SVG es el acrónimo de *Scalable Vector Grafics* (graficos vectoriales expandibles), un estandard W3C (*World Wide Web Consortium*), basado en XML (e*X*t*ensible* M*arkup* L*anguage*). Utilizamos SVG para crear imágenes vectoriales bidimensionales.

Por defecto

- la esquina izquierda arriba de la ventana grafica se encuentra en un punto de coordenadas {x:0,y:0}
- la anchura de un elemento SVG es de 300px y la altura de 150px.

Líneas

La línea empieza en el punto `x1,y1` y acaba en el punto `x2,y2`. El atributo `stroke` define el color, mientras que el atributo `stroke-width` define el grosor de línea.

Ejemplo:

```
<svg width="250" height="200" viewBox="0 0 250 200">
    <line x1="10" y1="20" x2="220" y2="70" style="stroke:#f00; stroke-width:3"></line>
    <line x1="10" y1="40" x2="220" y2="180" stroke="red" stroke-width="3"></line>
</svg>
```

Atributo	ejemplo	Descripción
x1	`<line x1="10" y1="10" x2="200" y2="10" stroke="black" stroke-width="2" />`	La línea empieza en el punto `x1,y1` y acaba en el punto `x2,y2`
y1		
x2		
y2		
stroke	`style = "stroke:rgb(255,0,0);"` o `stroke="red"`	Define el aspecto de una línea. Su valor puede ser un color, un degradado (*gradient*) o una imagen.
stroke-width	`style = "stroke-width:2;"` o `stroke-width = "2"`	Define el grosor de una línea.
stroke-linecap	`stroke-linecap="butt"`	Define diferentes tipos de terminaciones de un trazado posibles valores: `butt` | `round` | `square`

Atributo	ejemplo	Descripción
stroke-dash array	`stroke-dasharray="5,5"`	Se utiliza para crear líneas discontinuas

Uniones de línea

Podemos dibujar un trazado en SVG conectando lineas y curvas. Al conectar dos líneas, exactamente como en canvas podemos escoger también el tipo de unión. Para esto utilizamos el atributo stroke-linejoin cuyas posibles valores son: bevel (*biselado*), round (*redondeado*) y miter (*en angulo*):

```
<svg width="250" height="400" viewBox="0 0 250 400">
    <polyline stroke-linejoin="round" points="50,150 125,50 200,150" stroke="blue" stroke-width="30" fill="none"></polyline>
    <polyline stroke-linejoin="bevel" points="50,250 125,150 200,250" stroke="blue" stroke-width="30" fill="none"></polyline>
    <polyline stroke-linejoin="miter" points="50,350 125,250 200,350" stroke="blue" stroke-width="30" fill="none"></polyline>
</svg>
```

Formas geométricas

Círculo

En SVG el elemento <circle> es utilizado para dibujar círculos. Los atributos cx y cy definen las coordenadas del centro y el atributo r define el radio del círculo. Si cx y cy no están especificados, el centro del círculo está en el origen del canvas, el punto cuyas coordenadas son: x=0, y=0 (la esquina izquierda arriba del lienzo SVG).

```
<svg width="250" height="150" viewBox="0 0 250 150">
   <circle cx="125" cy="75" r="40" stroke="black" stroke-width="4" fill="#6ab150"></circle>
</svg>
```

Elipse

En SVG para dibujar una elipse utilizamos el elemento <elipse>. Los atributos cx y cy definen las coordenadas del centro mientras que los atributos rx y ry definen los radios horizontal (rx) y vertical (ry) respectivamente.
Si rx = ry el resultado es un círculo. En el siguiente ejemplo rx > ry, lo que hace que la elipse sea más ancha que alta. Huelga decir que si rx == ry el resultado es un círculo

```
<svg width="250" height="150" viewBox="0 0 250 150">
   <ellipse cx="125" cy="75" rx="70" ry="50" fill="#6ab150" stroke="red" stroke-width="5"></ellipse>
```

```
</svg>
```

Rectángulo

Para dibujar un rectángulo podemos utilizar el elemento <rect>. Los atributos x e y definen las coordenadas de la esquina arriba izquierda del rectángulo, mientras que los atributos height y width definen la anchura y la altura del mismo.

```
<svg width="250" height="150" viewBox="0 0 250 150">
   <rect x="70" y="25" height="100" width="120" style="stroke:#000; fill: #6ab150"></rect>
</svg>
```

Rectángulos con esquinas redondeadas

Podemos crear rectángulos con esquinas redondeadas declarando otros dos atributos: rx y ry, donde, exactamente como en el caso de las elipses, rx representa el radio horizontal, y ry el radio vertical.
Si no declaramos ry se considera que ry = rx.

Polígono

En SVG podemos dibujar también polígonos. El elemento <polygon> dibuja líneas entre los puntos definidos por el atributo points

```
<svg width="250" height="150" viewBox="0 0 250 150">
   <polygon points="125,10 175,130 75,130" style="fill:#6ab150;stroke:#003300;stroke-width:3"></polygon>
</svg>
```

Polilínea

Parece que la única diferencia entre <polygon> y <polyline> es que el <polygon> cierra el trazado, mientras que <polyline> no.

```
<svg width="250" height="150" viewBox="0 0 250 150">
   <polyline points="125,10 175,130 75,130" style="fill:#6ab150;stroke:#003300;stroke-width:3"></polyline>
</svg>
```

Trazados

El elemento <path> se utiliza en SVG para dibujar formas complejas, una combinación de líneas, arcos y curvas Bézier. Para entender como funciona el elemento <path> veamos un ejemplo:

```
<svg width="250" height="150" viewBox="0 0 250 150" id="verde">
```

```
   <path d="M50,130   L90,105  A40,45 -30 1,1 150 80  L200,55"
style="stroke:#007700; stroke-width:3; fill:none;"></path>
</svg>
<svg width="250" height="150" viewBox="0 0 250 150" id="rojo">
   <path d="M50,130   l40,-25  a40,45 -30 1,1 60 -25  l50,-25"
style="stroke:#ff0000; stroke-width:3; fill:none;"></path>
</svg>
```

Instrucciones de control (commands) disponibles para <path>:

M(absoluto) **m(relativo)**	`moveto` "Mueve el "lapiz" en este punto (x,y). No dibuja ningúna línea	Mx,y
L(absoluto) **l(relativo)**	`lineto` Dibuja una línea desde el punto actual (x,y) hasta este punto (x1,y1)	Lx1,y1
H(absoluto) **h(relativo)**	`horizontal lineto` Dibuja una línea horizontal desde el punto actual (x,y) hasta un punto (x1,y)	Hx1
V(absoluto) **v(relativo)**	`vertical lineto` Dibuja una línea vertical desde el punto actual (x,y) hasta un punto (x,y1)	Vy1
C(absoluto) **c(relativo)**	`curveto` Dibuja una curva cúbica de Bézier desde el punto actual (x,y) hasta un punto (x3,y3). x1,y1 x2,y2 son las coordenadas de los puntos de anclaje *(control points)* P1 y P2.	Cx1,y1 x2,y2 x3,y3
S(absoluto) **s(relativo)**	`smooth curveto` Dibuja una curva cúbica de Bézier desde el punto actual (x,y) hasta un punto (x2,y2). El primer punto de anclaje es el **reflejo** del segundo punto de anclaje de la curva anterior; x1,y1 son las coordenadas del segundo punto de anclaje *(control point)* P1.	Sx1,y1 x2,y2
Q(absoluto) **q(relativo)**	`quadratic Bézier curve` Dibuja una curva cuadrática de Bézier desde el punto actual	Qx1,y1 x2,y2

	(x,y) hasta un punto (x2,y2); x1,y1 son las coordenadas del punto de anclaje *(control point)* P1.	
T(absoluto) **t(relativo)**	`smooth quadratic Bézier curveto` Dibuja una curva cuadrática de Bézier desde el punto actual (x,y) hasta un punto (x1,y1). El punto de anclaje *(control point)* es el **reflejo** del punto de anclaje de la curva anterior;	Tx1,y1
A(absoluto) **a(relativo)**	`elliptical arc` Dibuja un arco elíptico desde el punto actual hasta un punto (x1,y1); rx y ry son los radios elipticos en la dirección x e y respectivamente. x-axis-rotation indica la rotación de la élipse relativamente al sistema de coordinadas actuales. large-arc-flag y sweep-flag ayudan al calcular como e arco esta dibujado	Arx,ry x-axis-rotation large-arc-flag (0 o 1), sweep-flag (**1 = clockwise; 0 = anti-clockwise**), x1,y1
Z **z**	`closepath` Cierra el trazado	Z

Texto

Para escribir en el SVG utilizamos el elemento <text>. En el siguiente ejemplo la línea roja dibujada con <path> representa la línea base del texto (*alignment baseline*).

```
<svg width="250" height="60" viewBox="0 0 250 60">
   <text x="30" y="40" style="font-family: Consolas">línea base del texto</text>
   <path class="linea_base" d="m30,40 h180" style="stroke:#f00; stroke-width:.5; fill:none;"></path>
</svg>
```

A continuación, damos una lista indicativa y no exhaustiva con los atributos que pueden ser utilizados con <text>.

Atributo	Ejemplo	Descripcion

stroke	`style = "stroke: #000000";`
	o
	`stroke = "black"`
		El color de la línea que dibuja un borde alrededor del texto. Por defecto el texto tiene solo color de relleno. Al ponerle un borde, el texto aparece como negrita.

Atributo	Ejemplo	Descripcion
stroke-width	`style = "stroke-width : 2";` o `stroke-width = "2"`	El grosor de línea que dibuja un borde alrededor del texto.
fill	`style = "fill: #00ff00;"` o `fill="#00ff00"`	El color de relleno del texto.
rotate	`rotate="0 10 20 30..."`	Puede tomar como valor una lista de números indicando la rotación individual en grados sexagesimales de cada carácter del elemento <text> o <tspan>.
font-family	`style = "font-family: Verdana;"`	La familia de fuente a utilizar .
font-size	`style = "font-size:16px;"`	El tamaño de fuente.
kerning	`style = "kerning:2;"`	Espaciado entre letras.
letter-spacing	`style = "letter-spacing:3;"`	Espaciado entre letras; similar a kerning.
word-spacing	`style = "word-spacing:3;"`	Espaciado entre palabras.
text-decoration	`style = "text-decoration: underline;"`	Decoración de texto. Puede tomar uno de estos valores: underline, overline o line-through.
text-anchor	`style = "text-anchor: start"`	start, middle, end
dominant-baseline	`dominant-baseline: middle`	auto | use-script | no-change | reset-size | ideographic | alphabetic | hanging | mathematical | central | middle | text-after-edge | text-before-edge | inherit
baseline-shift	`baseline-shift: super;`	auto | baseline | sup | sub | *porcentaje* | *extensión* | inherit
text Length	`textLength = "150"`	*número*

Atributo	Ejemplo	Descripcion
length Adjust	`lengthAdjust = "spacing"`	spacing | spacingAndGlyphs
writing-mode	`style = "writing-mode: tb;"`	tb (Top to Bottom = de arriba a abajo); rl (right to left = de derecha a izquierda); lr (left to right = de izquierda a derecha);
start Offset	`startOffset = "30%"`	porcentaje
glyph-orientation-vertical	`glyph-orientation-vertical = "90"`	grados sexagesimales
glyph-orientation-horizontal	`glyph-orientation-horizontal = "270"`	grados sexagesimales
direction	`style = "direction: rtl;"`	rtl (right to left); ltr (left to right);

Transformaciones

Con el atributo transform podemos transformar las imágenes (formas geométricas, trazados, imágenes. . .) dibujadas en el lienzo SVG. Las trasformaciones disponibles en SVG incluyen trasladar con translate, redimensionar con scale y girar con rotate.

Trasladar con translate

Para trasladar objetos utilizamos translate, que toma dos parámetros: tx y ty.

transform="translate(tx,ty)"

El parámetro tx recoge el desplazamiento horizontal de la capa mientras que el parámetro ty recoge el desplazamiento vertical de la misma. Si ty no esta especificado el SVG considera que ty = 0.

```
<svg width="250" height="150" viewBox="0 0 250 150">
   <rect x="60" y="10" height="80" width="100" stroke="#003300"
fill="#6ab150" fill-opacity="0.5">
   </rect>
   <rect transform="translate(30,20)" x="60" y="10" height="80"
width="100" stroke="#003300" fill="#6ab150" fill-opacity="0.9">
   </rect>
```

```
</svg>
```

Girar con rotate

Para girar objetos en SVG utilizamos rotate.

transform="rotate(a cx cy)"

donde a es el ángulo de rotación en grados sexagesimales, y cx y cy son las coordenadas del centro de rotación. Si no especificamos los parámetros cx y cy SVG considera que la rotación se hace alrededor del origen del lienzo SVG, o sea la esquina izquierda arriba de este

```
<svg width="250" height="200" viewBox="0 0 250 200">
  <rect x="70" y="20" height="80" width="100" stroke="#003300"
fill="#6ab150" fill-opacity="0.5">
  </rect>
  <rect transform="rotate(25)" x="70" y="20" height="80" width="100"
stroke="#003300" fill="#6ab150" fill-opacity="0.9">
  </rect>
</svg>
```

Redimensionar con scale

Para redimensionar objetos en SVG utilizamos el método scale().

transform="scale(sx sy)"

Con scale podemos reducir o ampliar a escala el dibujo actual, donde sx representa la escala horizontal, y sy la escala vertical. Por ejemplo: si sx = 0.6 la anchura se verá reducida a un 60% de su valor inicial.

```
<svg width="250" height="150" viewBox="0 0 250 150">
  <rect x="20" y="10" height="80" width="100" stroke="#006600" stroke-
width="5" fill="#6ab150" fill-opacity="0.5">
  </rect>
  <rect transform="scale(0.6, 0.6)" x="20" y="10" height="80" width="100"
stroke="#006600" stroke-width="5" fill="#6ab150" fill-opacity="0.9">
  </rect>
</svg>
```

Animaciones con Animate

Pasos a seguir:

1º) Anidar dentro de un elemento geométrico un elemento anímate.

2º) Modificar el atributo que queremos animar mediante attributeName.

3º) Podemos utilizar los atributos from y to para indicar donde empieza y acaba la animación.

4º) Indicar la duración de la animación con el atribute dur.

h ➤ horras

min ➤ minutos

s ➤ segundos

ms ➤ milisegundos

5º) Si queremos que la animación se repita indefinidamente se utiliza el atributo repeatCount.

```
<svg width="400" height="120">
    <rect x="-110" y="20" width="100" height="80" style="fill:#6ab150;
stroke:#000; stroke-width:5;">
        <animate attributeName="x" attributeType="XML" from="-110"
to="400" dur="5s" repeatCount="indefinite"></animate>
    </rect>
</svg>
```

Si queremos que la animación pase una sola vez, tenemos que utilizar fill="freeze".

Utilizamos el atributo values para comunicar al SVG los valores que queremos que tome el atributo x durante la animación. Los valores van separadas por un punto y coma (;).

Trasladar - animateTransform

El elemento <animateTransform> anima el atributo transform de los objetos SVG de la misma manera como <animate> anima los atributos con valores numéricos y los colores.

Pasos a seguir:

1º) Especificar el nombre del atributo que queremos animar: attributeName=”transform”.

2º) El atributo “type” indica la transformación: translate, scale, rotate.

3º) El atributo “values” son los puntos que por los cuales se realiza la transformación.

El atributo calcMode controla el movimiento entre los puntos clave de la animación (decimos que establece una interpolación de movimiento). El atributo calcMode puede tomar uno de estos valores: *calcMode = "discrete | linear | paced | spline"*

Si calcMode="discrete" el objeto animado salta de un punto a otro sin interpolación alguna.

Si calcMode="linear" el objeto animado se mueve a velocidad constante de un punto a otro (se trata de una interpolación lineal entre valores).

Si calcMode="linear" la animación acepta otro atributo: keyTimes, que define la velocidad entre los puntos clave de la animación.

Animar con animateMotion

El elemento <animateMotion> nos permite, además de esto, animar objetos SVG a lo largo de trayectorias complejas definidas con <path>.

```
<svg width="250" height="200">
<g transform="translate(55,100)">
<path style="stroke: #00f;stroke-width:3; fill: none;" d="m0,0 a70,70 0 1, 0 0,-1 z"></path>
<polygon points="0,0 -10,-10 -10,10" style="fill: #ff0000;">

<animateMotion path="m0,0 a70,70 0 1, 0 0,-1 z" dur="10s" rotate="auto" repeatCount="indefinite"></animateMotion>
</polygon>
</g>
</svg>
```