dizer, associar notas de elementos já existentes. Além, estima-se que em esse processo estão envolvidas atividades como a sensação, percepção, simbolismo e imaginação. Portanto, a metáfora permitiria lograr constructos que se materializam em produtos criativos a maneira de expressões de linguagem, onde a visão é protagonista como meio e sustância desse processo de criação.

**Palavras chave:** Epistemologia - criatividade - conceito - design - cultura visual.

(*) **Mara Edna Serrano Acuña.** Mexicana. Doctora en Educación por UATx (Universidad Autónoma de Tlaxcala - México). Maestra en Ciencias de la Educación por la Benemérita Universidad Autónoma de Puebla - México y la Universidad de Camagüey, Cuba. Profesora del Colegio de Diseño Gráfico de la BUAP (Benemérita Universidad Autónoma de Puebla). Etching Gravure workshop en Crown Point Press; San Francisco, California. USA. Temas de investigación: Representación, Generación y aplicación de conocimiento y Tecnologías de la Educación. Profesional de diseño gráfico.

# Fundamentación gráfica y técnica del dibujo vectorial en el proceso de las curvas Bézier y el lenguaje Postscript

Actas de Diseño (2020, diciembre),
Vol. 32, pp. 232-237. ISSSN 1850-2032.
Fecha de recepción: julio 2016
Fecha de aceptación: septiembre 2017
Versión final: diciembre 2020

Inmaculada Villagrán Arroyal (*)

**Resumen:** Reconocemos las curvas Bézier como el alma y origen del procedimiento ineludible para construir formas gráficas basadas en vectores. Esta solución representativa, arranca de una descripción matemática que en su origen, tenía como única finalidad facilitar un sistema técnico para el diseño automovilístico. Estas fórmulas matemáticas son el fundamento del lenguaje de la programación gráfica en la descripción de objetos con los que hoy dibujamos formas, superficies o tipografías con naturaleza vectorial. Los programas actuales de representación gráfica son los herederos de esta identidad matemática que, gracias a sus herramientas, nos permiten adaptar cualquier forma en cualquier espacio y dirección haciendo de las curvas Bézier su principal aparato locomotor. Estas, junto al lenguaje PostScript, nos permiten hoy en día una perfecta construcción, visualización e impresión con medios digitales.

**Palabras clave:** Digital - vectorial - dibujo - Bézier - gráfico - PostScript - diseño.

[Resúmenes en inglés y portugués y currículum en p. 237]

## Introducción

Gracias al lenguaje de descripción de página *PostScript* se introdujo dentro de sus códigos de programación, el método Bézier, para así poder generar curvas y formas básicas desde la barra de herramientas de cada programa gráfico. De ahí que se hayan convertido en un estándar de calidad dentro del software general para el diseño. Algunos de estos programas llaman a esta herramienta de dibujo simplemente "Bézier", como en *CorelDraw*, o "Curva Bézier" en *InkScape*. La *Suite* de *Adobe*, que contiene programas como *Illustrator*, *InDesign*, *PhotoShop*, *Flash* o *Fireworks*; cada uno de ellos dispone de esta misma herramienta para el dibujo, a la que llaman sencillamente "Pluma". Sin embargo, su base matemática funcional, es la misma para todos. En este artículo, invitamos a los usuarios de los programas basados en vectores, que se familiaricen y conozcan el origen de estas maravillosas curvas que tantas posibilidades ofrecen en nuestro día a día de trabajo digital.

## Origen de las curvas Bézier

Las curvas de Bézier son un sistema que se desarrolló en los años 60 para el trazado de dibujos técnicos. El nombre se lo debemos al ingeniero y matemático francés Pierre Étienne Bézier (1910-1999). Su trayectoria profesional comenzó dentro de la empresa Renault, donde su mayor preocupación era la representación y medición gráfica de las diferentes partes de la carrocería para los automóviles. Para ello, descubrió un método muy seguro, argumentado mediante unas descripciones matemáticas que permitía generar esas curvas. Estas fórmulas se utilizaron para el modelado de las piezas de los vehículos y sus carrocerías. Más adelante fueron aplicadas con gran éxito en un sistema propio de CAD/CAM llamado UNISURF.
El propio Bezier, en su artículo sobre la base matemática del sistema UNISURF, decía que estaba basado en funciones paramétricas polinómicas, con coeficientes vectoriales, cumpliendo los siguientes requisitos:

• Es fácilmente comprensible y utilizable por los diseñadores y otros, cuyo conocimiento profesional se basa en la geometría, en lugar de cálculo.

• Puede generar curvas de espacio y las llamadas superficies retorcidas.
• Esto implica que los polinomios son de grado tres o más.
• Ofrece resultados a corto plazo, solo unas horas para dibujos grandes y unos días para un modelo de coche 3D a gran escala.

La formulación o enunciación de las curvas Bézier están basadas en unos polinomios definidos por el matemático ucraniano Serguéi Bernstein (1880-1968). Aunque, alternativamente, también se basan en un sistema de algoritmos formulados por el físico y matemático francés Paul de Casteljau (1930), que al igual que su coetáneo Pierre Bézier, trabajaba para el sector automovilístico, pero en este caso, para la empresa Citroën. El método matemático de Casteljau fue pionero en ofrecer fórmulas para la construcción de curvas mediante datos estables, pero Bézier se adelantó, publicando en 1966 esos mismos enfoques matemáticos, que permitían ajustar con precisión los diseños de las curvas; de ahí que finalmente llevaran su nombre.
La visión del algoritmo de Casteljau, consistía en cambiar directamente los puntos de los polígonos de control, para que la curva o superficie le continuara de una forma intuitiva y suave, ofreciendo un resultado mucho más preciso que si se modificaba directamente cualquier parte de la curva. Un polígono de control se conoce como la forma constituida por los segmentos que unen los puntos de control, que permiten dar una trayectoria a la curva. En la representación gráfica de las curvas Bézier estos puntos de control son invisibles, pero su fórmula matemática realmente los necesita y los ubica dentro de unas coordenadas determinadas, aunque no se muestren gráficamente.
Para conocer el funcionamiento matemático de las curvas Bézier presentaremos, de una manera descriptiva, los conceptos generales que las definen, evitando todo tipo de tecnicismos y otras complejas fórmulas polinómicas en las que se basan.
La curva Bézier más simple que podemos construir, mediante puntos de control, es la curva lineal. Esta curva es un segmento entre dos de estos puntos ($P_0$) y ($P_1$). Si establecemos una distancia entre ellos, que llamaremos ($t$), daría origen a un nuevo punto ($B$), que podemos registrar como el primer nodo del que va a nacer la curva. Sin embargo, con un solo nodo, aunque dispongamos de dos puntos de control, no podemos aún crear ninguna forma, ya que los puntos de control deben configurar una forma poligonal o polígono de control. Por tanto, la curva se comenzará a visualizar solamente si esta forma está compuesta por tres o más puntos de control. Mientras nuestro vector quede únicamente definido por dos puntos, no hay opción de visualizar la forma o dar flexibilidad a la curva, ya que el punto ($B$) se mantiene en el mismo lugar definido por la distancia de ($t$).
En el momento que se construye un polígono de control, con tres puntos, surge la curva llamada cuadrática, donde el punto ($B$), o nodo, sería el punto tangente al segmento que resulta de las dos curvas lineales principales ($P_0$-$P_1$) y ($P_1$-$P_2$). A medida que cambia la distancia ($t$), dentro de los segmentos que componen esos puntos de control, la curva va adquiriendo su dirección y forma.
Por otro lado, están las curvas llamadas cúbicas, compuestas por cuatro puntos de control, siendo su sistema de construcción similar al anterior, pero donde la longitud ($t$) crea dos curvas cuadráticas y estas, a su vez, una nueva curva lineal que vuelve a definir el punto ($B$) tangente a la curva final.
A partir de las curvas cúbicas, podemos construir nuevas curvas, creando más puntos de control de grados superiores. Estos se basarán en el mismo sistema de construcción hasta conseguir formas más complejas.
Las curvas Bézier, por tanto, son el resultado del movimiento y la dirección de unos vectores que recorren distancias entre segmentos y puntos de control que se van interpolando en el recorrido del trazado. En el momento de utilizar esta herramienta de dibujo, los puntos que ubicamos son los conocidos nodos o puntos de ancla. No son puntos de control, ya que estos últimos solamente conforman el polígono invisible que construye la forma resultante.
Además de las curvas Bézier, también podemos encontrar en los programas de gráficos basados en vectores otras curvas conocidas como *Spline,* un conjunto de curvas formadas con secciones polinómicas que satisfacen la continuidad de la curva, las cuales, sí se manipulan mediante los propios puntos de control. Una *Spline* se puede generar mediante interpolación, es decir, la propia curva contiene en su recorrido todos los puntos de control. Otro sistema para crearlas es por aproximación, donde los puntos de la *Spline* quedan próximos y fuera de la trayectoria.
A este tipo de curvas no las atenderemos en esta investigación, ya que son una generalización de las Bézier, y su diferencia principal respecto a estas son las descripciones matemáticas basadas en otros algoritmos, que gráficamente no son del todo apreciables. Además, las *Spline* son más frecuentes entre las herramientas de los programas de diseño asistido y 3D, como también lo son las curvas *B-Splines* y NURBS (*Non Uniform Rational B-Splines*), utilizadas principalmente para la construcción y representación de planos y superficies tridimensionales. El funcionamiento de estas no se asemeja demasiado al de las Bézier ya que trabajan principalmente las superficies y su respuesta gráfica busca principalmente la suavidad de esas áreas. La trayectoria principal de las NURBS no pasa por los nodos, sino que son de aproximación, no tienen manejadores de tangencia y la superficie se modela mediante los llamados vértices de control (CVs). La variación de la curva afecta más allá de los nodos adyacentes al que movemos, por lo que es más complicada la precisión para el dibujo bidimensional, pero, sin embargo, son perfectas para la continuidad de las curvaturas tridimensionales.

## Dibujar con Bézier

En los programas de dibujo bidimensional, cuando utilizamos la herramienta Bézier –o pluma–, su funcionamiento no resulta tan complejo como aparenta la naturaleza que la engloba. Gracias a esas personas entregadas al cálculo matemático y a la programación, podemos disfrutar de una aplicación de dibujo gratamente intuitiva. Aun así, cuando nos iniciamos en esta herramienta, es necesario seguir unas pautas para hacer buen uso de ella.

Como ya sabemos, el funcionamiento de la pluma Bézier se basa principalmente en la colocación y distribución de nodos para crear un recorrido lineal abierto o cerrado. De cada uno de estos nodos dependerá la forma que adquieran los segmentos contiguos a ellos. Cuando los segmentos son rectos, es evidente que el dibujante solo debe ubicarlos en el lugar deseado, creando una polilínea continua para construir la forma deseada. Por el contrario, cuando buscamos segmentos curvos, puede parecer algo más complicado determinar dónde deben quedar esos nodos para que definan un recorrido suave sin imperfecciones, o saber cómo de largos deben ser los manejadores de tangencia, para que no aparezcan indeseables deformaciones en la forma final. En este apartado proponemos algunos consejos para una buena práctica con la pluma Bézier, que permita crear estas curvas de una forma continua y limpia, dándole un sentido lógico a cada movimiento, configuración y distancia entre nodos.

La correcta construcción de formas vectoriales mediante Bézier, confirma la famosa expresión de Mies van der Rohe, "menos es más". Cuantos menos puntos ubiquemos en una trayectoria, mejor resultado obtendremos de ella. Al iniciarnos en el manejo de esta herramienta, se suele tender a crear una infinidad de nodos en todo el recorrido de la trayectoria y esto solo puede asegurar una línea llena de irregularidades.

En los programas de diseño asistido (CAD), las curvaturas de las superficies mediante NURBS, disponen de herramientas que analizan las áreas que han podido sufrir errores de curvatura y permiten corregir la continuidad de la geometría de la pieza. Sin embargo, en los programas de dibujo e ilustración, como *Illustrator* o *CorelDraw*, la correcta continuidad de la curva solo dependerá de nuestro buen criterio.

Por ejemplo, si nos viéramos en la necesidad de trazar (manualmente con la pluma Bézier), un símbolo que originariamente estuviera en mapa de bits, lo primero que debemos atender es a su forma y comenzar la trayectoria por algunos de sus puntos de vértices. El nodo de arranque suele colocarse como un punto sin arrastre, para que el siguiente tenga libertad suficiente para controlar la curva que deja atrás. Es aconsejable, que cuando las formas sean circulares, se mantenga el arrastre de los puntos de tangente dentro de una ortogonal horizontal o vertical, así la curva quedará más equilibrada. El siguiente punto o nodo debe mantener una distancia apropiada respecto al anterior y colocarlo aproximadamente en el lugar donde cambia la curva de dirección. Cuando la trayectoria es más sinuosa, lo más adecuado es que el arrastre siga el perfil que se desea trazar. En el momento de encontrar un ángulo que obligue a cambiar la dirección del recorrido es mejor marcar el nodo sin ese arrastre, para que el próximo punto modele la siguiente curva.

La distancia entre nodos debe quedar holgada, sobre todo si la curva es amplia y tiene una forma continua sin cambios de dirección. En ese caso, el nodo debe colocarse en el punto medio de la curva, creando un arrastre suficientemente largo para que esta se adapte a la forma deseada. Si los trayectos son cortos debemos cuidar la distancia del arrastre del manejador, ya que en los siguientes puntos puede haber un cruce indeseado entre ellos. No se aconseja ubicar un nuevo nodo cerca del lugar donde se soltó el arrastre anterior, ya que la curva se vería aprisionada entre esos puntos.

Una vez finalizado el recorrido, siempre tenemos opción a modificar cualquiera de estos nodos, e incluso a eliminar algunos sobrantes, ya que los puntos de vértice, aunque no hayan sido arrastrados y si el segmento que le acompaña es curvo, también dispondrán de manejadores, en estos casos asimétricos.

Para editar las curvas Bézier, se utilizan unas herramientas específicas en cada software de dibujo. Los programas de la *Suite* de *Adobe*, disponen de la herramienta "Selección directa", que selecciona y permite mover los puntos de ancla o nodos. Dentro del grupo del icono de la herramienta "Pluma", se encuentran otras opciones para añadir, restar o convertir –de curvos a rectos o viceversa– los puntos de ancla. En *CorelDraw*, los nodos se editan con una única herramienta llamada "Forma". Esta dispone de un menú de propiedades muy completo, donde se pueden modificar todas las características de los nodos y además, permite con un sencillo doble *click* sobre la curva, añadir nodos al instante. Si se pulsa directamente sobre ellos, los elimina.

A menudo, nos podemos encontrar con errores bastante frecuentes que se cometen al trazar con las herramientas Bézier. Por ejemplo, cuando se intenta unir dos segmentos rectos, mediante una curva, debemos disponer de los manejadores de tangencia en esos nodos, para que la curva sea continua y no surjan relieves indeseados.

Los trazados vectoriales, a simple vista, pueden aparentar estar correctos, pero si hacemos uso de la herramienta de aumento o *zoom*, podemos encontrar muchas irregularidades en algunas de las curvas, errores que debemos corregir para ofrecer un trabajo de buena calidad técnica.

Otro caso suele ser el de los manejadores desalineados a mitad de algún punto de la curva. Si la curva es continua, los manejadores de tangente deben estar alineados o simétricos.

También puede suceder el caso contrario, es decir, que los manejadores de tangente de un nodo estén alineados en un punto de vértice y no cambie correctamente el sentido de la curva, y puede que se haya creado un bucle indeseable en esa unión de segmentos. En esta situación, debemos hacer asimétrico el nodo o desalinear el punto de ancla. Así, cada manejador tendrá libertad de movimiento para modelar su curva adyacente.

Por otro lado, debemos controlar la longitud de los manejadores para que no se hagan extremadamente largos. Cuando sucede esto, pueden cruzarse y provocar nuevos salientes que deslucen la trayectoria. A medida que se va adquiriendo práctica en el manejo de la herramienta Bézier, se irá perfeccionando la decisión de establecer la longitud de los manejadores de tangente. A veces, es mejor añadir nuevos nodos antes que abusar de su extensión.

Otro aspecto general a tener en cuenta es la propiedad que debe tener cada punto de ancla o nodo. Si buscamos trayectorias lineales, no conviene que estos dispongan de manejadores, así no se corre el riesgo de que la trayectoria se curve. Por tanto, en esta situación, siempre debemos convertir los puntos curvos en lineales.

Entre otras herramientas similares a Bézier, en el programa *CorelDraw* por ejemplo, disponemos también de la "Pluma" como en *Adobe*. Su funcionamiento es muy

similar al de las anteriores. Con ella, podemos visualizar en todo momento la curva antes de marcar el siguiente nodo, facilitando las posibles dudas que puedan surgir sobre la dirección de la trayectoria que se quiera elegir.
También podemos utilizar herramientas vectoriales de "Mano alzada" y "Medios artísticos". Estas tienen también una respuesta de trayectorias compuestas por nodos, pero en estos casos, se obliga a mantener un trazo continuo como el que realizan las pinceladas de píxeles y no podemos controlar la ubicación de cada uno de estos nodos hasta su posterior edición.
En cuanto al resto de programas gráficos, cada vez son más los que incorporan la herramienta "Bézier" o "Pluma" dentro de sus opciones habituales. Aunque estos no sean específicos para el dibujo basado en vectores, esta aportación es para ellos una gran ayuda para crear selecciones o máscaras, por ejemplo en *Adobe PhotoShop* o dibujos y trayectorias de movimientos en *Flash* o *After Effects*.

## Acerca del lenguaje Postsrcript

Cualquier documento digital de texto, dibujo, diseño, maquetación, etc., lo realizamos con un programa específico –*Microsoft Word*, *Adobe Illustrator*, *CorelDraw*, *Adobe InDesign*, etc.– que finalmente almacenamos como archivo propio del programa en nuestro ordenador. Cuando estos documentos los enviamos a imprimir, por ejemplo a un dispositivo profesional de gran tamaño, el formato de ese documento debe traducirse a un tipo de archivo que pueda interpretar el RIP (Procesador de Imagen Raster), que es el que se encarga de procesar los datos para convertirlos en puntos de impresión, por ejemplo para impresoras de alto nivel.
Para hacer esa conversión, se utiliza un lenguaje de descripción de páginas, LDP –o PDL en inglés–, que indican al procesador o a la impresora los elementos que contienen las páginas, es decir, los textos, las imágenes *bitmap* y los objetos basados en vectores. Además de indicarles la posición en relación a la página e incluso la descripción de las tramas de semitonos.
Los PDLs funcionan con el mismo principio que los lenguajes de programación. Cuando un documento está listo para la impresión, la PC o estación de trabajo toma las imágenes, la información tipográfica y la distribución del documento, y los utiliza como objetos que forman instrucciones para que la impresora los procese. La impresora luego traduce estos objetos en *rasters*, una serie de líneas escaneadas que forman una imagen del documento (*Raster Image Processing* o RIP), e imprime la salida en la página como una imagen completa con texto y gráficos incluidos. Este proceso hace los documentos impresos más consistentes, resultando en ninguna o muy poca variación cuando se imprime el documento en modelos diferentes de impresoras. Los PDLs están diseñados para ser portables en cualquier formato y escalable para ajustarse a cualquier tamaño de papel.
Desde principios de los 80, se han ido desarrollando diferentes lenguajes de descripción de página para describir el formato de los documentos. Pero al estar estos lenguajes diseñados por las propias empresas, que fabricaban las impresoras, solo originó problemas de incompatibilidad, ya que no se podía asegurar que nuestro archivo utilizara el mismo LDP que el de las máquinas de la empresa donde se enviaba la impresión.
Antes de la llegada de la tecnología de inyección de tinta, las impresoras de impacto solamente podían imprimir texto estándar, justificado y sin variaciones de tamaños o estilos de fuentes tipográficas. Hoy día, las impresoras son capaces de procesar documentos complejos con imágenes incrustadas, gráficos y tablas en múltiples esquemas y en diferentes idiomas, todo en una página. Tal complejidad se debe adherir a algunas convenciones de formatos. Esto es lo que ha desatado el desarrollo de los lenguajes de descripción de páginas (o PDL), un lenguaje especializado de formato de documentos creado para la comunicación con impresoras.
El lenguaje de descripción de páginas ofreció una nueva posición a las técnicas de impresión donde el concepto WYSIWYG –lo que ves es lo que obtienes– se llevó a la realidad abarcando en un mismo proceso la inclusión de todos los elementos de una publicación, es decir, textos, imágenes y gráficos vectoriales.
Uno de los primeros LDP fue conocido como Interpress, desarrollado a principios de los 80 en la Xerox PARC por John Warnock (1940) y Charles Geschke (1939) para las impresoras láser Xerox, pero la empresa se negó a comercializarlo en masa y esto provocó que ambos socios se independizaran creando la compañía *Adobe Systems*, donde continuaron con sus propósitos de diseñar un lenguaje para controlar las impresoras de otros fabricantes. De esta manera nació en 1984 el lenguaje *PostScript*.
Un año más tarde, *Apple* adaptó el *PostScript* a sus impresoras láser y fue uno de los componentes que contribuyó, junto a los ordenadores *Apple Macintosh* y al programa de maquetación *Aldus PageMaker*, en el desarrollo y rotundo éxito de la Autoedición o DTP (*Desktop Publishing*), términos acuñados por Paul Brainerd (1947), fundador de la empresa Aldus Corporation.
Antes de la introducción del *PostScript*, los sistemas de edición se basaban en un proceso estrictamente mecánico. Cuando los proyectos finales combinaban elementos de texto, fotografías y arte lineal, la única opción que tenía un editor era dar salida a cada componente en un sistema propietario diferente y armar todo manualmente en una hoja de diseño (original para negativo) usando un cutter, cinta y pegamento. La hoja de diseño era entonces fotografiada y la imagen resultante era transferida a una placa de impresión. Aunque artesanos expertos podían producir trabajos hermosos con esta técnica, lo pesado e intensivo de este proceso limitaba severamente la creatividad y la productividad.
El *PostScript* cambió en definitiva el sector del diseño, las artes gráficas y la preimpresión, pero sobre todo destacó por su capacidad de definir los textos como cualquier otro objeto gráfico; es decir, mediante vectores reducidos a líneas rectas y curvas.
Antes del *PostScript*, las tipografías digitales se encontraban definidas de forma fija en las impresoras. Eran caracteres que se almacenaban en mapa de bits definidos por píxeles, por lo que se necesitaba un dibujo diferente para cada tamaño. Por tanto, no se podían escalar o rotar sin que sufrieran distorsiones. Pero además, este formato

necesitaba mucho espacio y memoria para almacenar todas las variantes tipográficas.
Sin embargo, el lenguaje de página *PostScript*, nos ofreció el sistema vectorial de caracteres sin limitación escalar. Cada carácter de un mismo tipo siempre mantiene el mismo contorno a cualquier tamaño. El lenguaje de descripción traduce el conjunto de curvas mediante algoritmos y fórmulas matemáticas para que la impresora las interprete reconstruyéndolas a cualquier escala y finalmente imprimiéndolas en modo rasterizado, sin agrandar los píxeles, sino interpretando las formas y reubicando cada punto impreso para formar esas líneas y curvas dentro de unas coordenadas concretas sin deformar sus contornos. El lenguaje *PostScript* trata todos los elementos de la página como gráficos basados en objetos, no como un conjunto de caracteres o imágenes de tamaño fijo.
De la misma manera que el *PostScript* funciona con la tipografía, también lo hace con los gráficos vectoriales. Por ejemplo, una elipse, un rectángulo o cualquier forma trazada con curvas Bézier, son reinterpretadas mediante el lenguaje de páginas para que se impriman a cualquier tamaño: son gráficos independientes de sus dimensiones. Los documentos creados en programas como *QuarkXPress* o *Adobe Indesign* se describen mediante el formato específico de ese programa particular. Para visualizar un documento en pantalla, el código del programa debe traducirse a un lenguaje que el monitor sea capaz de comprender. Cuando imprimimos el documento, el código del programa es traducido a código PostScript con sus correspondientes ajustes. El RIP recibe la información *PostScript*, interpreta qué debe hacer en la página y lleva a cabo todos los cálculos. Una vez calculada la página entera, incluidas las imágenes, los textos, los logotipos, etc., se crea un mapa de bits, es decir, un dibujo formado por píxeles. Cuanto más compleja es la página, más tiempo se necesita para realizar los cálculos y más se tarda en rasterizarla.
Con esto queremos decir que realmente la gráfica de los vectores no es más que un concepto abstracto –pero maleable–, que al final visualizamos. Por un lado, en nuestra pantalla, definida por una resolución compuesta de píxeles; por otro lado, en el momento de imprimir. El sistema de representación vectorial se traduce también a los dispositivos de impresión que materializan la página o la ilustración a base de millones de puntos impresos. En ambos casos, tanto las tipografías como los gráficos basados en objetos han pasado por el proceso de rasterización, siendo esta la única manera de visualizar y reconocer sus formas o contenidos.
Por tanto, en líneas generales, la gráfica del dibujo vectorial en el lenguaje PostScript no es más que un almacenamiento de datos geométricos programados, que reciben instrucciones para ubicar su posición dentro de unas coordenadas, para indicar dónde se unen los puntos que forman las trayectorias, que reciben grosores de trazos, rellenos de color y que pueden construir formas primitivas como círculos, rectángulos o polígonos, pero su representación final siempre será una versión rasterizada de su completa descripción.
John Warnock, que también fue uno de los fundadores de *Adobe Systems Incorporated*, decía en su Proyecto Camelot que, dentro del contexto *PostScript* y *Display PostScript* el problema de "ver e imprimir en cualquier lugar" ha sido implementado y resuelto. La mayoría de las aplicaciones tienen controladores de impresión *PostScript*. Documentos de una amplia variedad de aplicaciones se pueden ver en los sistemas operativos que utilizan *Display PostScript*. Los archivos *PostScript* pueden ser enviados por las redes de comunicación e imprimir de forma remota.
Existen tres tipos o niveles de lenguaje *PostScript*.
El *PostScript* Nivel 1 se constituyó a mediados de los años 80 y tenía la capacidad de trabajar con objetos vectoriales y fuentes de contorno, todo definido por una serie de descripciones matemáticas que formaban su propio lenguaje. Además, gestionaba para los rellenos puntos de semitono de diferentes formas, aplicando hasta un máximo de 256 tonos de grises en una impresión.
El *PostScript* Nivel 2 llegó en 1991, consiguiendo con esta versión que todos los productos admitiesen el modo de color CMYK y la capacidad para realizar la separación de color. También mejoraron las técnicas de tramado, los filtros de compresión y descompresión y un soporte mejorado para las funciones de ciertas impresoras.
El *PostScript* Nivel 3 apareció en 1997, con la diferencia de que alcanzaba más velocidad de procesamiento al imprimir y hacía uso de 12 bits para describir las lineaturas de semitono. Por tanto, admitía hasta 4096 tonos del mismo color.
En conclusión, podríamos decir que el lenguaje *PostScript* fue el principio de la salida al exterior de los diseños de páginas y de los dibujos o ilustraciones realizados con medios digitales, pudiendo contener textos, formas vectoriales e incluso imágenes. Este lenguaje ofrece una respuesta de impresión rápida y fiable, ya que es el intérprete que permite que una impresora entienda lo que le pide un ordenador que imprima.
Actualmente *PostScript* es un lenguaje estandarizado y la mayoría de impresoras de gama media-alta trabajan con él, pero no es el único LDP que existe. En los 80, aparte de *Adobe*, otras compañías como HP (*Hewlett Packard*) también diseñaron un lenguaje de descripción de páginas para impresoras láser, el PCL (*Printer Control Language*), y muchos fabricantes de impresoras lo han adquirido para sus máquinas como una opción más de configuración, y sus *drivers* pueden instalarse conjuntamente con PostScript.
De todas maneras, *PostScript* hoy en día no es más que el punto de partida de la impresión mediante códigos en el lenguaje de descripción de páginas, ya que uno de sus descendientes le ha destronado desde hace unos años atrás. Hablamos del conocido formato de documento portátil PDF (*Portable Document Format*), una especie de *PostScript* optimizado que le ha tomado el relevo y ha revolucionado el mercado de la producción gráfica. En 1991, John Warnock también comentaba en su proyecto:

> Nuestra visión para el proyecto Camelot es proporcionar una colección de utilidades, aplicaciones y software de sistema para que cualquier organización pueda capturar documentos desde cualquier aplicación, enviar versiones electrónicas de aquellos documentos desde cualquier lugar y además poder verlos e imprimirlos en cualquier máquina.

Por entonces, Warnock buscaba una solución para el problema de la comunicación visual entre las diferentes aplicaciones y los diferentes sistemas informáticos y reparó en la necesidad de crear una comunicación universal para que se pudiera ver y compartir la información impresa electrónicamente.
Años más tarde, el PDF se ha convertido en el estándar obligado para todas las imprentas a nivel internacional y, al igual que el *PostScript*, está directamente relacionado con el dibujo basado en vectores, consiguiendo ser hoy en día el formato más utilizado para todo uso de intercambio de imágenes y textos.
Con este artículo, recopilamos una serie de datos relacionados con la historia y el origen de las curvas gráficas más utilizadas en los actuales programas de dibujo digital. Una información de reconocimiento histórico para aquellos que iniciaron esta admirada solución gráfica y que hemos querido presentar como una técnica, tan funcional como creativa, ofreciendo una base metodológica para su correcto uso para el diseño o la ilustración.

**Bibliografía**


Arrieche, L. (2014). *La Interpolación y sus problemas*. Recuperado de http://issuu.com/luisarrieche/docs/la_interpolaci__n_y_sus_problemas.d/1 19-10-2015

Bezier, P.E. (1986). Introduction in *The Mathematical Basis of the UNISURF CAD System.* Butterworth-Heinemann. Disponible en: http://www.sciencedirect.com/science/article/pii/B9780408221757500042 20-09-2015

Clark, A. y Clark, E. (2004). *Diccionario Inglés a Español de Computación e Internet.* Florida, USA: Universal Publishers.

Johansson, K; Lundberg, P y Ryberg, R. (2011). *Manual de producción gráfica. Recetas.* Barcelona: Gustavo Gili.

Kline, M. (1974). *Matemáticas en el mundo moderno.* Madrid: Versión española de Guzmán Ozamiz, M. Blume.

Manovich, L. (2005). *El lenguaje de los nuevos medios de comunicación.* Barcelona: Paidós.

Paluszny, M., Prautzsch, H. y Boehm, W. (2005). Métodos de Bézier y B-splines. Universitätsverlag Karlsruhe. Recuperado de https://www.researchgate.net/publication/36450267_Mtodos_de_Bzier_y_B-splines 27-09-2015

PostScript Language Reference (1999). *Adobe System Edition.* 3rd Ed. Disponible en https://www.adobe.com/products/postscript/pdfs/PLRM.pdf

Red Hat Enterprise Linux 4. (2005). Introducción a la administración de sistemas. Cap. 7, *Impresoras e impresión.* Red Hat, Inc. Disponible en http:// web.mit.edu/rhel-doc/4/RH-DOCS/pdf/rhel-isa-es.pdf

Warnock, J. (1991). *The Camelot Project.* Disponible en http://www.planetpdf.com/planetpdf/pdfs/warnock_camelot.pdf

Wong, W y Wong, B. (2004). *Diseño gráfico digital.* Barcelona: Gustavo Gili



**Abstract:** We recognize the Bézier curves as the soul and origin of the inescapable procedure for constructing vector-based graphic forms. This representative solution starts from a mathematical description that originally had the sole purpose of facilitating a technical system for automotive design. These mathematical formulas are the foundation of the language of graphic programming in the description of objects with which today we draw shapes, surfaces or typographies with a vector nature. The current graphic representation programs are the heirs of this mathematical identity and, thanks to their tools, they allow us to adapt any shape in any space and direction, making the Bézier curves their main locomotor apparatus. These, along with the PostScript language, allow us today a perfect construction, visualization and printing with digital media.



**Keywords:** Digital - vector, drawing - Bézier, graphic - PostScript - design.



**Resumo:** Reconhecemos as curvas Béizer como a alma e origem do procedimento inevitável para construir formas gráficas baseadas em vectores. Esta solução representativa começa de uma descrição matemática que em seu origem tinha como única finalidade facilitar um sistema técnico para o design automobilístico. Estas fórmulas matemáticas são o fundamento da linguagem da programação gráfica na descrição de objetos com os que hoje desenhamos formas, superfícies ou tipografias com natureza vectorial. Os programas atuais de representação gráfica são os herdeiros desta identidade matemática que, graças a usas ferramentas, permitem adaptar qualquer forma em qualquer espaço e direção fazendo das curvas Béizer seu principal aparato locomotor. Estas, junto à linguagem PostScript, permitem hoje uma perfeita construção, visualização e impressão com meios digitais.



**Palavras chave:** Digital - vectorial - desenho - Béizer - gráfico - PostScript - design.



(*) **Inmaculada Villagrán.** Doctora por la Universidad de Málaga en Bellas Artes y licenciada por la Universidad de Barcelona donde comenzó como ilustradora y dibujante de animación, En los 90, descubrió la tecnología digital gráfica que aún no ha abandonado. Desde 1994 fue docente ocupacional en cursos de diseño y multimedia. Ha publicado manuales para la formación en diseño. Desde 2003 trabajó para el Centro Cívico de Málaga como diseñadora. Compagina el diseño gráfico con la investigación en las gráficas digitales y la docencia universitaria en dibujo, diseño y fotografía en Bellas Artes de Málaga.