SVG

LIMA UD4 – Tema 1

IES Plurilingüe Antón Losada Diéguez

Adrián Fernández González

CC BY NC SA

Tabla de contenido

1. Introducción....2
2. Uso de SVG....2
2.1. SVG como imagen en HTML....2
2.2. SVG como código en HTML....2
2.3. SVG con CSS....3
3. Etiquetas SVG....3
3.1. Línea. Line....3
3.2. Multilínea. Polyline....4
3.3. Rectángulo. Rectangle....4
3.4. Polígono. Polygon....5
3.5. Círculo. Circle....6
3.6. Elipse. Ellipse....6
3.7. Texto. Text....7
3.7.1. Grupo de texto. tspan....7
3.8. Ruta. Path....7
3.9. Grupo. G....8
3.10. Predefinido. Defs y Use....8
3.11. Marcadores. Marker....9
3.11.1. Atributos....10
4. Personalización de bordes....10
4.1. Color. Stroke....11
4.2. Opacidad. Stroke-opacity....11
4.3. Grosor. Stroke-width....11
4.4. Final. Stroke-linecap....11
4.5. Unión. Stroke-linejoin....12
4.6. Punteado o continuidad. Stroke-dasharray....12
5. Campo de visión. ViewBox....13
6. Programas utilizados....14

Gráficos Escalables Vectoriales. SVG

1. Introducción

Los Gráficos Escalables Vectoriales (*Scalable Vector Graphics* en inglés) o SVG son un formato de representación gráfica bidimensional que se basa en funciones matemáticas sobre un lenguaje de marcas para la representación de las imágenes. A mayores, también permiten la creación de animaciones.

Están basadas en XML y pueden ser creadas y editadas fácilmente con un editor de textos, un editor enriquecido o un IDE. Aun así, dada la complejidad de su generación a mano mediante código, suelen crearse gráficamente mediante programas de diseño específicos.

La principal característica de los SVG es que, al estar basados en leguajes de marcas y funciones matemáticas, pueden ser modificados en tamaño, forma y color sin perder calidad.

2. Uso de SVG

Los SVG son archivos *.svg* en el que dentro está la codificación de la imagen mediante etiquetas y funciones. Estos se crean en base a *.txt* cambiándoles la extensión.

Una vez creado el gráfico, este puede ser visualizado mediante un programa de imágenes o el propio navegador web.

2.1. SVG como imagen en HTML

Al igual que cualquier otra imagen, los SVG pueden ser añadidos a una web mediante la etiqueta *img*, funcionando entonces como una imagen más.

```
<img src="imagen.svg">
```

Otra forma de añadirlos es mediante la etiqueta *object*, lo que permite referenciar el código del propio SVG como si estuviese creado directamente ahí y así modificarlo o darle estilo mediante CSS.

```
<object type="image/svg+xml" data="imagen.svg">SVG no soportado</object>
```

2.2. SVG como código en HTML

A mayores, los SVG pueden ser embebidos y creados directamente dentro del HTML mediante la etiqueta *svg*.

```
<svg width="100" height="100">
    <circle cx="50" cy="50" r="40" fill="yellow"/>
</svg>
```

En este ejemplo, se crea un círculo amarillo de 40 px de radio cuyo centro está en las coordenadas (50, 50) en un contenedor de 100x100 px.

Al crearlo dentro del HTML, este y su contenido puede referenciarse como cualquier otro elemento y, por tanto, darle formato con CSS y manipularlo dinámicamente con JavaScript.

2.3. SVG con CSS

Si se crea embebido en HTML o importado como objeto, sus elementos pueden ser referenciados mediante CSS y aplicarles estilos como cualquier otro elemento.

Hay que tener en cuenta que las propiedades CSS para los elementos SVG difieren de las de los de HTML, ya que disponen de propiedades y valores especiales. Por ejemplo, para ponerle un color a un elemento SVG se utiliza la propiedad *fill*, no *background-color*, para bordes es *stroke* en vez de *border-color*.

Estas propiedades son las mismas, en la mayoría de los casos, que los atributos de las propias etiquetas SVG.

3. Etiquetas SVG

SVG dispone de una serie de elementos con los que se puede representar cualquier imagen. Estos van de algo tan sencillo como una línea o un rectángulo a elipsis o caminos multiforma.

Todos ellos se basan en coordenadas en el plano de dibujo, siendo la esquina superior izquierda el punto de referencia (0, 0) y aumentando positivamente hacia abajo y la derecha.

A diferencia de HTML, el formato de los elementos suele incluirse en la propia etiqueta mediante el uso de atributos independientes para cada una de las propiedades como el tamaño, el borde o el color. Como se mencionó antes, estas pueden ser modificadas mediante CSS.

El contenedor de todas las etiquetas es la *svg*. En su interior se definen los elementos y el respaldo en caso de que el navegador no soporte SVG u ocurra cualquier tipo de fallo.

```
<svg width="100" height="100">
  <circle cx="50" cy="50" r="40" fill="yellow"/>
  SVG no soportado
</svg>
```

En este caso, el texto “SVG no soportado” aparecerá si no funciona el SVG. También es habitual poner una etiqueta *img* con una imagen en otro formato.

Como puede observarse, es habitual estipular el tamaño en la propia etiqueta.

3.1. Línea. Line

El elemento más sencillo es la línea, que se define mediante la etiqueta *line*.

Los atributos básicos son:

x1 y1: Las coordenada x e y de origen.

x2 y2: Las coordenadas x e y de destino.

```
<svg height="210" width="500">
  <line x1="0" y1="0" x2="200" y2="200" stroke="rgb(255,0,0)" stroke-width="3"/>
  El navegador no soporta SVG.
</svg>
```

En este ejemplo se crea una línea desde el punto (0, 0) hasta el (200, 200) con una línea roja de 3 px de grosor.

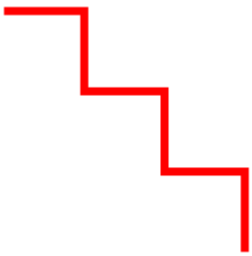
El atributo *stroke* permite definir el color del borde, en este caso la propia línea y *stroke-width* define el grosor de la misma. Ambas propiedades pueden ser añadidas como atributos o como propiedades mediante CSS interno (*style*) o externo (archivo CSS).

3.2. Multilínea. Polyline

La multilínea, como su nombre indica es una línea con múltiples puntos.

Los puntos se definen en el atributo ***points***, que admite indefinidos pares de coordenadas separas por espacios. Las coordenadas se estipulan "x,y".

```
<svg height="180" width="500">
  <polyline points="0,40 40,40 40,80 80,80 80,120 120,120 120,160"
            style="fill:white;stroke:red;stroke-width:4" />
  Tu navegador no soporta SVG.
</svg>
```



En este ejemplo que representa una escalera, se le ha dado formato mediante CSS en el atributo *style* propio de HTML.

3.3. Rectángulo. Rectangle

El elemento *rectangle* permite crear rectángulos con un tamaño determinado.

Sus atributos son:

x e ***y***: Las coordenadas de origen.

width y ***height***: el ancho y alto del rectángulo.

```
<svg width="400" height="180">
  <rect x="50" y="20" width="150" height="150"
  style="fill:blue;stroke:pink;stroke-width:5;
     fill-opacity:0.1;stroke-opacity:0.9" />
  Tu navegador no soporta SVG.
</svg>
```

En este ejemplo, se crea un rectángulo de 150x150 px en la posición (50, 20). A mayores, mediante el atributo *style* se le da color, borde y opacidad a ambos.

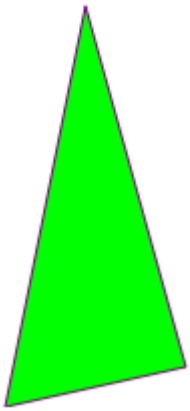
Como puede observarse, mediante CSS no se le podía otorgar opacidad a los bordes de los elementos HTML directamente, pero SVG si lo permite.

3.4. Polígono. Polygon

El elemento *polygon* permite crear polígonos de infinitos lados estableciendo las coordenadas de sus vértices.

Al igual que la *multilínea*, se usa el atributo ***points*** con las coordenadas "x,y" de cada vértice separadas por espacios.

```
<svg height="210" width="500">
  <polygon points="200,10 250,190 160,210"
    style="fill:lime;stroke:purple;stroke-width:1" />
  Tu navegador no soporta SVG.
</svg>
```



Como puede observarse, SVG dibujará líneas uniendo los tres puntos establecidos, uniendo el último con el primero.

3.5. Círculo. Circle

El elemento *circle* permite crear círculos determinando las coordenadas de su centro y su radio.

Sus atributos son:

cx y ***cy***: Las coordenadas x e y del centro.

r: El radio del círculo.

```
<svg height="100" width="100">
  <circle cx="50" cy="50" r="40" fill="tomato" />
  Tu navegador no soporta SVG.
</svg>
```

En este ejemplo se dibuja un círculo de color *tomato* cuyo centro está en el punto (50, 50) y tiene un radio de 40 px.

3.6. Elipse. Ellipse

El elemento *ellipse* se utiliza para representar elipses de una forma muy similar al círculo, con la diferencia de que se estipula el radio máximo vertical y horizontal.

Sus atributos son:

cx y ***cy***: Las coordenadas x e y del centro.

rx y ***ry***: El radio máximo de la elipse en el eje horizontal y vertical respectivamente.

```
<svg height="150" width="500">
  <ellipse cx="150" cy="80" rx="100" ry="50"
    style="fill:wheat;stroke:purple;stroke-width:2" />
  Tu navegador no soporta SVG.
</svg>
```

En este ejemplo se crea una elipse con un radio máximo horizontal de 100px y uno vertical de 50, de ahí que sea más ancha.

3.7. Texto. Text

En SVG también se puede dibujar texto mediante el elemento *text*. Este elemento dispone de los atributos x e y para indicar la posición de inicio del texto y el contenido de la propia etiqueta es el texto a mostrar.

Hay que tener en cuenta que las coordenadas son la línea base de escritura.

```
<svg height="60" width="200">
  <text x="5" y="15" transform="rotate(30 20,40)">Texto SVG</text>
    Tu navegador no soporta SVG.
</svg>
```

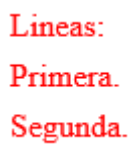
En este ejemplo se puede ver como se le aplicó una rotación mediante el *rotate* dentro del atributo *transform*. Los valores son el número de grados de rotación tomando como punto de giro las coordenadas (20, 40).

3.7.1. Grupo de texto. tspan

El elemento *text* de SVG permite agrupar varios elementos texto mediante el uso de la etiqueta *tspan* en su interior. Además, de la agrupación, todo el estilo aplicado a la etiqueta *text* será aplicado a todas las *tspan* que contenga.

Su uso es igual que el de la propia etiqueta de texto, estipulando las coordenadas de posicionamiento y el texto en su interior.

```
<svg height="90" width="200">
  <text x="10" y="20" style="fill:red;">Lineas:
    <tspan x="10" y="45">Primera.</tspan>
    <tspan x="10" y="70">Segunda.</tspan>
  </text>
    Tu navegador no soporta SVG.
</svg>
```



En este ejemplo se pueden ver tres líneas de texto a las que se le aplicó el mismo color de relleno, estipulado en la etiqueta *text* que las contiene.

3.8. Ruta. Path

La ruta o elemento *path* permite crear gráficos complejos mediante la combinación de diferentes formas.

La principal ventaja de este elemento es que permite representar cualquier cosa y mediante coordenadas relativas al anterior punto, pero su gran desventaja es su gran complejidad.

No es recomendable su uso manual más allá de algunas formas sencillas. Para cosas complejas se recomienda la automatización del proceso desde un lenguaje de alto nivel o el uso de una aplicación de dibujo SVG.

3.9. Grupo. G

Para un mayor control de los elementos, mayor estructuración en el documento o la aplicación de estilos conjuntos, SVG dispone de la etiqueta de grupo *g*.

Por si misma, esta etiqueta no aporta nada a los elementos que contiene y puede ser aplicada cualquier conjunto de elementos sin alterar su funcionamiento.

```
<g stroke="green" fill="white" stroke-width="5">
  <circle cx="25" cy="25" r="15"/>
  <circle cx="40" cy="25" r="15"/>
  <circle cx="55" cy="25" r="15"/>
  <circle cx="70" cy="25" r="15"/>
</g>
```

En este ejemplo se puede ver su utilidad para aplicar un estilo conjunto a varios elementos.

3.10. Predefinido. Defs y Use

Al declarar el SVG, este se dibuja directamente en el lugar en el que se define la etiqueta *svg*. Si lo que se necesita es declararlo antes para utilizarlo a posteriori o utilizarlo varias veces, se utiliza la etiqueta *defs*.

Con *defs* se definen los elementos con un id al que referenciar posteriormente mediante el atributo *href* de la etiqueta *use*.

```
<defs>
  <circle id="myCircle" cx="0" cy="0" r="5" />
</defs>

<use x="5" y="5" href="#myCircle" fill="silver"/>
```

Otra ventaja del uso de *defs*, es que, si una propiedad no es establecida en el elemento, por ejemplo, el color, esta puede ser definida en el momento de su uso, permitiendo reutilizar la misma forma para distintas partes.

La etiqueta *use* dispone de dos atributos, **x** e **y** que permiten establecer el desplazamiento en dichos ejes de la figura original. Por ejemplo, si se establecen a 10, la figura aparecerá 10 px a la derecha y 10px abajo. Es equivalente al *traslation* de CSS.

```svg
<svg viewBox="0 0 100 100" xmlns="http://www.w3.org/2000/svg">
  <!-- Definiciones -->
  <defs>
    <circle id="myCircle" cx="0" cy="0" r="5" />

    <linearGradient id="myGradient">
      <stop offset="20%" stop-color="gold" />
      <stop offset="90%" stop-color="red" />
    </linearGradient>
  </defs>

  <!-- Uso -->
  <use x="5" y="5" href="#myCircle" fill="url('#myGradient')" />
</svg>
```

La etiqueta *use* se puede utilizar dentro de cualquier etiqueta svg tras la definición del elemento a utilizar.

El elemento a referenciar puede ser un elemento unitario o un grupo, pudiendo reutilizar gran cantidad de elementos a la vez.

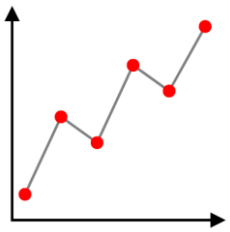
Las definiciones pueden ser realizadas incluso en un documento externo, pudiendo importar y combinar elementos de distintas fuentes.

3.11. Marcadores. Marker

Los marcadores permiten definir un SVG para ser utilizado como extremo o cambio de dirección de una línea. Estos son un elemento predefinido, por lo que se crean en un elemento *marker* dentro de un elemento *defs*.

A diferencia de las otras definiciones que necesitan de una etiqueta *use*, estos se usan directamente en el elemento en el que se aplican mediante los atributos *marker-start*, *marker-end* y *marker-mid* para inicio, fin y cambio de dirección respectivamente.

```
<defs>
  <!-- Punta de flecha -->
  <marker id="arrow" viewBox="0 0 10 10" refX="5" refY="5"
      markerWidth="6" markerHeight="6"
      orient="auto-start-reverse">
    <path d="M 0 0 L 10 5 L 0 10 z" />
  </marker>

  <!-- Punto -->
  <marker id="dot" viewBox="0 0 10 10" refX="5" refY="5"
      markerWidth="5" markerHeight="5">
    <circle cx="5" cy="5" r="5" fill="red" />
  </marker>
</defs>

<!-- Eje de coordenadas -->
<polyline points="10,10 10,90 90,90" fill="none" stroke="black"
 marker-start="url(#arrow)" marker-end="url(#arrow)" />

<!-- Línea de datos -->
<polyline points="15,80 29,50 43,60 57,30 71,40 85,15" fill="none" stroke="grey"
 marker-start="url(#dot)" marker-mid="url(#dot)"  marker-end="url(#dot)" />
```



En este ejemplo se pueden ver todas las posibilidades de uso, definiendo un marcador de punta y un marcador punto para los cambios de dirección.

3.11.1. Atributos

La etiqueta *marker* disponen de varios atributos que permiten estipular su colocación.

refX** y **refY: Permiten estipular el desplazamiento en el eje x e y a la hora de colar, de lo contrario, el centro de la línea será el punto de colocación del punto (0, 0).

markerWidth y ***markerHeight***: Definen el tamaño real del marcador. Este puede ser escalado posteriormente.

Orientation: Permite estipular el comportamiento del marcador con respecto a la posición en la línea. Por defecto, valor 0, se dibuja siempre en la misma orientación, con un número se estipulan los grados de rotación, con *auto* se coloca en el sentido en el que se dibuja la línea y con *auto-start-reverse* va en el sentido de la línea menos en el inicio, que se invierte.

4. Personalización de bordes

Uno de los pilares fundamentales de SVG que no dispone HTML es el alto nivel de personalización de los bordes.

Todas las propiedades pueden ser estipuladas en forma de atributo o como una propiedad CSS. En ambos casos, el nombre y los valores posibles son los mismos.

4.1. Color. Stroke

Con *stroke* se estipula el color de la línea. Admite cualquier formato de color admitido por CSS. Por defecto toma el valor *black*.

```
<path stroke="#aaa" d="M5 20 l215 0"/>
<path stroke="wheat" d="M5 40 l215 0"/>
<path stroke="rgb(255, 255, 0)" d="M5 60 l215 0"/>
```


4.2. Opacidad. Stroke-opacity

Con *stroke-opacity* se puede estipular la transparencia del borde, sin necesidad de recurrir a darle color mediante *rgba* como sucedía en HTML.

```
<path stroke="black" stroke-opacity="0.5" d="M5 40 l215 0"/>
```


4.3. Grosor. Stroke-width

Mediante el uso de *stroke-width* se puede estipular el grosor del borde. Por defecto es 1px.

```
<path stroke="#aaa" d="M5 20 l215 0" stroke-width="1"/>
<path stroke="wheat" d="M5 40 l215 0" stroke-width="2"/>
<path stroke="rgb(255, 255, 0)" d="M5 60 l215 0" stroke-width="3"/>
```


4.4. Final. Stroke-linecap

Con *stroke-linecap* se puede estipular la forma en la que termina el borde.

Esto solo es aplicable para rectas o rutas lineales, ya que en polígonos u otro tipo de elementos cerrados, no hay finales de líneas visibles.

Dispone de tres valores posibles, ***butt***, que termina abruptamente y es el valor por defecto, ***round***, que termina de forma redondeada y ***square***, que termina de forma recta.

```
<path stroke-linecap="butt" d="M5 20 l215 0" />
<path stroke-linecap="round" d="M5 40 l215 0" />
<path stroke-linecap="square" d="M5 60 l215 0" />
```

Como se puede observar en este ejemplo, los valores *round* y *square* aplican el final del borde más allá de la longitud de la línea.

4.5. Unión. Stroke-linejoin

Esto es similar a lo anterior, pero para vértices, puesto que con *stroke-linejoin* se puede establecer la forma que tomará una unión de líneas.

En este caso, puede aplicarse a polígonos o líneas que cambien de dirección en algún momento.

Los tres valores posibles son ***miter***, su valor por defecto y que le da un aspecto puntiagudo, ***round***, que le da forma redondeada y ***bevel***, que le da una forma de corte o escalón.

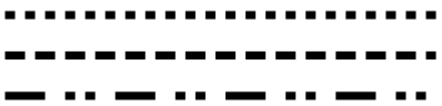
```
<polyline points="40 60 80 20 120 60" stroke="#ccc" stroke-width="20"
    fill="none" stroke-linejoin="miter"/>

<polyline points="40 100 80 60 120 100" stroke="wheat" stroke-width="20"
    fill="none" stroke-linejoin="round"/>

<polyline points="40 140 80 100 120 140" stroke="gold" stroke-width="20"
    fill="none" stroke-linejoin="bevel"/>
```


4.6. Punteado o continuidad. Stroke-dasharray

Otro tipo de estilo que se le puede dar es el punteado mediante *stroke-dasharray*. Esta propiedad permite un alto nivel de personalización, pudiendo representar punteados de todo tipo tan complejos como se quiera.

El valor es un conjunto de números separados por comas que representan un patrón repetitivo formado por el número de píxeles de línea dibujada y espacio.

```
<path stroke-dasharray="5,5" d="M5 20 1215 0" />
<path stroke-dasharray="10,5" d="M5 40 1215 0" />
<path stroke-dasharray="20,10,5,5,5,10" d="M5 60 1215 0" />
```



En este ejemplo se ven tres líneas con tres patrones distintos:

El primero dibuja 5 píxeles y luego deja otros 5 de espacio.

El segundo dibuja 10 píxeles y deja otros 5 en blanco.

El tercero dibuja 20, luego deja 10, dibuja 5, deja otros 5 de espacio, dibuja 5 y deja 10 de espacio.

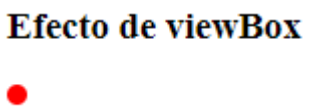
Hay que tener en cuenta que el patrón se repite para cada línea a dibujar o no, por lo que, si se establece un número impar de números, en la segunda vuelta los valores que en la primera representaban líneas, en esta representarán espacios y viceversa.

5. Campo de visión. ViewBox

El atributo *viewBox* establece lo que representa el tamaño total del contenedor respecto al svg representado. Por ejemplo, si se estipula 0 0 10 10, el contenedor irá de las coordenadas (0, 0) a las (10, 10) del SVG que contiene, independientemente del tamaño del contenedor. Esto permite que, si el contenedor se establece de 100x100 px, la imagen se hará mucho más grande.

Además de poder estipularse en la etiqueta svg, puede establecerse en los *marker*.

```
<svg>
  <circle cx="5" cy="5" r="5" fill="red" />
</svg>
```



En este ejemplo, el tamaño del contenedor es de 500x500, por lo que se ve extremadamente pequeño el círculo de 10x10

```
<svg viewBox="0 0 100 100">
  <circle cx="5" cy="5" r="5" fill="red" />
</svg>
```

En este otro ejemplo, se ha estipulado el viewBox a 0 0 100 100, por lo que el svg representa una zona de 100x100, ampliando el tamaño del círculo en 5 veces el original.

6. Programas utilizados

Como se comentó anteriormente, dada la complejidad de su creación mediante código, existen múltiples programas en el mercado diseñados expresamente para su creación.

Adobe Illustrator es uno de los más conocidos y uno de los más utilizados a nivel profesional. Diseñado en exclusiva para crear gráficos vectoriales SVG. Es de pago bajo suscripción mensual, aunque disponible en pack junto con otros de los programas de Adobe.

Corel Draw es otro de los más utilizados a nivel profesional. Permite trabajar con múltiples formatos de imágenes, incluyendo SVG. Es de pago bajo suscripción o pago único sin actualizaciones.

Inkscape es otro ampliamente usado por la comunidad dado que tiene opción para MAC y GNU/Linux, es gratuito y bajo licencia de software libre.