Una introducción al diseño de curvas por ordenador

Contenidos

1 El algoritmo de De Casteljau **1**
- 1.1 Parábolas 1
- 1.2 Curvas de Bézier generales 4
- 1.3 La complejidad del algoritmo de De Castelajau 6

2 Propiedades de las curvas de Bézier **7**
- 2.1 La propiedad de la envoltura convexa 7
- 2.2 Interpolación inicial y final 8
- 2.3 Pseudocontrol local. 8
- 2.4 Vectores tangentes. 8

3 Desarrollos más avanzados **9**

4 Para leer más **10**

5 Ejercicios **10**

La representación de curvas más usada en el diseño por ordenador fue descubierta de manera independiente por Pierre Bézier (1910–1999) y por Paul de Casteljau que trabajaron para las empresas automovilísticas de Rénault y Citröen respectivamente. De Casteljau publicó en un informe secreto en 1959 el algoritmo que lleva su nombre destinado a generar por ordenador unas curvas sencillas e intuitivas de manipular, mientras que Bézier en el principio de la década de los 60 derivó una forma diferente de diseñar el mismo tipo de curvas.

Los trabajos de Bézier y de De Casteljau estaban orientados a la industria automovilística, pero ahora las curvas de Bézier (en su versión plana) son la base de todos los programas

informáticos de diseño gráfico (como *Adobe Illustrator* y *Corel Draw*) y del diseño de tipos de fuentes de letras como las *PostScript* o las *TrueType*.

1 El algoritmo de De Casteljau

1.1 Parábolas

Comencemos con el siguiente algoritmo que genera una curva: Sean $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2$ tres puntos en $\mathbb{R}^3$ y $t \in [0,1]$. Construimos los siguientes dos puntos:

$$\mathbf{b}_0^1(t) := (1-t)\mathbf{p}_0 + t\mathbf{p}_1, \qquad \mathbf{b}_1^1(t) := (1-t)\mathbf{p}_1 + t\mathbf{p}_2.$$

Observemos que $\mathbf{b}_0^1(t)$ es un punto situado en el segmento de extremos $\mathbf{p}_0$ y $\mathbf{p}_1$ y análogamente $\mathbf{b}_1^1(t)$ está entre $\mathbf{p}_1$ y $\mathbf{p}_2$. A continuación construimos otro punto:

$$\mathbf{b}_0^2(t) := (1-t)\mathbf{b}_0^1(t) + t\mathbf{b}_1^1(t).$$

Un dibujo será de momento de más ayuda que las fórmulas anteriores: Para ello podemos observar la parte izquierda de la figura 1. A medida que t varía entre 0 y 1, el punto $\mathbf{b}_0^2(t)$ describe una curva, como se puede ver en la parte derecha de la figura 1. La curva $\mathbf{b}_0^2(t)$ se llama *curva de Bézier* asociada a los puntos $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2$.

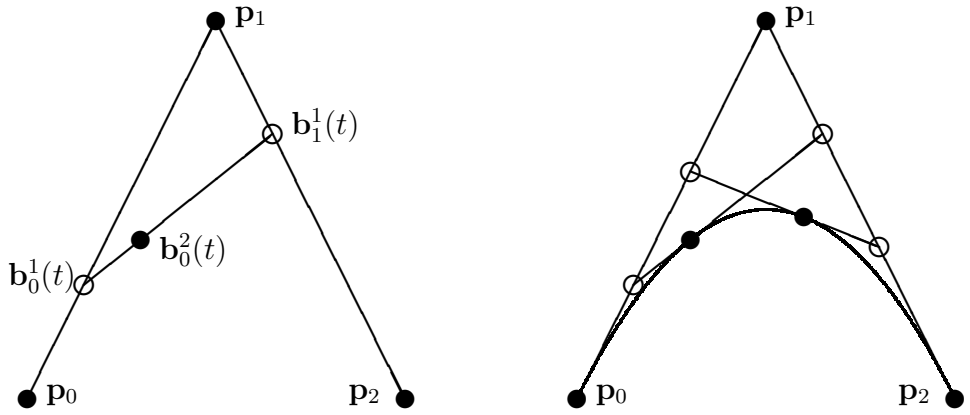

Figura 1: Se muestra a la izquierda el algoritmo de De Casteljau. A la derecha se muestra la curva de Bézier que resulta tras aplicar el algoritmo.

En la siguiente dirección hay un programa (Java) que dibuja curvas de Bézier asociadas a tres puntos de $\mathbb{R}^2$, mediante el algoritmo descrito.

http://personales.upv.es/jbenitez/cajon_sastre/bez_par.jar

¿Qué tipo de curva es la curva de Bézier asociada a los puntos $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2$? El siguiente ejercicio muestra que es una parábola.

Ejercicio: Usando las definiciones de $\mathbf{b}_0^1(t), \mathbf{b}_1^1(t)$ y $\mathbf{b}_0^2(t)$ pruébese que

$$\mathbf{b}_0^2(t) = (1-t)^2\mathbf{p}_0 + 2t(1-t)\mathbf{p}_1 + t^2\mathbf{p}_2. \tag{1}$$

Vamos a implementar una función de MATLAB que dibuje la curva de Bézier asociada a tres puntos de $\mathbb{R}^2$. Si los argumentos de entrada son los puntos

$$\mathbf{p}_0 = (P_{11}, P_{12}), \qquad \mathbf{p}_1 = (P_{21}, P_{22}), \qquad \mathbf{p}_2 = (P_{31}, P_{32}),$$

almacenaremos a estos puntos en la siguiente matriz 3×2

$$P = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \\ P_{31} & P_{32} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix}.$$

Vamos a calcular $\mathbf{b}_0^1(t)$ y $\mathbf{b}_1^1(t)$ de forma simultánea:

$$\begin{aligned} \begin{bmatrix} \mathbf{b}_0^1(t) \\ \mathbf{b}_1^1(t) \end{bmatrix} &= \begin{bmatrix} (1-t)\mathbf{p}_0 + t\mathbf{p}_1 \\ (1-t)\mathbf{p}_1 + t\mathbf{p}_2 \end{bmatrix} \\ &= \underbrace{(1-t)\begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \end{bmatrix} + t\begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix}}_{\star} = \underbrace{\begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \end{bmatrix} + t\left(\begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix} - \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \end{bmatrix}\right)}_{\triangle} \end{aligned}$$

Aunque las expresiones $(\star)$ y $(\triangle)$ sean iguales, se va a usar $(\triangle)$ en vez de $(\star)$ debido a que hay una multiplicación menos.

Por último,

$$\mathbf{b}_0^2(t) = (1-t)\mathbf{b}_0^1(t) + t\mathbf{b}_1^1(t) = \mathbf{b}_0^1(t) + t\left(\mathbf{b}_1^1(t) - \mathbf{b}_0^1(t)\right).$$

```
function cast2(P)
plot(P(1,1),P(1,2),'o')
hold on, axis off, axis equal
plot(P(2,1),P(2,2),'o'), plot(P(3,1),P(3,2),'o')
line(P(:,1),P(:,2))
PI=P(1,:); PF=P(3,:);
for t=0.1:0.1:0.9
    Q=P([1 2],:)+t*(P([2 3],:)-P([1 2],:));
    q1=plot(Q(1,1),Q(1,2),'*');
```

```
    q2=plot(Q(2,1),Q(2,2),'*');
    h=line(Q(:,1),Q(:,2));
    R=Q(1,:)+t*(Q(2,:)-Q(1,:));
    plot(R(1),R(2),'o')
    line([PI(1),R(1)],[PI(2),R(2)])
    pause
    PI=R;
    delete(h), delete(q1), delete(q2)
end
line([R(1),PF(1)],[R(2),PF(2)])
```

Ejercicio: Dibuje varias curvas de Bézier asociadas a distintos puntos $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_2$. Observe que $\overrightarrow{\mathbf{p}_0\mathbf{p}_1}$ es tangente a la curva en $\mathbf{p}_0$ y que $\overrightarrow{\mathbf{p}_1\mathbf{p}_2}$ es tangente a la curva en $\mathbf{p}_1$. Pruebe esta afirmación.

Ejercicio: Imagine que desea enlazar de forma suave los segmentos dibujados en la parte izquierda de la figura 2 con el fin de obtener la parte derecha. Para ello se construirá una parábola de Bézier cuyo punto inicial sea $\mathbf{p}$ y su punto final sea $\mathbf{q}$. Suponga que $\mathbf{p} = (1, 0)$ y $\mathbf{q} = (3, 1)$. ¿Qué coordenadas han de tener los puntos de control? Represente estos puntos en MATLAB y dibuje la curva de Bézier. Ayuda: piense en el ejercicio anterior.

Figura 2: ¿Cómo se ha de proceder para enlazar los dos segmentos de forma suave?

1.2 Curvas de Bézier generales

Las parábolas son curvas planas; sin embargo es interesante en las aplicaciones construir curvas tridimensionales. Además las parábolas permiten poca flexibilidad en el diseño gráfico asistido por ordenador. Esto se logra modificando el algoritmo anterior.

Dados los siguientes $n + 1$ puntos: $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n$ y dado el número $t \in [0, 1]$, en primer lugar se calculan n puntos

$$\mathbf{b}_i^1(t) = (1 - t)\mathbf{p}_i + t\mathbf{p}_{i+1}, \qquad i = 0, \ldots, n - 1.$$

A continuación se calculan $n - 1$ puntos

$$\mathbf{b}_i^2(t) = (1 - t)\mathbf{b}_i^1(t) + t\mathbf{b}_{i+1}^1(t), \qquad i = 0, \ldots, n - 2.$$

Y así progresivamente hasta calcular

$$\mathbf{b}_0^n(t) = (1-t)\mathbf{b}_0^{n-1}(t) + t\mathbf{b}_1^{n-1}(t).$$

Este algoritmo se ve mejor si se pone en forma triangular. Como se ve en la tabla siguiente con cuatro puntos iniciales (en donde se ha escrito $\mathbf{b}_i^r$ por $\mathbf{b}_i^r(t)$).

$$\begin{array}{ccccccc}
\mathbf{p}_0 & & & & & & \\
 & \searrow & & & & & \\
\mathbf{p}_1 & \rightarrow & \mathbf{b}_0^1 = (1-t)\mathbf{p}_0 + t\mathbf{p}_1 & & & & \\
 & \searrow & & \searrow & & & \\
\mathbf{p}_2 & \rightarrow & \mathbf{b}_1^1 = (1-t)\mathbf{p}_1 + t\mathbf{p}_2 & \rightarrow & \mathbf{b}_0^2 = (1-t)\mathbf{b}_0^1 + t\mathbf{b}_1^1 & & \\
 & \searrow & & \searrow & & \searrow & \\
\mathbf{p}_3 & \rightarrow & \mathbf{b}_2^1 = (1-t)\mathbf{p}_2 + t\mathbf{p}_3 & \rightarrow & \mathbf{b}_1^2 = (1-t)\mathbf{b}_1^1 + t\mathbf{b}_2^1 & \rightarrow & \mathbf{b}_0^3 = (1-t)\mathbf{b}_0^2 + t\mathbf{b}_1^2
\end{array}$$

En general, el algoritmo de *de Casteljau* es el siguiente: Dados n puntos $\mathbf{p}_0, \ldots, \mathbf{p}_n$ y $t \in [0,1]$, se calculan los siguientes de forma recursiva:

1: $\mathbf{b}_i^0(t) = \mathbf{p}_i$ para $i = 0, \ldots, n$.

2: $\mathbf{b}_i^r(t) = (1-t)\mathbf{b}_i^{r-1} + t\mathbf{b}_{i+1}^{r-1}$ para $r = 1, \ldots, n$, e $i = 0, \ldots, n-r$.

3: La curva final es $\mathbf{b}_0^n(t)$.

Los puntos $\mathbf{p}_0, \ldots, \mathbf{p}_n$ se llaman *puntos de control* y la curva final se llama *curva de Bézier* asociada a los puntos $\mathbf{p}_0, \ldots, \mathbf{p}_n$, la cual será denotada en lo sucesivo por $\mathcal{B}[\mathbf{p}_0, \ldots, \mathbf{p}_n](t)$.

En la siguiente dirección hay un programa (Java) que dibuja cúbicas de Bézier asociadas a cuatro puntos de $\mathbb{R}^2$, mediante el algoritmo descrito.

http://personales.upv.es/jbenitez/cajon_sastre/bez_cub.jar

En la figura 3 se muestran 4 curvas de Bézier. Las dos de arriba tienen 4 puntos de control, mientras que las dos de abajo tienen 5 puntos de control. En la de abajo de la derecha, los puntos inicial y final coinciden para producir una curva cerrada.

La siguiente función de MATLAB genera una curva de Bézier donde los puntos de control se introducen de forma interactiva con el ratón.

```
function decast(n)
% n es el numero de puntos. Tiene que ser mayor que 2.
axis off, axis equal, hold on
P=ginput(n);
line(P(:,1),P(:,2))
j=1;
x=zeros(1,21); y=x;
```

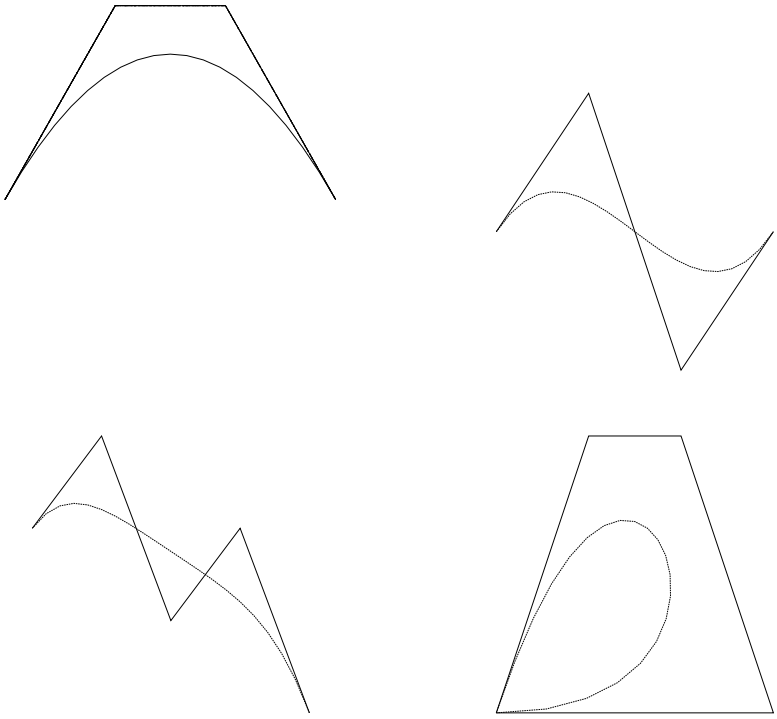

Figura 3: Diferentes curvas de Bézier.

```
for t=0:0.05:1
   Q=P;
   for i=1:n-1
      Q = (1-t)*Q(1:n-i,:)+t*Q(2:n-i+1,:);
      Q(n-i+1:n,:) = zeros(i,2);
   end
   x(j) = Q(1,1); y(j) = Q(1,2);
   j=j+1;
end
plot(x,y)
```

Ejercicio: Modifique el programa anterior de forma que el argumento de entrada en vez de ser el número de puntos que se introducen con el ratón, sea una matriz con n filas y 2 columnas, siendo cada fila las coordenadas de los puntos de control.

Podemos observar en la figura 3 cómo la curva de Bézier suaviza la poligonal. Variando los puntos de control, la curva se modifica por lo que estos puntos efectivamente proporcionan un control de la curva. Se supone que el diseñador de curvas debe saber "pocas" matemáticas.

1.3 La complejidad del algoritmo de De Castelajau

Comentemos a continuación algo sobre la complejidad del algoritmo descrito. En general, un algoritmo depende de n entradas. Por tanto, el número de operaciones depende de n, denotemos por $f(n)$ la cantidad de estas operaciones (en MATLAB se puede llevar la cuenta de las operaciones realizadas mediante el comando `flops`). Es deseable que el número de operaciones sea lo menor posible.

Podemos clasificar los algoritmos dependiendo de cómo sea esta función $f(n)$. En particular hay dos tipos. Los algoritmos tales que $f(n)$ es un polinomio se llaman de *tiempo polinómico* y los que $f(n)$ sea de la forma Ke^{bn}, donde $b > 0$, se llaman de *tiempo exponencial*. Como para n grande, cualquier polinomio es menor que cualquier exponencial, se prefiere los algoritmos de tiempo polinómico a los de tiempo exponencial.

¿Cómo es el algoritmo de De Casteljau? Hemos ejecutado el programa en varias ocasiones y hemos hecho una tabla en donde aparecen el número de puntos de entrada y de operaciones:

n	3	4	5	6	7	8
$f(n)$	1077	1586	2217	2974	3857	4866

El polinomio de segundo grado que mejor ajusta a los datos es $p(n) = 62'6429n^2 + 68'5n + 308'4286$ de modo que cumple

n	3	4	5	6	7	8
$p(n)$	1077	1584	2217	2975	3857	4866

Como se observa fácilmente, se tiene que $f(n)$ y $p(n)$ son muy parecidos para $n = 3, \ldots, 8$. Se puede demostrar efectivamente que en el algoritmo de De Casteljau se cumple que $f(n)$ es un polinomio de grado 2, lo que indica que este algoritmo es bastante rápido.

2 Propiedades de las curvas de Bézier

Vamos a ver algunas propiedades que hacen importantes desde el punto de vista del diseño de curvas por ordenador las curvas de Bézier.

2.1 La propiedad de la envoltura convexa

La curva de Bézier $\mathcal{B}[\mathbf{p}_0, \ldots, \mathbf{p}_t](t)$ siempre está contenida en el polígono cuyos vértices son los puntos $\mathbf{p}_0, \ldots, \mathbf{p}_n$. Esta propiedad se observa en la figura 3.

Esta propiedad es útil por lo siguiente: en muchas ocasiones es deseable saber si dos curvas de Bézier se cortan o no. Esto computacionalmente es costoso (hay que decidir si, dadas las curvas $\mathbf{r}, \mathbf{s} : [0,1] \to \mathbb{R}^2$, existen $t, s \in [0,1]$ tales que $\mathbf{r}(t) = \mathbf{s}(s)$). Si comprobamos que los polígonos no se solapan, que es menos costoso, entonces seguro que las curvas no se cortan. Sin embargo, si los polígonos se solapan, no podemos concluir nada.

2.2 Interpolación inicial y final

La curva de Bézier pasa por el primer y último punto de control.

2.3 Pseudocontrol local.

Supongamos que tenemos una curva de Bézier con puntos de control $\mathbf{p}_0, \ldots, \mathbf{p}_n$ y queremos modificarla levemente. ¿Cómo podemos hacerlo? ¿Qué ocurre si se mueve un punto de control?

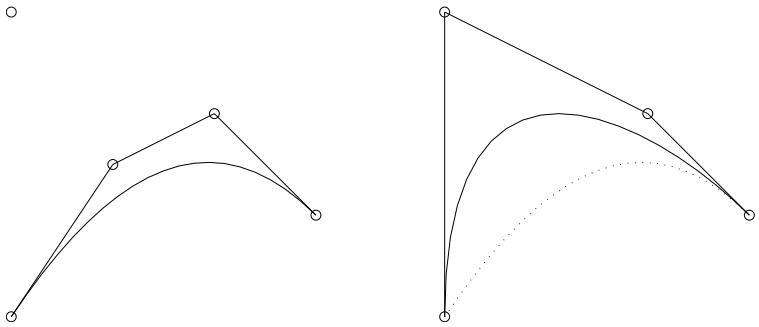

Figura 4: Si movemos un punto de control, la curva de Bézier trata de seguir a este punto.

Si movemos un punto de control, la variación de la curva se hace máxima alrededor del punto de control que movemos, como se indica en la figura 4.

2.4 Vectores tangentes.

En el diseño gráfico es importante saber calcular tangentes a las curvas. Se cumple lo siguiente: Sean $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{n-1}, \mathbf{p}_n$ puntos de $\mathbb{R}^2$ o de $\mathbb{R}^3$ y $\mathbf{r} = \mathbf{r}(t)$ la curva de Bézier asociada a estos puntos.

1. El segmento $\mathbf{p}_0\mathbf{p}_1$ es tangente a $\mathbf{r}$ en el punto inicial (que es $\mathbf{r}(0)$).
2. El segmento $\mathbf{p}_{n-1}\mathbf{p}_n$ es tangente a $\mathbf{r}$ en el punto final (que es $\mathbf{r}(1)$).
3. Los penúltimos puntos calculados por medio del algoritmo de De Casteljau sirven para dibujar la tangente a la curva.

Observe las figuras 1, 3 y 5.

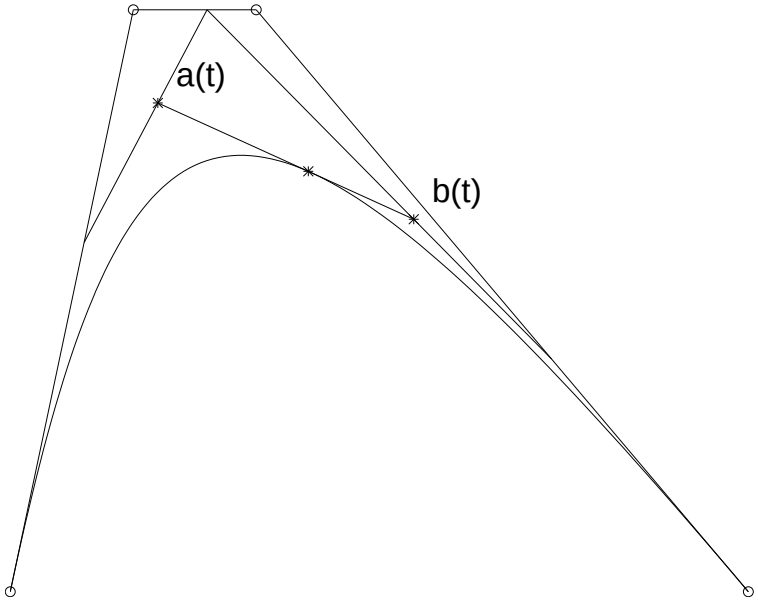

Figura 5: El algoritmo de De Casteljau sirve para trazar tangentes.

Pruebe a ejecutar los programas

http://personales.upv.es/jbenitez/cajon_sastre/bez_par.jar

y

http://personales.upv.es/jbenitez/cajon_sastre/bez_cub.jar

3 Desarrollos más avanzados

Éste no es el momento para profundizar más en la teoría del diseño gráfico por ordenador, ya que el objetivo de la práctica es simplemente mostrar alguna aplicación sencilla del análisis vectorial. Lo único que haremos ahora es señalar algunos defectos de la teoría expuesta y cómo se han resueltos para así indicar algunos esbozos de una teoría más avanzada.

- Si una curva de Bézier tiene un trozo recto, entonces toda la curva debe ser recta. Por tanto, es imposible diseñar una única curva que contenga partes rectas y no rectas. La solución es sencilla: diseñar por separado trozos de curvas que se unen.
- Si se desea generar curvas complicadas, el grado del polinomio debe ser elevado y por tanto los cálculos se ralentizan. La solución es la misma que la del punto previo: diseñar curvas de grado bajo que se ensamblan de forma adecuada. La solución más socorrida es la de usar cúbicas de Bézier.
- Es imposible que la gráfica de un polinomio de grado 2 sea parte de una circunferencia (o de una hipérbola). Por tanto es imposible usar curvas de Bézier para dibujar circunferencias. Hay dos posibles soluciones: una es aproximar un trozo de circunferencia mediante cúbicas de Bézier y la otra solución es usar las llamadas *curvas racionales de Bézier* , que no explicaremos aquí.

4 Para leer más

http://en.wikipedia.org/wiki/Bzier_curve

http://en.wikipedia.org/wiki/De_Casteljau's_algorithm

http://processingjs.nihongoresources.com/bezierinfo/

http://devmag.org.za/2011/04/05/bzier-curves-a-tutorial/

http://tom.cs.byu.edu/~557/text/cagd.pdf

5 Ejercicios

1– En este ejercicio se buscará una cúbica de Bézier para aproximar un cuarto de circunferencia. Por simplicidad se supondrá que la circunferencia es de radio r y está centrada en el origen. Por tanto, el objetivo de este ejercicio es hallar $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$ puntos de $\mathbb{R}^2$ tales que $\mathbf{r}(t) = \mathcal{B}[\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3](t)$ sea la cúbica que se desea hallar. Obsérvese la figura 6.

1. Ya que el cuarto de circunferencia debe pasar por $(r,0)$ y por $(0,r)$, se exige $\mathbf{r}(0) = (r,0)$ y $\mathbf{r}(1) = (0,r)$. Pruébese que $\mathbf{b}_0 = (r,0)$ y $\mathbf{b}_3 = (0,r)$.

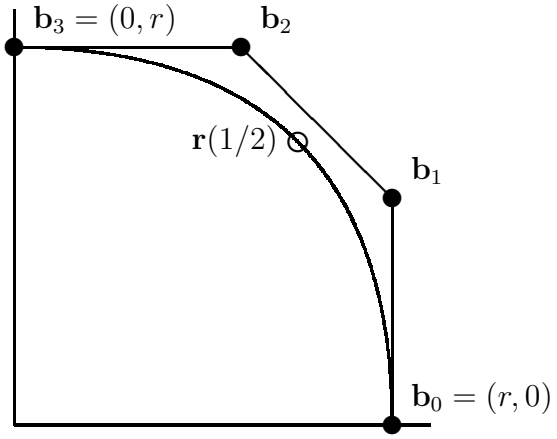

Figura 6: Aproximación de una circunferencia por una cúbica de Bézier.

2. Ya que la tangente al cuarto de la circunferencia en $(r,0)$ es vertical se exige $\mathbf{r}'(0) = (0,\lambda)$ para algún $\lambda > 0$ y por idéntico motivo se exige $\mathbf{r}'(1) = (-\mu, 0)$ para $\mu > 0$. Por cuestión de simetría, se tomará $\lambda = \mu$. Pruébese que $\mathbf{b}_1 = (r, \lambda/3)$ y que $\mathbf{b}_2 = (\lambda/3, r)$.

3. Por tanto, sólo hace falta determinar λ. Forzamos que el punto que está en la mitad de la curva de Bézier pase por la mitad del cuarto de circunferencia. Hágase $\mathbf{r}(1/2) = (r\sqrt{2}/2, r\sqrt{2}/2)$ para hallar λ.

4. Implemente una función de Matlab de modo que sus argumentos de entrada sean el centro y el radio de una circunferencia y dibuje (mediante el procedimiento descrito previamente, enlazando cuatro cuartos de circunferencia y usando los programas dados en esta práctica) esta circunferencia.

2– Observe la figura 5. Pruebe que el vector que une $\mathbf{a}(t)$ y $\mathbf{b}(t)$ es tangente a la curva de Bézier generada por 4 puntos.

3– Pruebe que una expresión explícita para la curva de Bézier generada por los puntos $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ (análoga a la expresión (1)) es

$$\mathbf{r}(t) = (1-t)^3\mathbf{p}_0 + 3t(1-t)^2\mathbf{p}_1 + 2t^2(1-t)\mathbf{p}_2 + t^3\mathbf{p}_3, \qquad t \in [0,1]. \tag{2}$$

4– Sean $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_2$ y $\mathbf{p}_3$ cuatro puntos de $\mathbb{R}^2$ y $\mathbf{v}$ un vector. Defínanse $\mathbf{r}$ la curva de Bézier generada por $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_2$ y $\mathbf{p}_3$ y $\mathbf{s}$ la curva de Bézier generada por $\mathbf{p}_0 + \mathbf{v}$, $\mathbf{p}_1 + \mathbf{v}$, $\mathbf{p}_2 + \mathbf{v}$ y $\mathbf{p}_3 + \mathbf{v}$. Pruebe que

$$\mathbf{r}(t) + \mathbf{v} = \mathbf{s}(t), \qquad t \in [0,1]. \tag{3}$$

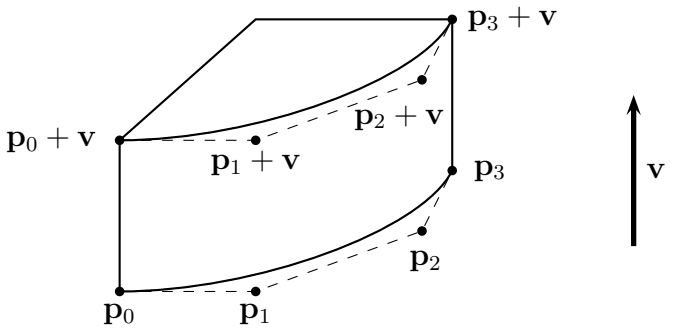

Figura 7: Utilidad de la igualdad (3).

Explique de forma intuitiva qué utilidad puede tener la identidad (3). Para ello piense sobre la figura 7.

5– En este ejercicio se razonará sobre la propiedad del pseudocontrol local.

Sean $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_2$ y $\mathbf{p}_3$ cuatro puntos de $\mathbb{R}^2$ y $\mathbf{v}$ un vector. Defínanse $\mathbf{r}$ la curva de Bézier generada por $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_2$ y $\mathbf{p}_3$ y $\mathbf{s}$ la curva de Bézier generada por $\mathbf{p}_0$, $\mathbf{p}_1 + \mathbf{v}$, $\mathbf{p}_2$ y $\mathbf{p}_3$. Pruebe que

$$\mathbf{r}(t) - \mathbf{s}(t) = 3t(1-t)^2\mathbf{v}, \qquad t \in [0,1]. \tag{4}$$

¿Qué valor de t maximiza $\|\mathbf{r}(t) - \mathbf{s}(t)\|$? ¿Qué significado geométrico posee (4)?

6– Si una curva $\mathbf{r} : [a,b] \to \mathbb{R}^2$ cumple que existe $t_0 \in [a,b]$ tal que $\mathbf{r}'(t_0) = \mathbf{0}$, es posible que tenga un "pico" en $\mathbf{r}(t_0)$, aunque $\mathbf{r}$ sea diferenciable.

1. Dibuje (con Matlab) la curva $\mathbf{r}(t) = (t^2, t^3)$, para $t \in [-1,1]$. ¿Qué ocurre para $t = 0$?
2. Sean $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2$ tres puntos de $\mathbb{R}^2$ no alineados. Pruebe que si $\mathbf{r}$ es la curva de Bézier asociada a estos tres puntos, entonces $\mathbf{r}'(t) \neq \mathbf{0}$ para cualquier $t \in [0,1]$.
3. Dé un ejemplo (dibújelo con Matlab y pruébelo de forma razonada) de 4 puntos de $\mathbb{R}^2$ cuya curva de Bézier asociada presente un "pico". Ayuda: Observe la figura 8.

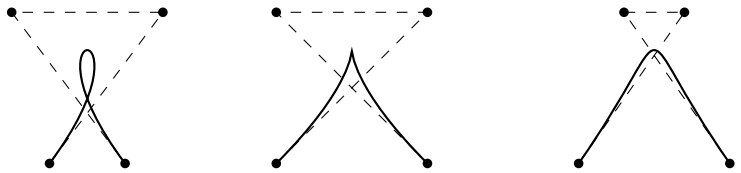

Figura 8: Una cúbica de Bézier puede presentar "picos".

7– Sean $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_2$ y $\mathbf{p}_3$ cuatro puntos de $\mathbb{R}^2$ y M una matriz cuadrada 2×2. Sean $\mathbf{q}_i = M\mathbf{p}_i$ para $i = 0, 1, 2, 3$. Denótese por $\mathbf{r}$ la curva de Bézier asociada a $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_2$ y $\mathbf{p}_3$ y por $\mathbf{s}$ la curva de Bézier asociada a $\mathbf{q}_0$, $\mathbf{q}_1$, $\mathbf{q}_2$ y $\mathbf{q}_3$. Pruebe que

$$M\mathbf{r}(t) = \mathbf{s}(t), \qquad t \in [0, 1]. \tag{5}$$

¿Qué utilidad práctica tiene (5)? Ayuda: Considere la matriz

$$M = \begin{bmatrix} \cos 30^{\text{o}} & -\operatorname{sen} 30^{\text{o}} \\ \operatorname{sen} 30^{\text{o}} & \cos 30^{\text{o}} \end{bmatrix},$$

que corresponde a un giro de ángulo 30^{o} grados y los puntos

$$\mathbf{p}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \mathbf{p}_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \quad \mathbf{p}_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad \mathbf{p}_3 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}.$$

Dibuje (en Matlab y por separado) las curvas de Bézier asociadas a $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_2$ y $\mathbf{p}_3$, y a $\mathbf{q}_0$, $\mathbf{q}_1$, $\mathbf{q}_2$ y $\mathbf{q}_3$. ¿Qué observa?