overline{\boldsymbol{Q_3}}$ en el segmento $\overline{\boldsymbol{Q_1 Q_2}}$. Como vemos, el punto $\overline{\boldsymbol{Q_3}}$ depende de los puntos $\overline{\boldsymbol{Q_1}}$ y $\overline{\boldsymbol{Q_2}}$, que a su vez dependen de los puntos de control $\overline{\mathbf{A}}$, $\overline{\mathbf{B}}$ y ., Esto significa que podemos escribir la expresión paramétrica de $\overline{\boldsymbol{Q_3}}$ en función del parámetro t y de los puntos de control.

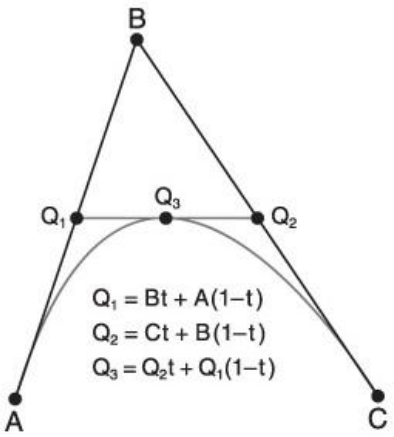


Figura 10. Expresión paramétrica de $\overline{\boldsymbol{Q_3}}$

$$\overline{\mathbf{Q_1}} = (\mathbf{1}-\mathbf{t})\overline{\mathbf{A}} + \mathbf{t}\overline{\mathbf{B}}$$

$$\overline{\mathbf{Q_2}} = (\mathbf{1}-\mathbf{t})\overline{\mathbf{B}} + \mathbf{t}\overline{\mathbf{C}}$$

$$\overline{\mathbf{Q_3}} = (\mathbf{1}-\mathbf{t})\overline{\mathbf{Q_1}} + \mathbf{t}\overline{\mathbf{Q_2}}$$

$$\overline{\mathbf{Q_3}} = (\mathbf{1}-\mathbf{t})^2\overline{\mathbf{A}} + \mathbf{2t}(\mathbf{1}-\mathbf{t})\overline{\mathbf{B}} + \mathbf{t}^2\overline{\mathbf{C}}$$

Esta última es la expresión paramétrica de una curva de Bézier cuadrática. La misma se puede generalizar para curvas con cualquier cantidad de puntos de control y haremos esto expresando la curva como una sumatoria de un conjunto de funciones base, multiplicadas por su correspondiente punto de control.

Esto significa que podemos expresar cualquier curva de Bézier en la forma:

$$\boldsymbol{C(t) = \sum_{i=0}^{n} B_i^n(t)\overline{P_i}}$$

- Donde $\mathbf{B_1^n(t)}$ es una función base, escalar, que veremos a continuación.
- $\overline{\mathrm{P}_1}$ es el punto de control número i.

Las funciones bases de la fórmula son conocidas como *polinomios de Bernstein,* donde a cada punto de control le corresponde un polinomio o función base.

Estos polinomios determinan cuánta influencia tiene cada punto de control para un valor del parámetro t. Es decir, si tenemos un conjunto de puntos de control $\{\overline{\mathbf{A}}, \overline{\mathbf{B}}, \overline{\mathbf{C}}, \overline{\mathbf{D}}\}$, como es de esperarse cerca del borde de comienzo de la curva, el punto de control que tendrá más influencia será el primer punto de control, esto es, el punto $\overline{\mathbf{A}}$**,**. En tanto, los puntos de control adyacentes y cercanos en la lista a $\overline{\mathbf{A}}$, es decir, $\overline{\boldsymbol{B}}$ y $\overline{\boldsymbol{D}}$**,** tendrán una influencia considerable, contrariamente a $\overline{\boldsymbol{D}}$*,* que será el de menos influencia.

La operación $n!$ se conoce como factorial de n, y se resuelve como el producto de todos los naturales desde 1 hasta n. Por ejemplo:

$2! = 1x2 = 2$

$3! = 1x2x3 = 6$

Un resultado particular es: $0! = 1$

Los polinomios de Bernstein se definen como:

$$\boldsymbol{B_i^n(t) = \frac{n!\, t^i (1-t)^{n-i}}{i!\,(n-i)!}}$$

Donde:

- *n* es la cantidad de puntos de control menos 1.
- *i* es el índice del punto de control al que corresponde el polinomio.

Podemos verificar que el resultado de la ecuación general de la curva es igual al resultado de la curva cuadrática de Bézier. Recordemos que al tener tres puntos de control, el grado $n = 2$ y el índice de los puntos comienza en 0. Usando las expresiones generales:

Recordemos que:

$a^0 = 1$

Para todo número natural a

Además consideramos en este contexto:

$0^0 = 1$

$$\mathbf{B_0^2(t) = \frac{2!\, t^0 (1-t)^2}{0!\, 2!} = (1-t)^2}$$

$$\mathbf{B_1^2(t) = \frac{2!\, t^1 (1-t)^1}{1!\, 1!} = 2t(1-t)}$$

$$\mathbf{B_2^2(t) = \frac{2!\, t^2 (1-t)^0}{2!\, 0!} = t^2}$$

$$\boldsymbol{C(t) = \sum_{i=0}^{n} B_i(t)\overline{P_i}}$$

$$\mathbf{C(t) = (1-t)^2\overline{A} + 2t(1-t)\overline{B} + t^2\overline{C}}$$

Como vemos, el resultado es el mismo al que llegamos en el ejemplo.

En la Figura 11 podemos observar un gráfico de los polinomios de Bernstein para una curva de Bézier de cuatro puntos de control, donde es posible notar que, en el comienzo de la curva, la influencia más importante es la de los puntos de control que están primeros en la lista; en la parte central de la curva, los más influyentes son los puntos centrales, y al final, los puntos más influyentes son los últimos.

Cabe destacar que el valor máximo de los polinomios es siempre 1, y sólo se da en el polinomio del punto de control inicial y final. Por eso, la curva pasa por encima, o los interpola, sólo a ellos Además, la suma de todos los polinomios de control en una curva da como resultado 1 para cualquier valor (en el rango $[0,1]$) del parámetro t,. Esto no es una coincidencia, sino una característica importante que tiene muchas implicaciones en las cuales no nos explayaremos, aunque sí señalaremos su implicancia más directa:

Una ventaja que poseen las curvas de Bézier radica en que, para transformarlas, sólo hay que transformar sus puntos de control y construirla, en lugar de transformar una miríada de puntos de la curva.

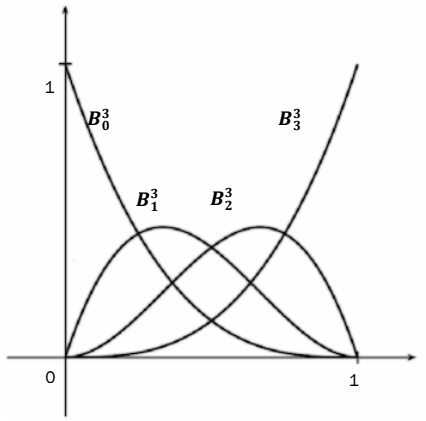


Figura 11. Polinomios de Bernstein $\boldsymbol{B_i^3(t)}$ *para* $\boldsymbol{t}$ *entre* $[\mathbf{0},\mathbf{1}]$

8. Entonces ¿cómo evaluamos las curvas de Bézier?

Vimos dos formas en las cuales puede ser evaluada una curva de Bézier:

- Usando el algoritmo de De Casteljau.
- Usando la representación paramétrica de la curva.

En general, usando la representación paramétrica de la curva, podemos evaluarla más rápidamente, esto es, la evaluación es menos costosa, computacionalmente hablando. El problema reside en que la evaluación directa no es numéricamente estable, lo cual significa básicamente que el resultado de la evaluación puede estar *mal* (en general para un número considerable de puntos de control). Por *mal* debemos entender que el resultado no sólo será impreciso, sino que ni siquiera se aproximará al resultado correcto. En cambio, el algoritmo De Casteljau, si bien puede llegar a ser un poco más costoso, es numéricamente estable, es decir, no presentará los problemas que implica el cálculo directo de la representación paramétrica, la cual –en general– se usa para analizar propiedades de la curva.

9. Limitaciones de las curvas de Bézier

La ventaja que tienen estas curvas reside en que su representación es sencilla y su evaluación es simple de implementar. Pero tiene limitaciones importantes:

- *Control global:* cada punto de control de la curva afecta todos los puntos de la curva (a excepción de los puntos de control de inicio y final). Esto significa que al cambiar sólo un punto de control, la curva cambiará en su totalidad, lo cual es

indeseable en CAGD, donde siempre se prefiere el control local, esto es, que la curva sólo varíe en las cercanías del punto de control movido.

- *Muchos puntos de control -> computo lento:* el hecho de que todo punto de la curva esté determinado por todos los puntos de control provoca que el cómputo sea lento.

La primera limitación es la más importante, ya que torna más difícil describir las curvas para un juego. Existen otros tipos de curvas, como NURBS, que solucionan esta limitación. En cuanto a la segunda limitación, se *soluciona* usando curvas de órdenes bajos, es decir, con 3, 4 o 5 puntos de control.

10. Splines por unión de curvas de Bézier

Spline
Unión *suave* de curvas independientes.

Las curvas de Bézier son adecuadas para representar tramos aislados de cualquier curva libre. Para representar la curva entera, podemos hacer una unión *suave*, esto es, agradable a la vista, de varias curvas de Bézier. La unión suave de curvas independientes se denomina *spline.*

Unir un par de curvas de Bézier de forma *no suave,* tal como se ve en la primera imagen de la Figura 12*,* es muy sencillo, ya que las curvas de Bézier pasan por el primer y último punto de control. El problema radica en que esta unión, que se conoce como C^0, hace que se unan los bordes de la curva, pero sus tendencias de crecimiento y decrecimiento (derivadas) no coinciden y por ello la unión se ve rara y mal a la vista.

Para hacer una unión *suave* que sea agradable a la vista, debemos lograr una continuidad conocida como C^1, en la que en los bordes las curvas coincidan en las tendencias de crecimiento y decrecimiento (derivadas). Esta continuidad también es fácil de conseguir. Lo que debemos hacer es lo que vemos en la segunda imagen de la *Figura :* lograr que los polígonos de control de las dos curvas, que el último y primer segmento de los mismos, sean colineales.

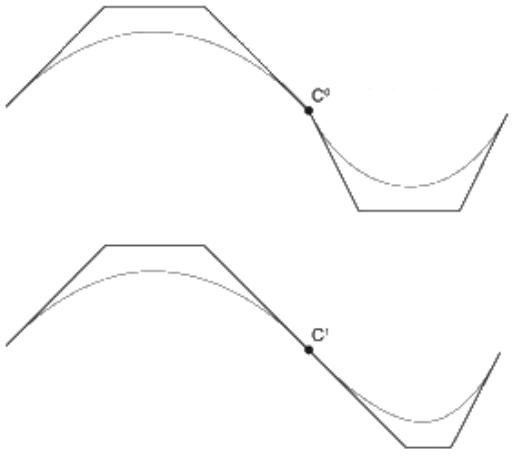


Figura 12. Uniendo curvas de Bézier

BIBLIOGRAFÍA

Dempski K. *Focus on Curves and Surfaces.* Premier Press, 2003.

Hearn, D.; Baker, P. *Computer Graphics, C version.* Second Edition. Pearson Prentice Hall, 1997.

Finding a Point on a Bézier Curve: De Casteljau's Algorithm [en línea] http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/spline/Bezier/de-casteljau.html