Chapitre 2 Formes de base

"What you see is all you get."
- Brian Kernighan4

"In theory, there is no difference between theory and practice. But, in practice, there is."
- Jan L.A. van de Snepscheut

Objectifs du chapitre

L'en-tête SVG

- Les éléments ‘svg’, ‘desc’ et ‘title’
- Attributs de présentation: Stroke and Fill
- Formes de base ‘line’, ‘circle’, ‘rect’, ‘ellipse’, ‘polyline’, ‘polygon’
- Tableau de référence
- L'élément ‘image’

Aperçu

Comme nous l'avons vu au chapitre 1, SVG comporte trois types d'objets : formes, images, texte. Dans ce chapitre nous allons explorer les fondements de la structure d'un document SVG, les attributs de base pour la présentation, formes, les images et aborder le texte. Les aspects du texte seront vus en détail au chapitre 5. Ceci nous permettra de construire des blocs pour ce que nous voulons faire avec SVG.

L'en-tête SVG

Ce code montre la structure complète d'un document SVG. Il est composé d'une déclaration XML, de la déclaration de la DOCTYPE et du fragment de SVG.
Voici l'en-tête complet :

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
                     "http://www.w3.org/TR/SVG/DTD/svg10.dtd">
<svg width="350" height="300" xmlns="http://www.w3.org/2000/svg">
  <!-- content goes here -->
</svg>
```

La première ligne est l'instruction standard XML conforme aux spécifications XML 1.0. Elle précise la table de codage des caractères, ici UTF-8 et indique que le document dépend d'une définition du type de document (DTD) externe au document. Où est localisée la DTD?
La seconde ligne donne cette information. Dans cette seconde ligne le 'DOCTYPE' est accessible à la location de la DTD et le nom du document est précisé, ici 'svg'. La DTD sera appliquée à un élément du document nommé ‘svg’ et indiquera la grammaire et les règles pour celui-ci.

Avec cet en-tête, les lettres accentuées ne seront pas dessinées. Si vous utilisez les lettres accentuées, vous devez remplacer la table de codage UTF-8 par ISO-8859-1 et la première ligne devient

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
```

Voici un exemple simple qui utilise l'en-tête complet et dessine quelques éléments :

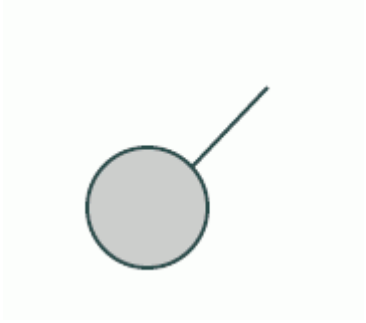

Figure 2-1. Ligne et cercle avec style

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
                     "http://www.w3.org/TR/SVG/DTD/svg10.dtd">
<svg width="350" height="300">
  <title>SVG - Introduction</title>
  <desc> This graphic demonstrates many exciting features of SVG.</desc>
    <circle cx="50" cy="70" r="30" fill="grey"
            fill-opacity="0.4" stroke="darkslategrey"
            stroke-width="2">
      <desc>Basic circle</desc>
    </circle>
    <line x1="72" y1="50" x2="110" y2="10" stroke="darkslategrey"
          stroke-width="2"/>
</svg>
```

Voyons de près les éléments ajoutés au code.

L'élément <svg>

Après l'en-tête, l'élément svg contient tous les autres objets. Il est 'document element' pour le document.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
                     "http://www.w3.org/TR/SVG/DTD/svg10.dtd">
<svg width="350" height="300">
  <!-- content goes here -->
</svg>
```

Vous remarquez les attributs de l'élément 'svg'. Que signifient-ils ?
Les attributs ‘width’ et ‘height’ définissent la largeur et la hauteur du graphique SVG. Nous retrouverons ces attributs pour l'élément <rect> par exemple.

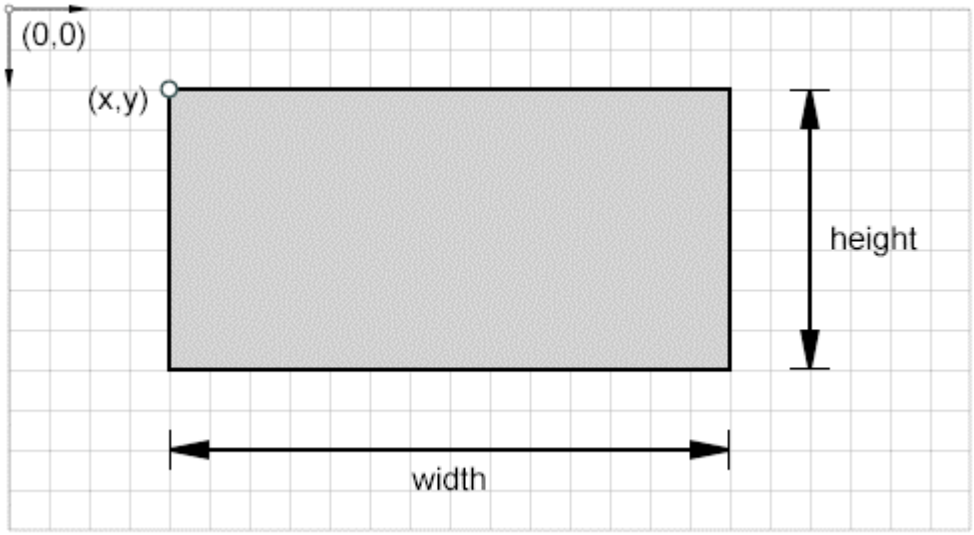

Figure 2-2. Rectangle avec les attributs 'x' 'y' 'width' et 'height'

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
                     "http://www.w3.org/TR/SVG/DTD/svg10.dtd">
<svg width="350" height="300">
  <rect x="40" y="20" width="140" height="70"
      stroke="black" fill="lightgrey" />
</svg>
```

De nombreuses propriétés peuvent être définies pour l'élément 'svg'. Nous les verrons dans le prochain chapitre et au cours de ce livre.

Voyons maintenant les objets ajoutés au document. Vous voyez les balises 'title', 'desc', 'circle', 'line'. Commençons par 'title' et 'desc', nous verrons 'circle' et 'line' un peu plus loin..

Les éléments 'title' et 'desc'

Voyons une nouvelle fois ce code:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
                     "http://www.w3.org/TR/SVG/DTD/svg10.dtd">
<svg width="350" height="300">
  <title>SVG - Introduction</title>
  <desc> This graphic demonstrates many exciting features of SVG.</desc>
    <circle cx="50" cy="70" r="30" fill="grey"
            fill-opacity="0.4" stroke="darkslategrey" stroke-width="2">
      <desc>Basic circle</desc>
    </circle>
    <line x1="72" y1="50" x2="110" y2="10" stroke="darkslategrey"
          stroke-width="2"/>
</svg>
```

En SVG, chaque élément peut avoir les éléments fils 'title' et 'desc'. Chaque élément peut donc avoir un titre et une description. Le contenu de ces balises 'title' et 'desc' n'est pas dessiné.

Les spécifications SVG laissent toute liberté pour l'utilisation de ces éléments au processeur SVG. Ceci signifie qu'un visualiseur peut gérer des fonctionnalités pour ces attributs, un autre non. Par exemple un visualiseur SVG peut utiliser le contenu de 'desc' pour un "tooltip", texte informatif s'affichant quand le pointeur est sur l'objet et celui de 'title' de l'élément 'svg' comme titre pour la page Web. Cet élément ‘title’ en SVG est différent de l'attribut 'title' en HTML. En utilisant le SVG DOM et un script, nous pouvons créer nos propres “tooltips”.

Note au lecteur

Dans la suite de ce livre, nous laisserons de côté l'en-tête SVG pour nous concentrer sur les éléments et attributs SVG et améliorer la lisibilité du code. Ceci dit, nous vous recommandons de toujours mettre l'en-tête complète dans vos documents SVG. Si vous utilisez le visualiseur de Batik, sans en-tête vos documents ne seront pas affichés. Donc pour la lisibilité du code les exemples de code prendront cette forme :

```
<?xml version="1.0"?>
<svg width="350" height="300">
  <!-- content goes here -->
</svg>
```

Unités de mesure: Nous n'utiliserons aucune unité CSS (pt, mm, px ...) dans la suite du livre. Ceci car le groupe de travail SVG examine de meilleurs choix pour les unités dans les futures versions de SVG. Le groupe de travail recommande d'éviter ces unités pour l'instant et nous suivons cette recommandation. En ne spécifiant pas d'unité, notre graphique SVG aura comme unité le type par défaut soit 'user units'. Nous exprimerons les coordonnées ainsi.

Propriétés du style

Les objets graphiques doivent être stylisés. Nous allons voir comment appliquer les propriétés du style aux formes en utilisant les attributs de présentation.

Style avec CSS ou Attributs de présentation

Nous pouvons appliquer un style de nombreuses manières. Dans la suite du livre nous utiliserons les attributs de présentation plutôt que CSS pour appliquer un style à nos images.

Un attribut de présentation a la syntaxe suivante :

name of style = “value”

Un style CSS a cette syntaxe:

style = “name of style : value”

Les attributs de présentation tels que fill="red" seront utilisés dans les exemples de code plutôt que des styles **CSS styles** tels que style="fill:red".

Couleurs

Nous ne pouvons aborder les propriétés 'fill' et ‘stroke’ sans dire comment définir une couleur.

Les couleurs en SVG viennent des spécifications CSS. Il y a deux manières principales de définir une couleur, par son nom ou sa valeur **RGB**.

Si vous voulez que votre rectangle ait son intérieur coloré en gris clair, vous pouvez utiliser le nom "lightgray".

Figure 2-3. Rectangle gris clair

```
<svg width="350" height="300">
      <rect x="50" y="30" width="200" height="100" fill=”lightgray” />
</svg>
```

Les couleurs peuvent être définies par une notation hexadécimale avec 3 ou 6 nombres, par rgb(000,000,000) ou rgb(0%,0%,0%).
Voici quatre manières d'indiquer la couleur blanche:

```
#FFF = #FFFFFF = rgb(255,255,255) = rgb(100%,100%,100%)

Avec l'utilisation de ces quatre notations:

<rect x="50" y="30" width="200" height="100" fill=”#fff” />
<rect x="50" y="30" width="200" height="100" fill=”#ffffff” />
<rect x="50" y="30" width="200" height="100" fill=”rgb(255,255,255)” />
<rect x="50" y="30" width="200" height="100" fill=”rgb(100%,100%,100%)” />
```

Vous trouverez la liste des noms de couleurs reconnus avec leur valeur hexadécimale à l'annexe A.

Les propriétés ‘fill’ et ‘stroke’

Tous les objets graphiques ont un pourtour et un intérieur. Le pourtour est nommé ‘stroke’ et l'intérieur ‘fill’.

Vous pouvez donner des valeurs aux propriétés ‘stroke’ et ‘fill’ de l'objet avec les attributs de présentation.

Propriétés de ‘fill’

Vous pouvez utiliser la propriété ‘fill’ d'un objet graphique en indiquant une couleur, un gradient ou une texture. Nous traiterons des gradients et des textures au chapitre 7. L'attribut ‘fill’ indique la couleur pour l'intérieur de l'objet. La propriété ‘fill’ est utilisée pour remplir l'intérieur avec la valeur ‘paint’ spécifiée.

Table de référence pour la propriété 'fill'

La propriété 'fill' inclut:
fill
fill-opacity
fill-rule

Chacune de ces propriétés est décrite ci-dessous

Propriétés de remplissage:

Propriété	Description
fill	Une couleur qui définit l'intérieur de la forme ou du texte.
fill-opacity	Un nombre entre 0 (transparent) et 1 (opaque) qui définit l'opacité de la forme ou du texte.
fill-rule	Une valeur ‘nonzero’ ou ‘evenodd’ qui définit la manière de déterminer l'intérieur d'une forme entrelacée.

fill

Voici la syntaxe de la propriété ‘fill’.

'fill'

Valeur:	<paint>
Par défaut:	black
S'applique à:	Formes et texte
Transmissible:	oui
Pourcentages:	N/A
Media:	visuel
Animable:	oui

Voici quelques valeurs pour l'attribut ‘fill’ avec cette figure 2-4.

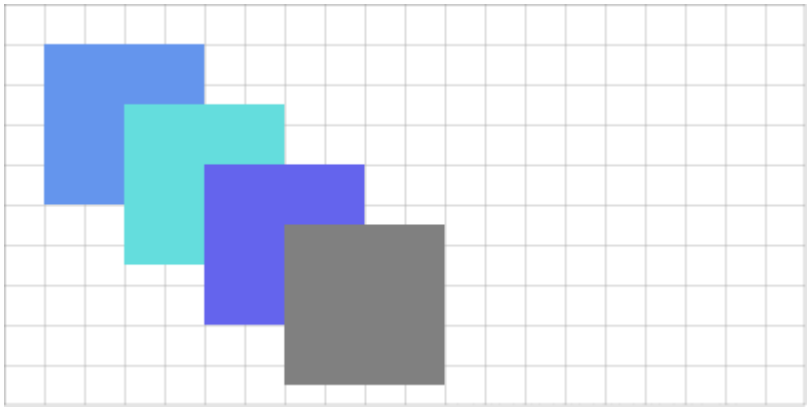

Figure 2-4. Couleurs de remplissage

```
<svg width="350" height="300">
      <rect x="10" y="10" width="40" height="40" fill="cornflowerblue"/>
      <rect x="30" y="25" width="40" height="40" fill="#64DDDD"/>
      <rect x="50" y="40" width="40" height="40" fill="rgb(100, 100, 237)"/>
      <rect x="70" y="55" width="40" height="40" fill="rgb(50%,50%,50%)"/>
</svg>
```

fill-opacity

Valeur:	<opacity-value> | inherit
Par défaut:	1
S'applique à:	éléments formes et texte
Transmissible:	oui

Pourcentages:	N/A
Media:	visuel
Animable:	oui

La propriété ‘fill-opacity’ définit l'opacité de l'intérieur de l'objet.

La valeur par défaut est 1 (**opaque**). Elle varie de 0 (**transparent**) à 1.
Pour la figure 2-5 nous avons indiqué des valeurs variées.

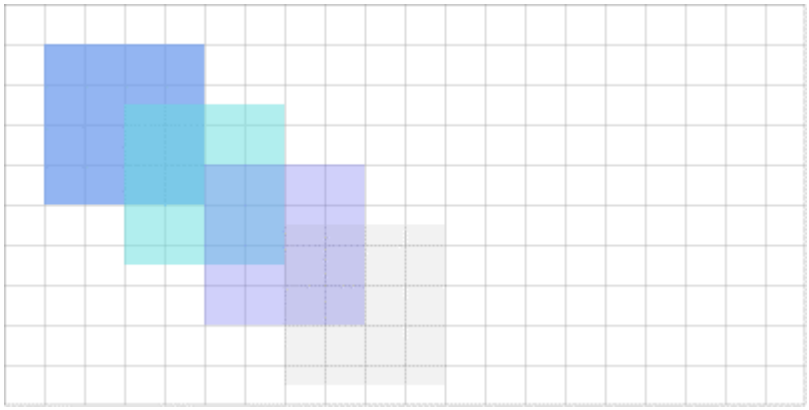

Figure 2-5. Propriété fill-opacity

```
<svg width="200" height="200">
      <rect x="5" y="5" width="50" height="50"
            fill="cornflowerblue" fill-opacity=”0.7” />
      <rect x="25" y="25" width="50" height="50"
            fill="#64DDDD" fill-opacity=”0.5” />
      <rect x="45" y="45" width="50" height="50"
            fill="rgb(100, 100, 237)" fill-opacity=”0.3” />
      <rect x="65" y="65" width="50" height="50"
            fill="rgb(50%,50%,50%)" fill-opacity=”0.1” />
</svg>
```

fill-rule

Valeur:	nonzero | evenodd | inherit
Par défaut:	Nonzero
S'applique à:	éléments formes et texte
Transmissible:	oui
Pourcentages:	N/A
Media:	Visuel
Animable:	oui

La propriété ‘fill-rule’ détermine les parties intérieure et extérieure d'une forme. Ce n'est pas toujours évident pour des formes complexes. Voici un exemple tiré des spécifications SVG 1.2. Nous reverrons en détail cette propriété dans le chapitre 4.

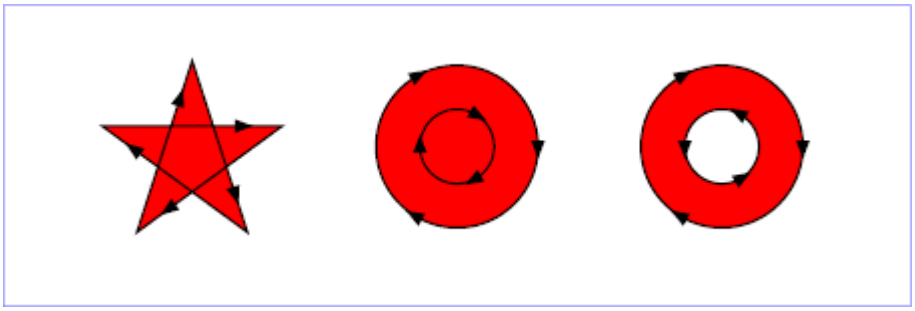

Figure 2-6. Propriété fill-rule

```
<?xml version="1.0" standalone="no"?>
<svg width="12cm" height="4cm" viewBox="0 0 1200 400" version="1.1"
     xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
  <desc>Example fillrule-nonzero - demonstrates fill-rule:nonzero</desc>

  <rect x="1" y="1" width="1198" height="398"
        fill="none" stroke="blue" />
  <defs>
    <path id="Triangle" d="M 16,0 L -8,9 v-18 z" fill="black" stroke="none" />
  </defs>
  <g fill-rule="nonzero" fill="red" stroke="black" stroke-width="3" >
    <path d="M 250,75 L 323,301 131,161 369,161 177,301 z" />
    <use xlink:href="#Triangle" transform="translate(306.21 249) rotate(72)"
overflow="visible"  />
    <use xlink:href="#Triangle" transform="translate(175.16,193.2) rotate(216)"
overflow="visible"  />
    <use xlink:href="#Triangle" transform="translate(314.26,161) rotate(0)"
overflow="visible"  />
    <use xlink:href="#Triangle" transform="translate(221.16,268.8) rotate(144)"
overflow="visible"  />
    <use xlink:href="#Triangle" transform="translate(233.21,126.98)
rotate(288)" overflow="visible"  />
    <path d="M 600,81 A 107,107 0 0,1 600,295 A 107,107 0 0,1 600,81 z
             M 600,139 A 49,49 0 0,1 600,237 A 49,49 0 0,1 600,139 z" />
    <use xlink:href="#Triangle" transform="translate(600,188) rotate(0)
translate(107,0) rotate(90)" overflow="visible"  />
    <use xlink:href="#Triangle" transform="translate(600,188) rotate(120)
translate(107,0) rotate(90)" overflow="visible"  />
    <use xlink:href="#Triangle" transform="translate(600,188) rotate(240)
translate(107,0) rotate(90)" overflow="visible"  />
    <use xlink:href="#Triangle" transform="translate(600,188) rotate(60)
translate(49,0) rotate(90)" overflow="visible"  />
    <use xlink:href="#Triangle" transform="translate(600,188) rotate(180)
translate(49,0) rotate(90)" overflow="visible"  />
    <use xlink:href="#Triangle" transform="translate(600,188) rotate(300)
translate(49,0) rotate(90)" overflow="visible"  />
    <path d="M 950,81 A 107,107 0 0,1 950,295 A 107,107 0 0,1 950,81 z
             M 950,139 A 49,49 0 0,0 950,237 A 49,49 0 0,0 950,139 z" />
    <use xlink:href="#Triangle" transform="translate(950,188) rotate(0)
translate(107,0) rotate(90)" overflow="visible"  />
    <use xlink:href="#Triangle" transform="translate(950,188) rotate(120)
translate(107,0) rotate(90)" overflow="visible"  />
    <use xlink:href="#Triangle" transform="translate(950,188) rotate(240)
translate(107,0) rotate(90)" overflow="visible"  />
    <use xlink:href="#Triangle" transform="translate(950,188) rotate(60)
translate(49,0) rotate(-90)" overflow="visible"  />
```

```
    <use xlink:href="#Triangle" transform="translate(950,188) rotate(180)
translate(49,0) rotate(-90)" overflow="visible"  />
    <use xlink:href="#Triangle" transform="translate(950,188) rotate(300)
translate(49,0) rotate(-90)" overflow="visible"  />
  </g>
</svg>
```

Propriétés de tracé

Quand nous utilisons la propriété ‘stroke’, le pourtour de la forme ou du texte sera dessiné avec la couleur, le gradient ou la texture indiquée.
Il y a plusieurs propriétés relatives au pourtour, parmi celles-ci :

stroke
stroke-width
stroke-opacity
stroke-linecap
stroke-linejoin
stroke-dasharray
stroke-dashoffset

Chacune de ces propriétés est décrite ci-dessous.

Propriétés pour le tracé du pourtour:

Propriété	Description
stroke	Une couleur pour le tracé du pourtour.
stroke-width	Un nombre ou un pourcentage qui définit l'épaisseur du tracé.
stroke-opacity	Un nombre entre 0 (transparent) et 1 (opaque) pour définir l'opacité du tracé.
stroke-dasharray	Une série de nombres qui définit la longueur du tracé et du non-tracé dans un pointillé.
stroke-dashoffset	Un nombre qui définit le point de départ du tracé en pointillé.
stroke-linecap	Une valeur ‘butt’ (par défaut), ‘round’ ou ‘square’ qui spécifie la forme du tracé à la fin de chaque segment
stroke-linejoin	Une valeur 'miter' (par défaut), 'round' ou 'bevel' qui indique la forme du tracé aux sommets d'une forme.

stroke et stroke-width

Voici la syntaxe pour ces propriétés ‘stroke’ et ‘stroke-width’.

'stroke'

Valeur:	<paint>
Par défaut:	none
S'applique à:	éléments formes et texte
Transmissible:	oui
Pourcentages:	N/A
Media:	visuel
Animable:	oui

'stroke-width'

Valeur:	<length> | inherit
Par défaut:	1
S'applique à:	éléments formes et texte
Transmissible:	oui
Pourcentages:	oui
Media:	visuel
Animable:	oui

Dans cet exemple nous avons utilisé les attributs de présentation pour définir les propriétés 'stroke color' et 'stroke-width' de ces lignes.

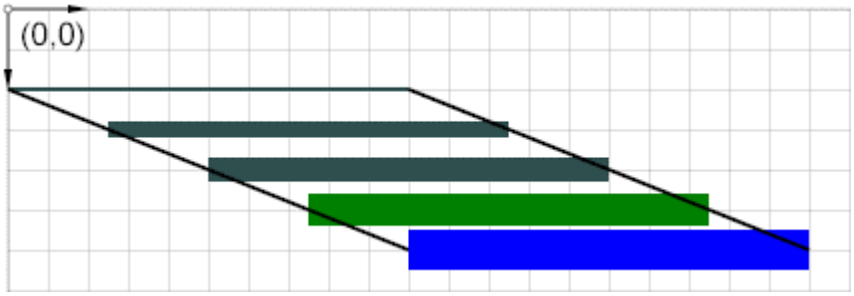

Figure 2-7. Propriétés stroke et stroke-width

```
<svg>
      <!-- Horizontal Lines -->
      <line x1="0" y1="20" x2="100" y2="20"
            stroke="darkslategray"/>
      <line x1="25" y1="30" x2="125" y2="30"
            stroke="darkslategray" stroke-width="4"/>
      <line x1="50" y1="40" x2="150" y2="40"
            stroke="darkslategray" stroke-width="6"/>
      <line x1="75" y1="50" x2="175" y2="50"
            stroke="green" stroke-width="8"/>
      <line x1="100" y1="60" x2="200" y2="60"
            stroke="blue" stroke-width="10"/>
      <!-- Diagonal Lines -->
      <line x1="0" y1="20" x2="100" y2="60"
            stroke="black" stroke-width="1"/>
      <line x1="100" y1="20" x2="200" y2="60"
            stroke="black" stroke-width="1"/>
</svg>
```

Comme vous le voyez, l'élément ‘line’ a différents aspects suivant les valeurs données aux propriétés ‘stroke’ et ‘stroke-width’.

Le tracé est toujours centré sur le pourtour de l'objet. Ce qui signifie que la moitié du tracé est à l'intérieur de l'objet et l'autre moitié à l'extérieur

stroke-opacity

Valeur:	<opacity-value> | inherit
Par défaut:	1
S'applique à:	éléments formes et texte
Transmissible:	oui
Pourcentages:	N/A
Media:	visuel
Animable:	oui

Ce nouvel exemple utilise la propriété ‘stroke-opacity’:

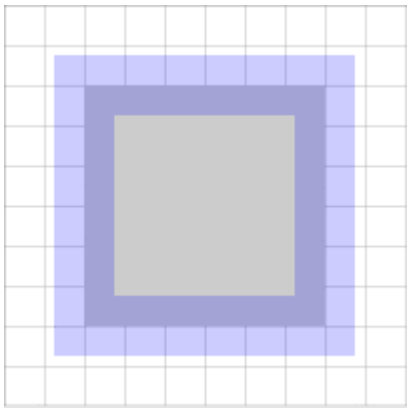

Figure 2-8. Stroke-width expliqué par stroke-opacity

```
<svg width="200" height="200">
      <rect  x="20" y="20" width="60" height="60"
             fill="#CCCCCC" stroke="blue" stroke-width="15"
             stroke-opacity="0.2"/>
</svg>
```

Comme vous pouvez le voir sur cet exemple, l'utilisation des propriétés 'stroke-width' et 'stroke-opacity' permet de créer des effets intéressants. Nous semblons avoir trois rectangles différents. En affectant un style à un objet, nous pouvons avoir quelques surprises. Par exemple qu'arrive-t-il si la valeur de ‘stroke-width’ est plus grande que l'intérieur de l'objet? Essayez! Comme vous le pressentez, le tracé peut entièrement occuper l'intérieur de l'objet et recouvrir le remplissage.

stroke-dasharray

Valeur:	none | <dasharray> | inherit
Par défaut:	aucune
S'applique à:	éléments formes et texte
Transmissible:	oui
Pourcentages:	oui
Media:	visuel
Animable:	oui (non-additive)

En utilisant seulement **stroke-dasharray** nous pouvons créer différents types de lignes pointillées.

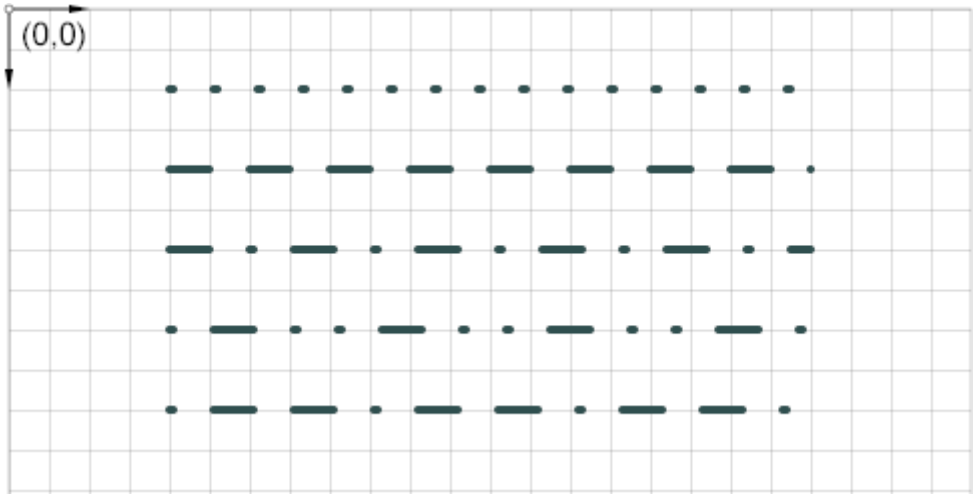

Figure 2-9. Propriété stroke-dasharray

```
<svg width="350" height="300">
  <line x1="40" y1="20" x2="200" y2="20"
      stroke-dasharray="1 10" fill="none"
      stroke="darkslategray" stroke-width="2" stroke-linecap="round" />
  <line x1="40" y1="40" x2="200" y2="40"
      stroke-dasharray="10 10 10 10" fill="none"
      stroke="darkslategray" stroke-width="2" stroke-linecap="round" />
  <line x1="40" y1="60" x2="200" y2="60"
      stroke-dasharray="10 10 1 10" fill="none"
      stroke="darkslategray" stroke-width="2" stroke-linecap="round" />
  <line x1="40" y1="80" x2="200" y2="80"
      stroke-dasharray="1 10 10 10 1 10" fill="none"
      stroke="darkslategray" stroke-width="2" stroke-linecap="round" />
  <line x1="40" y1="100" x2="200" y2="100"
      stroke-dasharray="1 10 10 10 10 10" fill="none"
      stroke="darkslategray" stroke-width="2" stroke-linecap="round" />
</svg>
```

Note: Quand vous utilisez les lignes avec le visualiseur d'Adobe, vous devez toujours indiquer fill=”none” pour éviter qu'une fine ligne ne soit dessinée tout au long de votre tracé.

stroke-dashoffset

Valeur:	<length> | inherit
Par défaut:	0
S'applique à:	éléments formes et texte
Transmissible:	oui
Pourcentages:	oui
Media:	visuel
Animable:	oui

La propriété '**stroke-dashoffset**' peut être utilisée pour déplacer la position de départ de ‘stroke-dasharray’. La propriété ‘stroke-dasharray’ permet également des tracés progressifs avec un élément 'animate', voyez quelques exemples au chapitre 9.

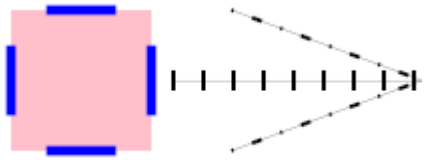

Figure 2-10. Stroke-dasharray et stroke-dashoffset

```
<svg width="300" height="300">
      <rect x="10" y="10" width="70" height="70"
            fill="pink" stroke="blue" stroke-width="5"
            stroke-dasharray="35 35" stroke-dashoffset="-17.5"/>
      <line x1="90" y1="45" x2="215" y2="45"
            stroke-width="10" stroke="black"
            stroke-dasharray="2 13"/>
      <line x1="120" y1="80" x2="215" y2="45"
            stroke-width="2" stroke="black"
            stroke-dasharray="1 10 5 10"/>
      <line x1="120" y1="10" x2="215" y2="45"
            stroke-width="2" stroke="black"
            stroke-dasharray="1 10 5 10"/>
</svg>
```

Dans l'exemple ci-dessus, les pointillés des côtés du rectangle ont été déplacés avec ‘stroke-dashoffset’ afin d'être centrés sur les côtés.

En utilisant ‘stroke-dasharray’ sur deux lignes, nous pouvons créer ce quadrillage.

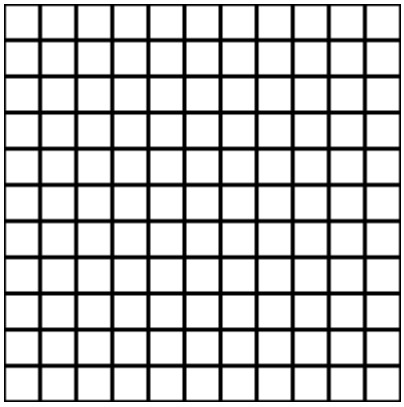

Figure 2-11. Quadrillage avec stroke-dasharray

```
<svg width="100" height="100">
      <line x1="0" y1="50" x2="100" y2="50"
            stroke-width="100" stroke="black"
            stroke-dasharray="2 18" />
      <line x1="50" y1="0" x2="50" y2="100"
            stroke-width="100" stroke="black"
            stroke-dasharray="2 18" />
</svg>
```

La propriété ‘stroke-dashoffset’ est encore plus explicite sur cet exemple.

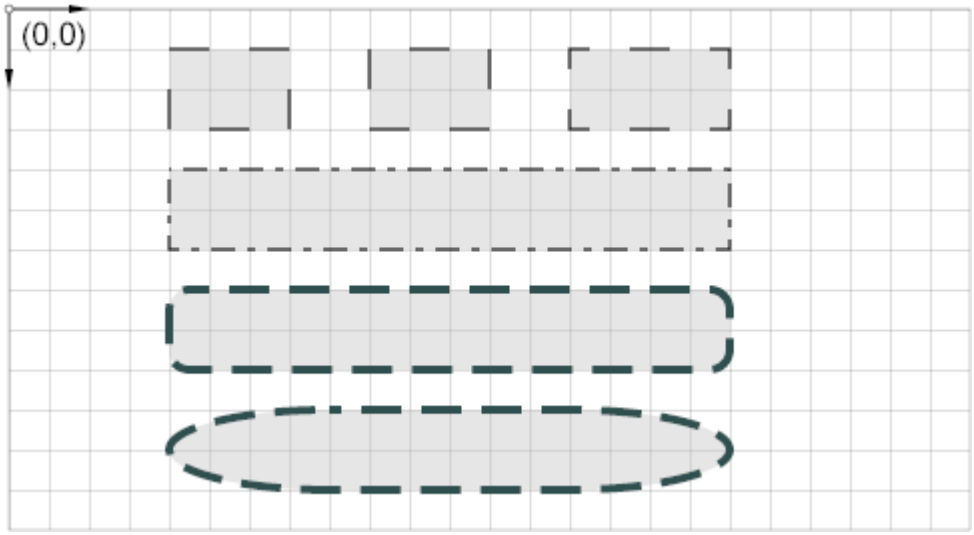

Figure 2-12. La propriété stroke-dashoffset

```
<svg width="350" height="300">
  <rect x="40" y="10" width="30" height="20"
      stroke="black" stroke-width="0.75"
      stroke-dasharray="10 10 10 10" stroke-dashoffset="0"
      fill="silver" fill-opacity="0.4" />
  <rect x="90" y="10" width="30" height="20"
      stroke="black" stroke-width="0.75"
      stroke-dasharray="10 10 10 10" stroke-dashoffset="-10"
      fill="silver" fill-opacity="0.4" />
  <rect x="140" y="10" width="40" height="20"
      stroke="black" stroke-width="0.75"
      stroke-dasharray="10 10 10 10" stroke-dashoffset="5"
      fill="silver" fill-opacity="0.4" />
  <rect x="40" y="40" width="140" height="20"
      stroke="black" stroke-width="0.75"
      stroke-dasharray="6 3 2 3" stroke-dashoffset="20"
      fill="silver" fill-opacity="0.4"/>
  <rect x="40" y="70" width="140" height="20" rx="5" ry="5"
      stroke="darkslategray" stroke-width="2"
      stroke-dasharray="10 5 10 5" stroke-dashoffset="500"
      fill="silver" fill-opacity="0.4" />
  <rect x="40" y="100" width="140" height="20" rx="40" ry="20"
      stroke="darkslategray" stroke-width="2"
      stroke-dasharray="10 5 10 5" stroke-dashoffset="7"
      fill="silver" fill-opacity="0.4" />
</svg>
```

Les propriétés ‘stroke-dasharray’ et ‘stroke-dashoffset’ sont très utiles pour le graphisme et l'animation comme nous le verrons à propos de l'élément 'path' dans le chapitre 6.

stroke-linecap

Valeur:	butt | round | square | inherit
Par défaut:	butt
S'applique à:	éléments formes et texte
Transmissible:	oui
Pourcentages:	N/A
Media:	visuel
Animable:	oui

La valeur par défaut de la propriété ‘stroke-linecap’ qui est ‘butt’ signifie que le tracé s'arrête brutalement aux extrémités de votre ligne. Avec la valeur ‘round’, un demi-cercle de rayon égal à la moitié de 'stroke-width' est dessiné aux extrémités du tracé. La dernière valeur possible est ‘square’ et dans ce cas le tracé est allongé de la moitié de la valeur de ‘stroke-width’ à chaque extrémité.

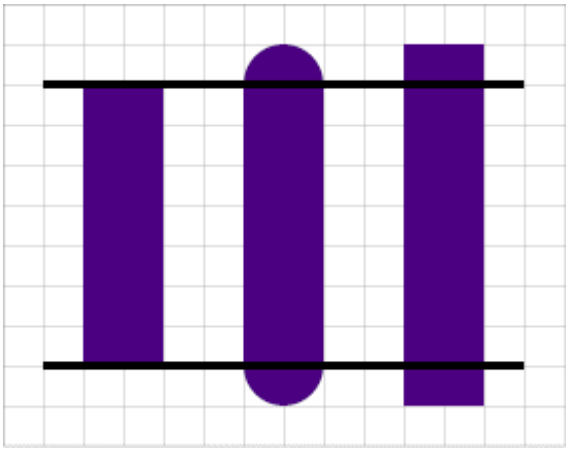

Figure 2-13. La propriété stroke-linecap

```
<svg>
  <!-- Line 1 -->
  <line x1="20" y1="20" x2="20" y2="120"
        fill="black" stroke="indigo" stroke-width="20"
        stroke-linecap="butt" />
  <!-- Line 2 -->
  <line x1="60" y1="20" x2="60" y2="120"
        fill="black" stroke="indigo" stroke-width="20"
        stroke-linecap="round" />
  <!-- Line 3 -->
  <line x1="100" y1="20" x2="100" y2="120"
        fill="black" stroke="indigo" stroke-width="20"
        stroke-linecap="square" />
  <!-- Guide Lines -->
  <line x1="0" y1="20" x2="120" y2="20"
        fill="black" stroke="black" stroke-width="2"/>
  <line x1="0" y1="120" x2="120" y2="120"
        fill="black" stroke="black" stroke-width="2"/>
</svg>
```

stroke-linejoin

Valeur:	miter | round | bevel | inherit
Par défaut:	miter
S'applique à:	éléments formes et texte
Transmissible:	oui
Pourcentages:	N/A
Media:	visuel
Animable:	oui

La valeur par défaut de la propriété ‘stroke-linejoin’ est ‘miter’ qui donne des angles bruts pour le tracé. Les autres valeurs sont ‘round’ et ‘bevel’ avec des coins arrondis ou coupés.

Figure 2-14. Stroke-linejoin 'round' 'bevel' et 'mitter'

```
<svg width="250" height="200">
  <rect x="10" y="10" width="50" height="50"
        fill="magenta" stroke="black" stroke-width="10"
        stroke-linejoin="round" />
  <rect x="80" y="10" width="50" height="50"
        fill="maroon" stroke="black" stroke-width="15"
        stroke-linejoin="bevel"      />
  <rect x="150" y="10" width="50" height="50"
        fill="gray" stroke="black" stroke-width="10"
        stroke-linejoin="miter" />
</svg>
```

Nous verrons plus en détail le style au chapitre 5. Pour le moment continuons sur les formes de base.

Formes de base

Autour de nous nous pouvons voir un nombre infini de formes, des formes naturelles comme les arbres, les fruits et les nuages mais aussi des formes créées comme les symboles, les vêtements et les ordinateurs. SVG permet de modéliser facilement ces formes avec six formes prédéfinies.

Ces formes de base sont:
rect
circle
ellipse
line
polyline
polygon

Chaque forme a ses attributs spécifiques qui définissent ses dimensions.

Par exemple, cette image est créée en utilisant deux éléments 'rect'.

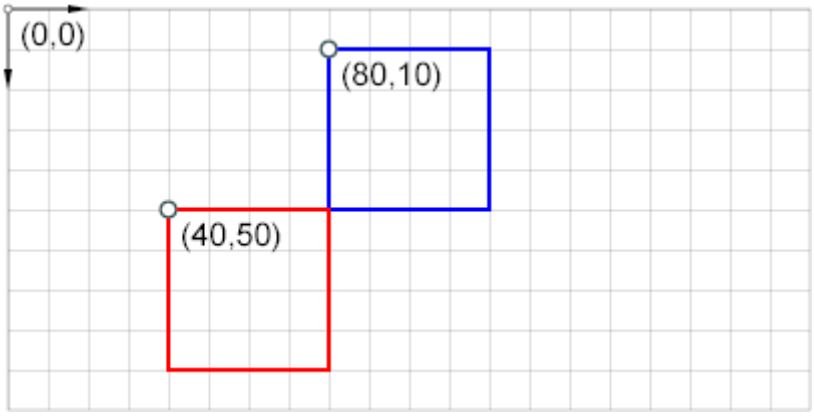

Figure 2-15. Deux carrés

```
<svg width="300" height="300">
      <rect x="80" y="10" width="40" height="40"
            fill="none" stroke="blue" />
      <rect x="40" y="50" width="40" height="40"
            fill="none" stroke="red" />
</svg>
```

Voyons chacune de ces formes de base en détail.

Lignes droites

L'élément ‘line’ possède quatre attributs:

(x1, y1, x2, y2)

Ces quatre attributs définissent les coordonnées des extrémités de cette ligne droite. Voici un exemple où sont représentés les attributs.

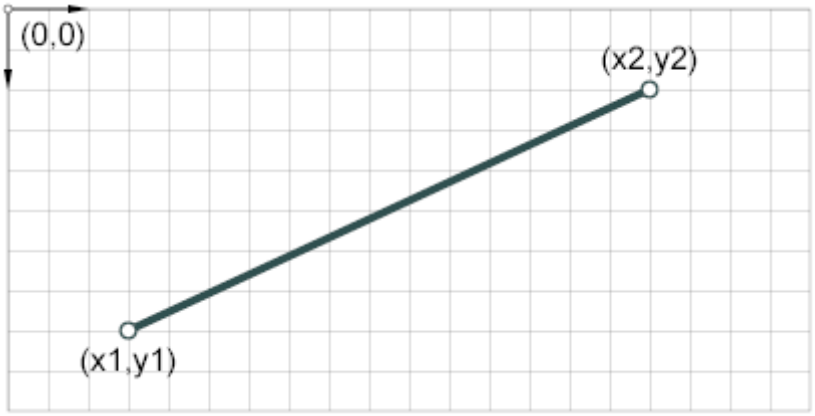

Figure 2-16. L'élément 'line'

```
<svg width="350" height="300">
  <line x1="30" y1="80" x2="160" y2="20"
      stroke="darkslategray" stroke-width="2" stroke-linecap="round" />
</svg>
```

Cette ligne commence au point situé à 30 unités de la gauche et 80 unités du haut du document SVG. Elle se termine à 160 unités de la gauche et 20 unités du haut.

Voici la syntaxe de l'élément ‘line’

```
<line id="name"
      x1="coordinate"
      y1="coordinate"
      x2="coordinate"
      y2="coordinate"
      style-attribute="style-value"/>
```

Les attributs x1, y1, x2 et y2 ont 0 comme valeur par défaut.

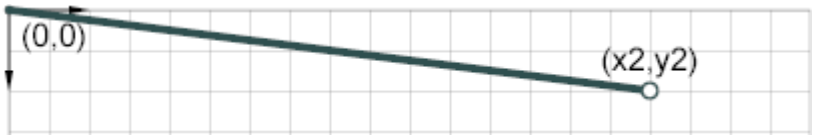

Figure 2-17. Valeurs par défaut pour 'line'

```
<svg width="350" height="300">
  <line x1="0" y1="0" x2="0" y2="0"
        stroke="darkslategray" stroke-width="2" stroke-linecap="round" />
  <line x2="160" y2="20"
        stroke="darkslategray" stroke-width="2" stroke-linecap="round" />
</svg>
```

Comme vous le voyez dans cet exemple si les points sont confondus ou si les coordonnées ne sont pas spécifiées, la ligne n'est pas dessinée.

Exemples de lignes

Voici quelques dessins intéressants avec seulement une ou deux lignes et en jouant sur les attributs de présentation.

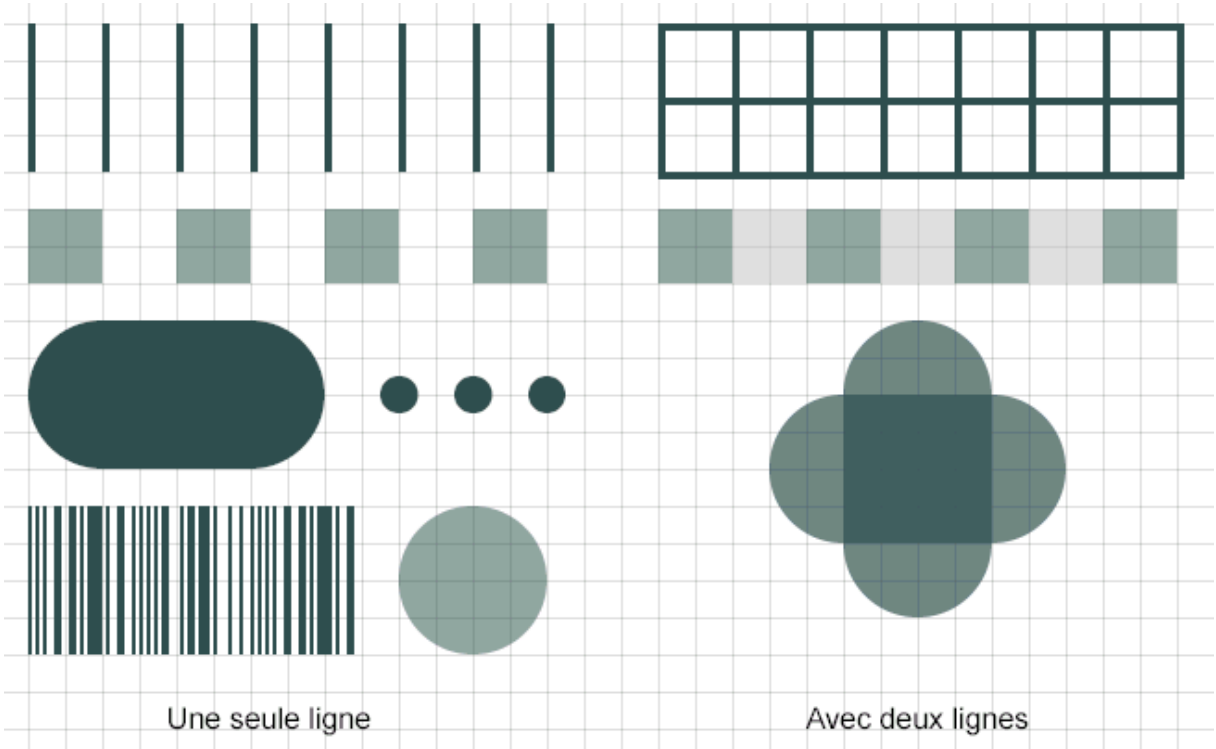

Figure 2-18. Formes étonnantes avec une ou deux lignes

```
<svg width="800" height="600" viewBox="0 0 400 300">
  <defs>
    <pattern id="gridPattern" x="0" y="0" width="10" height="10"
             patternUnits="userSpaceOnUse">
      <path d="M 10 0 L 0 0 0 10" fill="none" stroke="gray"
            stroke-width="0.25"/>
    </pattern>
  </defs>
  <rect id="grid" width="350" height="211" stroke="gray" stroke-width="0.1"
        fill="url(#gridPattern)" />

 <!-- grid -->
  <line x1="10" y1="30" x2="152" y2="30" stroke="darkslategray"
        stroke-width="40" stroke-dasharray="2 18"
        fill="none" />

  <line x1="10" y1="70" x2="150" y2="70" stroke="darkslategray"
        stroke-width="20" stroke-opacity="0.5"
        stroke-dasharray="20 20" fill="none" />

  <line x1="30" y1="110" x2="70" y2="110" stroke="darkslategray"
        stroke-width="40" stroke-linecap="round"
        fill="none" />

  <line x1="110" y1="110" x2="150" y2="110" stroke="darkslategray"
        stroke-width="10" stroke-dasharray="0 20"
        stroke-linecap="round" fill="none" />

  <line x1="130" y1="160" x2="130" y2="160" stroke="darkslategray"
        stroke-width="40" stroke-opacity="0.5"
        stroke-linecap="round" fill="none" />

  <line x1="10" y1="160" x2="100" y2="160" stroke="darkslategray"
        stroke-width="40"
        stroke-dasharray="1 1 1 1 1 2 2 2 2 1 1 1 4 1 1 2 2 2 1 1 1 1 1 1 1 1 2
                          3 1 1 2 1 3 1 1 3 1 2 1 2 1 1"
        fill="none" />

  <text x="75" y="200" font-size="8" text-anchor="middle">single lines</text>

  <line x1="180" y1="31" x2="322" y2="31" stroke="darkslategray"
        stroke-width="42" stroke-dasharray="2 18"
        fill="none" />
  <line x1="251" y1="10" x2="251" y2="52" stroke="darkslategray"
        stroke-width="142" stroke-dasharray="2 18"
        fill="none" />

  <line x1="180" y1="70" x2="320" y2="70" stroke="darkslategray"
        stroke-width="20" stroke-opacity="0.5"
        stroke-dasharray="20 20" fill="none" />
  <line x1="180" y1="70" x2="320" y2="70" stroke="silver"
        stroke-width="20" stroke-opacity="0.5"
        stroke-dasharray="20 20" stroke-dashoffset="20" fill="none" />

  <line x1="230" y1="130" x2="270" y2="130" stroke="darkslategray"
        stroke-width="40" stroke-opacity="0.7"
        stroke-linecap="round" fill="none" />
  <line x1="250" y1="110" x2="250" y2="150" stroke="darkslategray"
```

```
        stroke-width="40" stroke-opacity="0.7"
        stroke-linecap="round" fill="none" />

  <text x="250" y="200" font-size="8" text-anchor="middle">two lines
        each</text>

</svg>
```

Cercles

L'élément 'circle' possède trois attributs géométriques: **'cx', 'cy', et 'r'**.
Les attributs 'cx' et 'cy' donnent la position du centre du cercle tandis que 'r' indique le rayon du cercle.
Si vous voulez créer un cercle avec un diamètre de 80 unités, vous devez donner à l'attribut 'r' la valeur de 40 comme sur cet exemple.

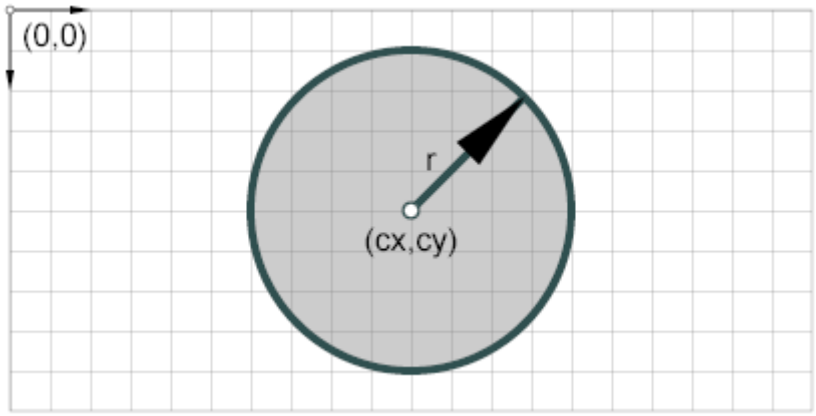

Figure 2-19. L'élément 'circle'

```
<svg width="350" height="300">
  <circle cx="100" cy="50" r="40"
      stroke="darkslategrey" stroke-width="2"
      fill="grey" fill-opacity="0.4"/>
</svg>
```

Voici la syntaxe de l'élément 'circle':

```
<circle id="name"
        cx="coordinate"
        cy="coordinate"
        r="length"
        style-attribute="style-value"/>
```

La valeur par défaut de 'r', 'cx' et 'cy' est 0. Si les valeurs de 'cx' et de 'cy' ne sont pas indiquées, le centre est alors placé en (0, 0). Toutefois une valeur pour 'r' est requise pour que le cercle soit dessiné en SVG. Sur cet exemple le premier cercle n'est pas dessiné.

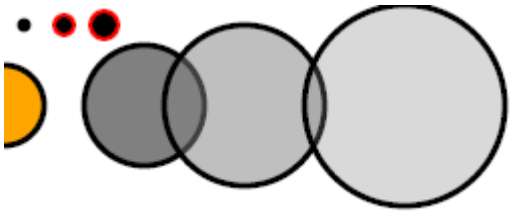

Figure 2-20. Exemples de cercles

```
<svg width="350" height="300">

  <!-- Points -->
  <circle r="0"
          fill="black" stroke="black"/>
  <circle cx="10" cy="10" r="3"
          fill="black" stroke="black"/>
  <circle cx="30" cy="10" r="5" fill="black" stroke="red"
          stroke-width="2"/>
  <circle cx="50" cy="10" r="7" fill="black" stroke="red"
          stroke-width="2"/>

  <!-- Circles -->
  <circle cx="0" cy="50" r="20"
          fill="orange" stroke="black" stroke-width="3" />
  <circle cx="70" cy="50" r="30" fill="grey" stroke="black"
          stroke-width="3" />
  <circle cx="120" cy="50" r="40" fill="grey" fill-opacity="0.5"
          stroke="black" stroke-width="3"/>
  <circle cx="200" cy="50" r="50" fill="grey" fill-opacity="0.3"
        stroke="black" stroke-width="3"/>
</svg>
```

L'élément ‘circle’ peut être utilisé comme forme de base pour des boutons, des roues, des bulles et bien d'autres choses encore comme nous le verrons dans ce livre.

Exemples de cercles

Voici quelques figures réalisées avec un ou deux cercles et en jouant sur les attributs de présentation.

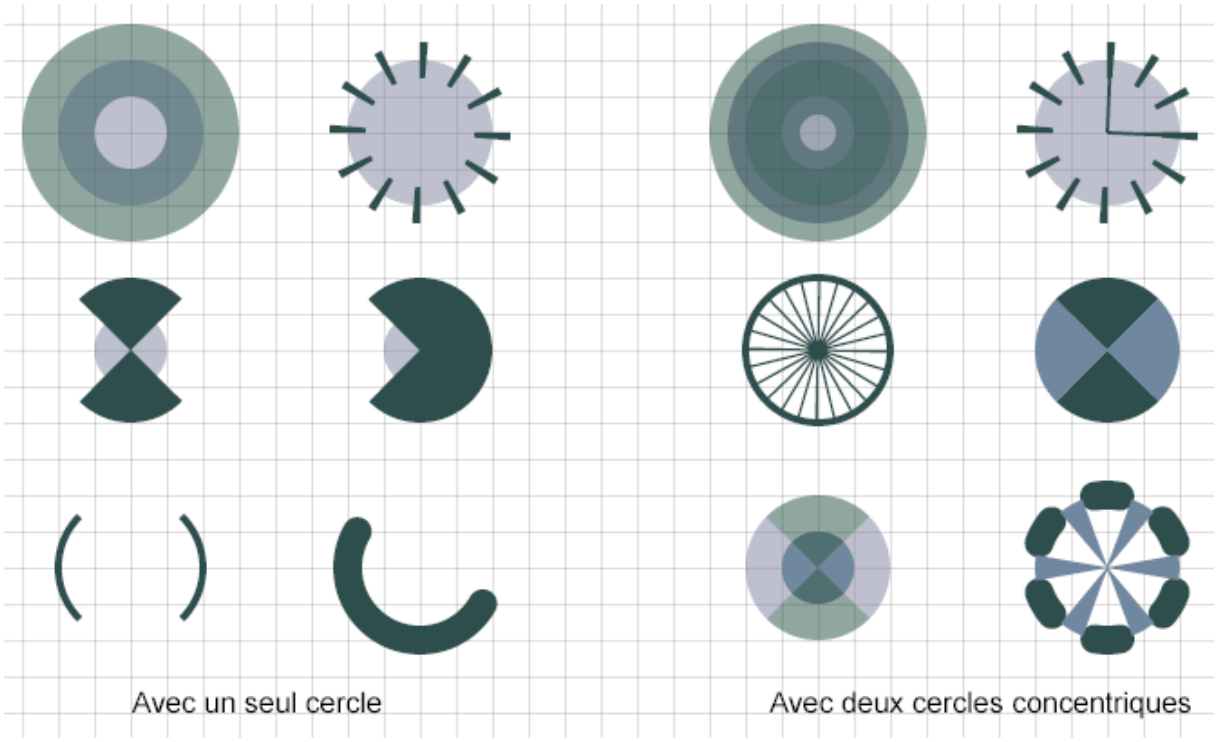

Figure 2-21. Formes étonnantes avec un ou deux cercles

Voyez le code!

```
<svg width="800" height="600" viewBox="0 0 400 300">
  <defs>
    <pattern id="gridPattern" x="0" y="0" width="10" height="10"
             patternUnits="userSpaceOnUse">
      <path d="M 10 0 L 0 0 0 10" fill="none" stroke="gray"
            stroke-width="0.25"/>
    </pattern>
  </defs>
  <rect id="grid" width="350" height="211" stroke="gray" stroke-width="0.1"
       fill="url(#gridPattern)" />

  <circle cx="40" cy="40" r="20" stroke="darkslategray" stroke-width="20"
          stroke-opacity="0.5" fill="lightslategray" fill-opacity="0.5" />

  <circle cx="120" cy="40" r="20" stroke="darkslategray" stroke-width="10"
          stroke-dasharray="1.75 8.72" fill="lightslategray"
          fill-opacity="0.5" />

  <circle cx="40" cy="100" r="10" stroke="darkslategray" stroke-width="20"
          stroke-dasharray="15.708 15.708" stroke-dashoffset="7.854"
          fill="lightslategray" fill-opacity="0.5"/>

  <circle cx="120" cy="100" r="10" stroke="darkslategray" stroke-width="20"
          stroke-dasharray="47.124 15.708" stroke-dashoffset="7.854"
          fill="lightslategray" fill-opacity="0.5"/>

  <circle cx="40" cy="160" r="20" stroke="darkslategray" stroke-width="2"
          stroke-dasharray="31.416 31.416" stroke-dashoffset="-15.708"
          fill="none" />

  <circle cx="120" cy="160" r="20" stroke="darkslategray" stroke-width="8"
          stroke-dasharray="62.832 100" stroke-dashoffset="-41.888"
          fill="none" stroke-linecap="round" />

  <text x="75" y="200" font-size="8" text-anchor="middle">single circle
  each</text>

  <circle cx="230" cy="40" r="20" stroke="darkslategray" stroke-width="20"
          stroke-opacity="0.5" fill="lightslategray" fill-opacity="0.5" />

  <circle cx="230" cy="40" r="15" stroke="darkslategray" stroke-width="20"
          stroke-opacity="0.5" fill="lightslategray" fill-opacity="0.5" />

  <circle cx="310" cy="40" r="20"
          stroke="darkslategray" stroke-width="10" stroke-dasharray="1.75 8.72"
          fill="lightslategray" fill-opacity="0.5" />

  <circle cx="310" cy="40" r="10" stroke="darkslategray" stroke-width="20"
          stroke-dasharray="0.875 14.833 0.875 100" fill="none" />

  <circle cx="230" cy="100" r="20" stroke="darkslategray" stroke-width="2"
          fill="none" />

  <circle cx="230" cy="100" r="10" stroke="darkslategray" stroke-width="20"
          stroke-dasharray="0.524 2.094" fill="none" />

  <circle cx="310" cy="100" r="10" stroke="darkslategray" stroke-width="20"
          stroke-dasharray="15.708 15.708"
          stroke-dashoffset="7.854" fill="none" />
```

```
  <circle cx="310" cy="100" r="10" stroke="lightslategray" stroke-width="20"
          stroke-dasharray="15.708 15.708" stroke-dashoffset="-7.854"
          fill="none" />

  <circle cx="230" cy="160" r="10" stroke="darkslategray" stroke-width="20"
          stroke-dasharray="15.708 15.708" stroke-dashoffset="7.854"
          stroke-opacity="0.5" fill="lightslategray" />
  <circle cx="230" cy="160" r="10" stroke="lightslategray" stroke-width="20"
          stroke-dasharray="15.708 15.708" stroke-dashoffset="-7.854"
          stroke-opacity="0.5" fill="none" />

  <circle cx="310" cy="160" r="20" stroke="darkslategray" stroke-width="8"
          stroke-dasharray="6.981 13.962" stroke-dashoffset="3.491"
          fill="none" stroke-linecap="round" />
  <circle cx="310" cy="160" r="10" stroke="lightslategray" stroke-width="20"
          stroke-dasharray="3.491 6.981" stroke-dashoffset="-3.491"
          fill="none" />

  <text x="275" y="200" font-size="8" text-anchor="middle">two concentric
circles each</text>

</svg>
```

Rectangles

Les rectangles demandent quatre attributs géométriques:

(x, y, width, height)

Les attributs ‘x’ (distance depuis la gauche) et ‘y’ (distance depuis le haut) donnent la position de l'angle supérieur gauche du rectangle, tandis que les attributs ‘width’ et ‘height’ donnent les dimensions du rectangle.

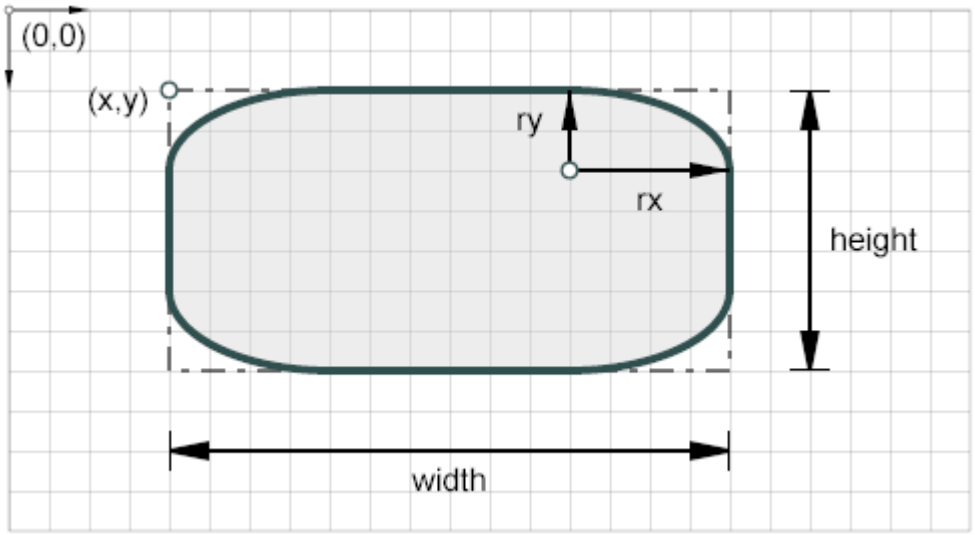

Figure 2-22. L'élément 'rectangle'

```
<svg width="350" height="300">
  <rect x="40" y="20" width="140" height="70"
        stroke="black" stroke-width="0.75" stroke-dasharray="6 3 2 3"
        fill="none" />
  <rect x="40" y="20" width="140" height="70" rx="40" ry="20"
        stroke="darkslategray" stroke-width="2"
```

```
        fill="lightgray" fill-opacity="0.4" />
</svg>
```

Voici la syntaxe de l'élément ‘rect’:

```
      <rect id="name"
          x="coordinate"
          y="coordinate"
          width="length"
          height="length"
          rx="length"
          ry="length"
        style-attribute="style-value"/>
```

La valeur par défaut pour 'x' 'y' 'width' et 'height' est 0. Si 'x' et 'y' ne sont pas spécifiés, l'angle supérieur gauche sera en (0,0).
Les attributs ‘rx’ et ‘ry’ sont optionnels, ils définissent les demi-axes de l'arc de l'ellipse qui arrondira les quatre angles du rectangle.
Voici quelques effets avec 'rx' et 'ry'.

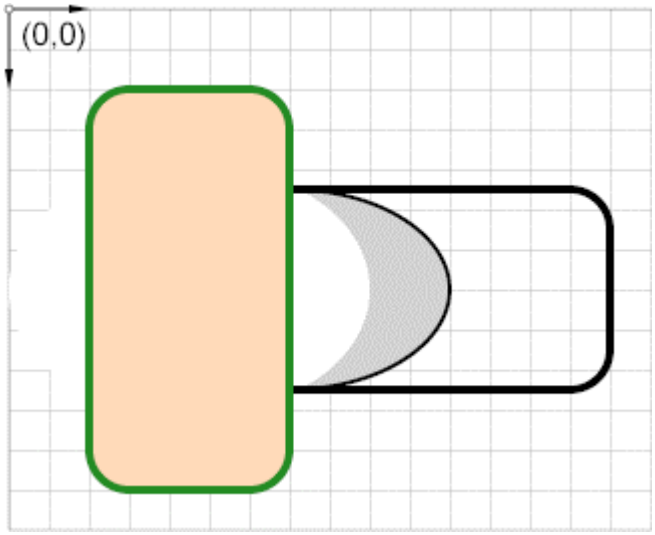

Figure 2-23. Utilisation de 'rx' et 'ry'

```
<svg width="500" height="300">

  <!-- Elliptical Rectangles -->
  <rect x="30" y="45" width="80" height="50" rx="50" ry="30"
        fill="lightgrey" stroke="black" stroke-width="1"/>
  <rect x="0" y="40" width="90" height="60" rx="40" ry="30"
        fill="white" stroke="none"/>

  <!-- Horizontal Rectangle -->
  <rect x="50" y="45" width="100" height="50" rx="10"
        fill="none" stroke="black" stroke-width="2"/>

  <!-- Verticle Rectangle -->
  <rect x="20" y="20" width="50" height="100" rx="10" ry="10"
        fill="peachpuff" stroke="forestgreen" stroke-width="2"/>
</svg>
```

Nous pouvons noter que si l'un des attributs ‘rx’ ou 'ry' seulement est spécifié, l'autre prend la même valeur et non 0. Ainsi si ‘rx’ vaut 5 unités et que ‘ry’ n'est pas spécifié, ‘rx’ et ‘ry’

seront tous deux égaux à 5 unités. Ceci s'applique au rectangle horizontal dans la figure précédente.

Ellipses

L'élément ‘ellipse’ demande quatre attributs géométriques: ‘**cx’, ‘cy’, ‘rx’ and ‘ry’.**

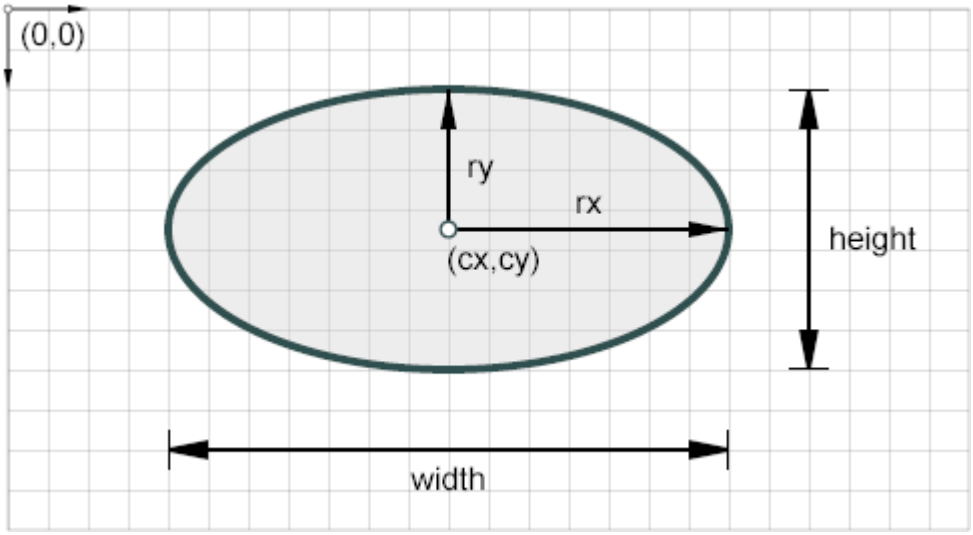

Figure 2-24. L'élément 'ellipse'

```
<svg width="350" height="300">
  <ellipse cx="110" cy="55" rx="70" ry="35"
           stroke="darkslategray" stroke-width="2"
           fill="lightgray" fill-opacity="0.4" />
</svg>
```

Les valeurs de ‘cx’ et ‘cy’ donnent la position du centre de l'ellipse tandis que les valeurs de ‘rx’ et ‘ry’ donnent les demi-axes de l'ellipse

.

Voici la syntaxe de l'élément ‘ellipse’:

```
<ellipse id="name"
        cx="coordinate"
        cy="coordinate"
        rx="length"
        rx="length"
        style-attribute="style-value"/>
```

Si vous donnez les mêmes valeurs à 'rx' et 'ry', vous obtenez un cercle comme sur cet exemple.

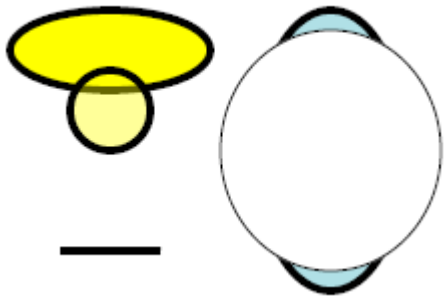

Figure 2-25. Exemples d'ellipses

```
<svg width="350" height="300">
  <!-- Horizontal Ellipse -->
  <ellipse cx="70" cy="50" rx="50" ry="20" fill="yellow" stroke="black"
```

```
          stroke-width="4"/>
  <!-- Vertical Ellipses -->
  <ellipse cx="180" cy="90" rx="35" ry="60" fill="powderblue" stroke="black"
          stroke-width="4"/>
  <ellipse cx="180" cy="110" rx="35" ry="60" fill="powderblue" stroke="black"
           stroke-width="4"/>
  <ellipse cx="180" cy="100" rx="55" ry="60" fill="white" stroke="black"/>
  <!-- Circle Ellipse -->
  <ellipse cx="70" cy="80" rx="20" ry="20" fill="yellow" fill-opacity="0.4"
          stroke="black" stroke-width="4"/>
  <!-- Odd Ellipse -->
  <ellipse cx="70" cy="150" rx="25" ry="0" fill="yellow" stroke="black"
          stroke-width="4"/>
</svg>
```

Les attributs ‘rx’ et ‘ry’ sont tous deux requis pour que l'ellipse soit dessinée. Toutefois, si l'une de ces valeurs est nulle, l'ellipse ressemble à une ligne comme ci-dessous.

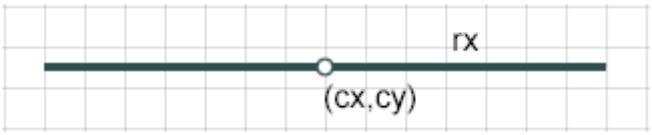

Figure 2-26. Ellipse très aplatie!

```
<ellipse cx="110" cy="55" rx="70" ry="0"
         stroke="darkslategray" stroke-width="2"
         fill="lightgray" fill-opacity="0.4" />
```

Polygones et Lignes brisées

L'attribut ‘points’ constitué de couples de valeurs définit les sommets du polygone. Voyons de plus près cet exemple.

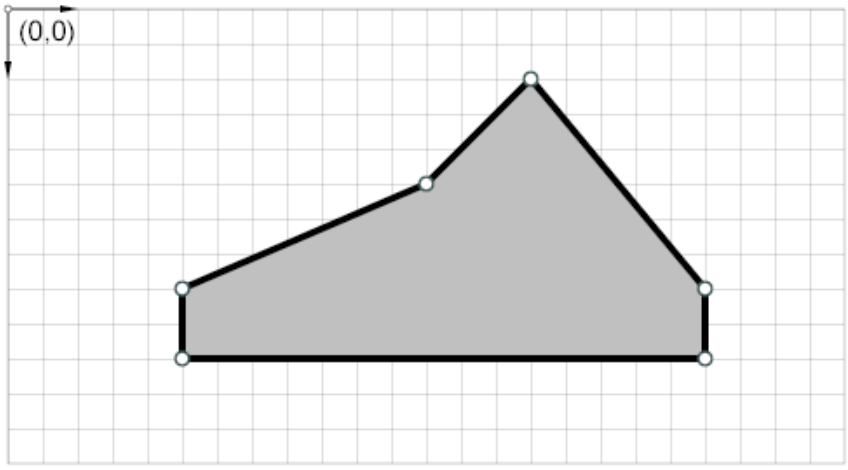

Figure 2-27. Polygone et ses sommets

```
<svg width="350" height="300">
  <polygon fill="silver" stroke="black" stroke-width="2"
        points="50,100 50,80 120,50 150,20 200,80 200,100" />
</svg>
```

Pour faciliter la lecture des sommets, ils sont ici présentés sous la forme de couples de valeurs séparées par une virgule, les couples étant eux-mêmes séparés par des espaces. Mais vous pouvez entrer votre suite de valeurs avec le même séparateur, virgule ou espace.

Soyez certain(e) d'entrer un nombre pair de coordonnées pour que tous vos sommets soient définis.

Polygones

Les polygones sont utiles pour créer très rapidement une grande variété de formes comme ici.

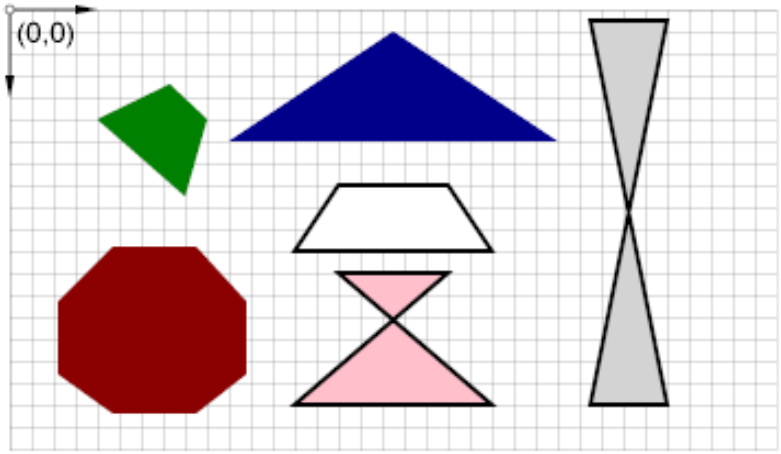

Figure 2-28. Exemples de polygones

```
<svg width="350" height="250">
  <!-- Triangle -->
  <polygon points="175,10 100,60 250,60" fill="darkblue"/>
  <!-- Plane -->
  <polygon points="73,34 90,50 80,85 40,50" fill="green" stroke-width="1"/>
  <!-- Trapezoid -->
  <polygon points="150,80 200,80 220,110, 130,110" fill="white" stroke="black"
           stroke-width="2"/>
  <!-- Octagon -->
  <polygon points="47,108 22,133 22,166 47,184 85,184 108,166 108,133 85,108"
           fill="darkred" stroke-width="1"/>
  <!-- ART -->
  <polygon points="150,120 200,120 130,180, 220,180" fill="pink" stroke="black"
           stroke-width="2"/>
  <polygon points="265,180 300,5 265,5 300,180"
           stroke="black" stroke-width="2" fill="lightgrey" />
</svg>
```

La syntaxe de l'élément ‘polygon’ est très simple:

```
<polygon id="name"
        points="coordinates"
        style-attribute="style-value"/>
```

Exemples de polygones

Voici quelques figures créées avec un ou deux triangles et en jouant sur les attributs de présentation.

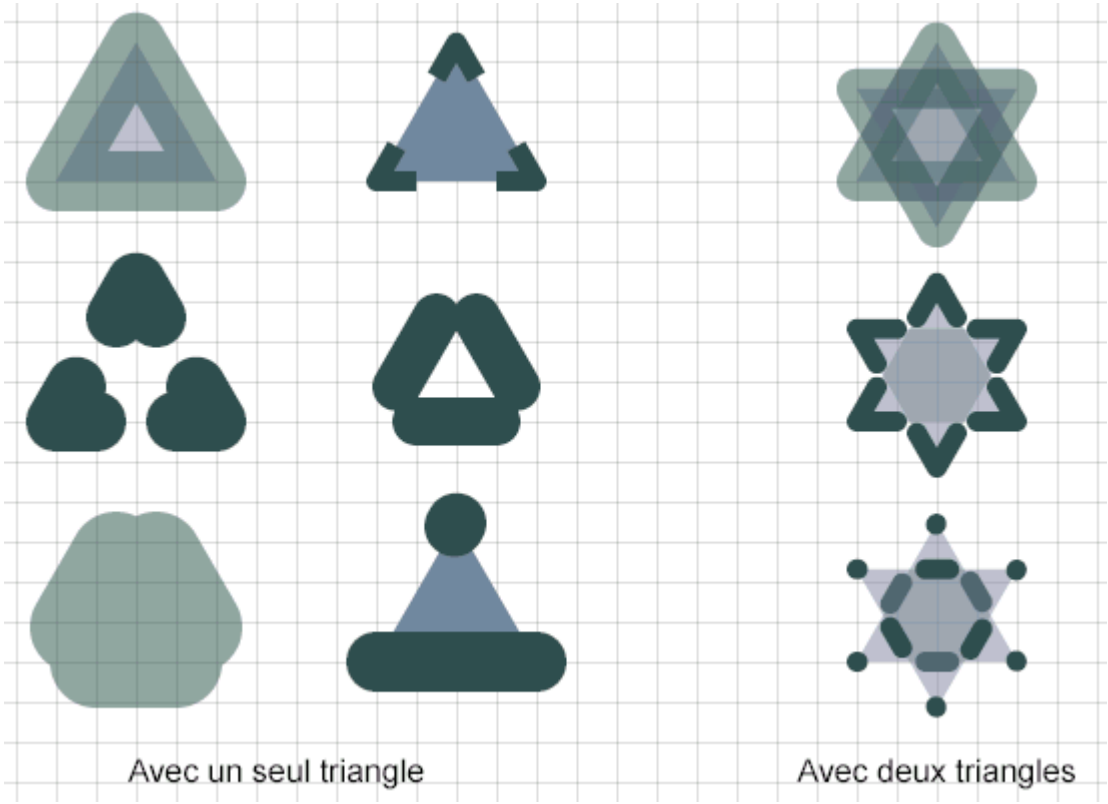

Figure 2-29. Formes étonnantes avec un ou deux triangles

```
<svg width="800" height="600" viewBox="0 0 400 300">
  <defs>
    <pattern id="gridPattern" x="0" y="0" width="10" height="10"
             patternUnits="userSpaceOnUse">
      <path d="M 10 0 L 0 0 0 10" fill="none" stroke="gray"
            stroke-width="0.25"/>
    </pattern>
  </defs>
  <rect id="grid" width="300" height="211" stroke="gray" stroke-width="0.1"
        fill="url(#gridPattern)" /> <!-- grid -->

  <polygon points="20,50 60,50 40,15.36" stroke="darkslategray"
           stroke-width="15" stroke-opacity="0.5" fill="lightslategray"
           fill-opacity="0.5" stroke-linejoin="round" />

  <polygon points="100,50 140,50 120,15.36" stroke="darkslategray"
           stroke-width="5" stroke-dasharray="20 20" stroke-dashoffset="10"
           fill="lightslategray" stroke-linejoin="round" />

  <polygon points="20,110 60,110 40,75.36" stroke="darkslategray"
           stroke-width="15" stroke-dasharray="20 20" stroke-dashoffset="10"
           fill="none" stroke-linejoin="round" stroke-linecap="round" />

  <polygon points="100,110 140,110 120,75.36" stroke="darkslategray"
           stroke-width="12" stroke-dasharray="20 20" stroke-dashoffset="-10"
           fill="none" stroke-linecap="round" />

  <polygon points="20,170 60,170 40,135.36" stroke="darkslategray"
           stroke-width="23" stroke-opacity="0.5" stroke-dasharray="20 20"
           stroke-dashoffset="-10" fill="none" stroke-linecap="round" />
```

```
  <polygon points="100,170 140,170 120,135.36" stroke="darkslategray"
           stroke-width="15" stroke-dasharray="40 40 1 40"
           fill="lightslategray" stroke-linecap="round" />

  <text x="75" y="200" font-size="8" text-anchor="middle">single triangle
each</text>

  <polygon points="220,50 260,50 240,15.36" stroke="darkslategray"
           stroke-width="10" stroke-opacity="0.5" fill="lightslategray"
           fill-opacity="0.5" stroke-linejoin="round" />

  <polygon points="220,26.9 260,26.9 240,61.54" stroke="darkslategray"
           stroke-width="10" stroke-opacity="0.5" fill="lightslategray"
           fill-opacity="0.5" stroke-linejoin="round" />

  <polygon points="220,110 260,110 240,75.36" stroke="darkslategray"
           stroke-width="5" stroke-dasharray="20 20" stroke-dashoffset="10"
           fill="lightslategray" fill-opacity="0.5" stroke-linecap="round"
           stroke-linejoin="round" />
  <polygon points="220,86.9 260,86.9 240,121.54" stroke="darkslategray"
           stroke-width="5" stroke-dasharray="20 20" stroke-dashoffset="10"
           fill="lightslategray" fill-opacity="0.5" stroke-linecap="round"
           stroke-linejoin="round" />

  <polygon points="220,170 260,170 240,135.36" stroke="darkslategray"
           stroke-width="5" stroke-dasharray="0.1 17 6 17"
           stroke-dashoffset="0" fill="lightslategray" fill-opacity="0.5"
           stroke-linecap="round" stroke-linejoin="round" />
  <polygon points="220,146.9 260,146.9 240,181.54" stroke="darkslategray"
           stroke-width="5" stroke-dasharray="0.1 17 6 17"
           stroke-dashoffset="0" fill="lightslategray" fill-opacity="0.5"
           stroke-linecap="round" stroke-linejoin="round" />

  <text x="240" y="200" font-size="8" text-anchor="middle">two triangles
each</text>

</svg>
```

Lignes brisées

Voyez le code pour décrire ces formes.

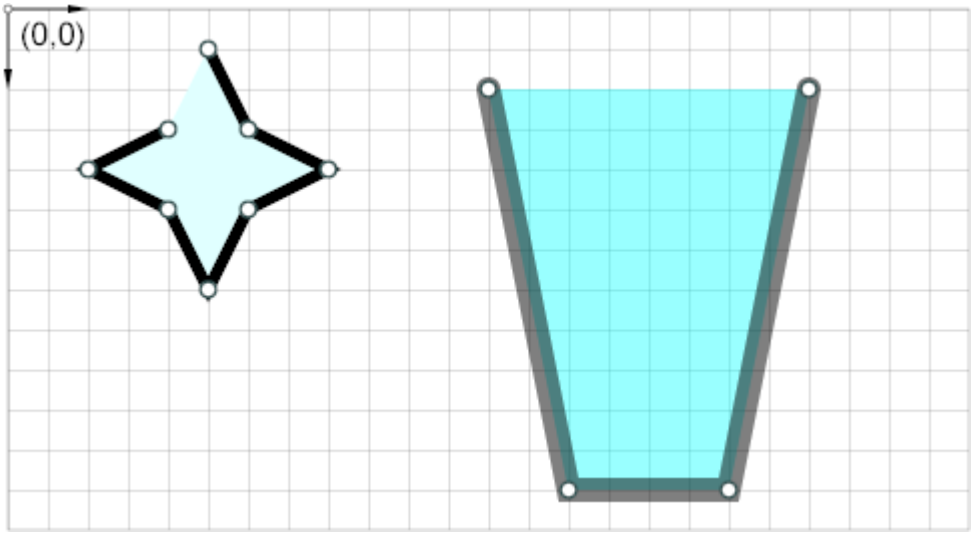

Figure 2-30. Exemples de lignes brisées

```
<svg width="350" height="250">

  <!-- Star Polyline -->
  <polyline points="50,10 60,30 80,40 60,50 50,70 40,50 20,40 40,30"
            fill="lightcyan" stroke="black" stroke-width="3"/>

  <!-- Line Polyline -->
  <polyline points="120,20 140,120 180,120 200,20"
            fill="cyan" fill-opacity="0.4" stroke="black"
            stroke-width="6" stroke-linecap="round" stroke-opacity="0.5" />
</svg>
```

Comme l'élément ‘polygon’, l'élément ‘polyline’ utilise l'attribut “points” pour spécifier les sommets. Remarquez que la ligne n'est pas fermée automatiquement comme pour l'élément 'polygon'. Pour fermer la ligne, vous devez ajouter le point de départ de votre tracé à la fin des données.

La syntaxe de l'élément ‘polyline’ est exactement la même que celle de ‘polygon’:

```
<polyline id="name"
          points="coordinates"
          style-attribute="style-value"/>
```

Qu'arrive-t-il si l'élément ‘polygon’ contient un seul point ou même deux points identiques? Le polygone n'est pas dessiné, même pas sous la forme d'un point.

Même avec cet exemple simple, nous commençons à voir que le codage dans un traitement de texte devient vite inefficace. Imaginez combien il faut de points pour rendre un simple diagramme statistique ou une carte géographique. En fait, SVG est rarement codé ainsi. De nombreux outils apparaissent qui génèrent la plupart des graphiques SVG. Mais pour mieux comprendre SVG, il n'est pas inutile de prendre ces exemples, de les modifier dans un éditeur et de voir.

Un petit concours

Les attributs de présentation nous apportent un supplément de créativité comme nous venons de le voir dans les exemples de ‘line’, ‘circle’, 'ellipse' et ‘polygon’.

Le concours

Créez un dessin original en utilisant les attributs de présentation et au plus trois formes de base. Envoyez votre dessin à *dev@learnsvg.com* et nous mettrons les meilleurs sur notre site. Vous pourrez même peut-être gagner le tee-shirt **Learn SVG**!

Tableau de référence pour les formes de base

Voici un tableau donnant les attributs géométriques pour chacune des formes de base.

Forme	**Elément**	**Attributs obligatoires**	**Attributs optionnels**
Ligne	<line>	(none)	x1, y1, x2, y2
Rectangle	<rect>	width, height	x, y, rx, ry
Cercle	<circle>	r	cx, cy
Ellipse	<ellipse>	rx, ry	cx, cy
Ligne brisée	<polyline>	points	
Polygone	<polygon>	points	

D'autres formes de base?

Quelques autres formes tels qu’arc (ouvert, fermé, secteur), spirale, étoile et polygones réguliers pourraient être incluses dans de prochaines versions de SVG.

Il est facile de les décrire, elles sont toutes redimensionnables, soumises à un script ou une animation!

Note: Les formes ne sont que des éléments 'path' prédéfinis. SVG a un élément ‘path’ qui permet de rendre n'importe quelle forme, même la plus complexe. Les formes et les éléments 'path' sont mathématiquement équivalents et si le groupe de travail SVG a inclus six de ces formes, ce n'est que pour faciliter notre compréhension et leur utilisation. Nous étudierons l'élément 'path' en détail dans le chapitre 4.

L'élément 'image'

Image bitmap

SVG est un type unique de graphique qui peut contenir des images de format varié, comme d'autres documents SVG aussi bien que des images bitmap au format JPEG ou PNG. Les images PNG et JPEG peuvent être transformées, animées, recevoir des filtres, contrôlées dynamiquement et interactives à travers un script.

Les attributs de l'élément 'image' sont:
x, y, width, height, xlink:href

Les attributs 'x' et 'y' définissent le coin supérieur gauche de l'image. Les attributs 'width' et 'height' donnent les dimensions de l'image et l'attribut 'xlink:href' donne l'adresse du fichier source de l'image.

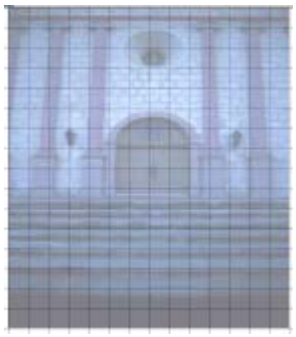

Figure 2-31. Image bitmap dans un graphique SVG

```
<svg width="350" height="250">
  <image xlink:href="3202948.jpg" x="340" y="0" width="140" height="160"
         opacity="0.5"/>
</svg>
```

Comparons avec HTML

En HTML nous utilisons la balise 'img' avec l'attribut 'src'. SVG fait de même. Pour inclure une image en SVG, nous utilisons l'élément 'image'. Comme en HTML, les attributs 'width' et 'height' peuvent être utilisés pour imposer des dimensions à l'image JPEG ou PNG. L'attribut qui permet de définir le fichier source de l'image est 'xlink:href'. Le "xlink:" fait référence à un nom d'espace XML qui sera discuté en détail au chapitre 11. Pour le moment, nous considérons simplement que 'xlink:href' joue le même rôle que l'attribut 'src' en HTML.

Types d'image supportés

Les possibilités en SVG de positionner précisément ou de transformer (faire tourner, déformer, appliquer un filtre ..) les images en font un outil idéal pour l'affichage d'images bitmap. Actuellement, PNG et JPEG sont les seuls formats que les visualiseurs doivent supporter selon les spécifications.

Les développeurs Web savent que les formats JPEG et PNG sont les plus intéressants pour afficher des graphiques détaillés que SVG aurait beaucoup de difficulté à rendre mathématiquement.

Le format JPEG est le plus intéressant pour des photographies de bonne qualité.

Note au lecteur

Les visualiseurs ne supportent pas nécessairement le format GIF. Ce n'est pas une difficulté, le format PNG est un excellent substitut (sauf pour les GIF animés) et peut être généré et exporté dans la plupart des outils graphiques. Notons que le plugin d'Adobe, dans sa version 3, supporte le format GIF.

Adresses absolues et relatives

Selon l'environnement, les deux méthodes présentent des avantages et des inconvénients. Tous les navigateurs utilisant un plugin ne traitent pas les adresses absolues et relatives de la même façon.

Des adresses relatives permettent une maintenance du site plus facile. Quand un graphique inséré dans une balise 'image' n'est pas dessiné, la première chose à vérifier est l'adresse du fichier source.

Un exemple: j'ai créé plusieurs répertoires à la racine de mon site Web et j'ai l'arborescence suivante:

\root
- \images
- \script
- \stylesheets
- \svg

Les différents type de fichiers utilisés pour concevoir mes pages Web sont dans des répertoires distincts.

Le fichier SVG est donc localisé dans le répertoire \svg et il fait référence à une image dont le fichier source logo.jpg est dans le répertoire \images. Nous devons donc spécifier comme valeur pour l'attribut 'xlink:href' (ou 'src' en HTML) “../images/logo.jpg”. Comme en HTML les deux points signifient que nous remontons d'un niveau dans l'arborescence avant de redescendre dans le répertoire \images où est stocké notre fichier.

```
<image xlink:href="../images/logo.jpg">
```

Analyse d'un graphique SVG

Le temps de la création! Nous allons reprendre l'image vue au chapitre 1 et voir en détail ses composants.

Maintenant que nous connaissons mieux la structure de SVG et quelques éléments, nous allons explorer ce document et comprendre comment sont formés ses composants.

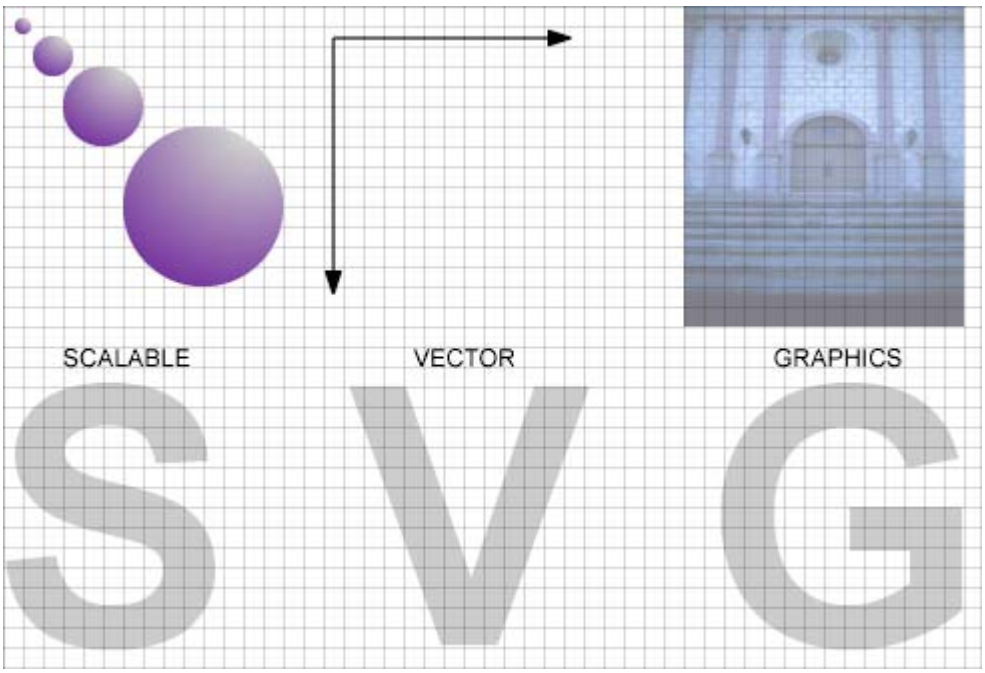

Figure 2-32. Illustration des capacités de SVG

Le code SVG complet

Voici le code complet de ce graphique.

```
<svg width="100%" height="100%">
  <title>Scalable Vector Graphics - Introduction</title>
  <desc>This graphic demonstrates many exciting features of SVG.</desc>
  <defs>
    <circle id="Ball01" cx="100" cy="100" r="40" fill="url(#Grad01)"/>
    <text x="50%" y="80%" text-anchor="middle" writing-mode="lr">S V G</text>
    <pattern id="Pat01" width="10" height="10" patternUnits="userSpaceOnUse">
      <rect width="10" height="10" fill="#FFFFFF" stroke="#000000"
            stroke-width="0.1"/>
    </pattern>
    <radialGradient id="Grad01" gradientUnits="userSpaceOnUse" cx="120" cy="60"
                    fx="130" fy="50" r="100">
      <stop offset="0%" stop-color="white"/>
      <stop offset="20%" stop-color="#cccccc"/>
      <stop offset="100%" stop-color="rgb(100,20,150)"/>
    </radialGradient>
    <marker id="Marker01" viewBox="0 0 10 10" refX="0" refY="5"
            markerUnits="strokeWidth" markerWidth="10" markerHeight="6"
            orient="auto">
      <path d="M 0 0 L 15 5 L 0 10 z" />
    </marker>
  </defs>
  <g letter-spacing="0.02em" enable-background="new">

    <!--FOND-->
    <rect id="Background" fill="url(#Pat01)" stroke-width="0.5"
          stroke="#000000" x="0" y="0" width="100%" height="100%">
    <desc>Background Pattern</desc>
  </rect>
  <g id="watermark" fill="#000000" font-weight="bold" font-size=”180"
     font-family="san-serif,Arial,Verdana" opacity="0.2">
```

```
      <text x="50%" y="70%" text-anchor="middle" stroke="none"
            writing-mode="lr">S V G</text>
  </g>

    <!--SCALABLE-->
    <text x="30" y="180" stroke="none" writing-mode="lr">SCALABLE</text>
    <use xlink:href="#Ball01" x="0" y="0" transform="scale(0.1)"/>
    <use xlink:href="#Ball01" x="0" y="0" transform="scale(0.25)"/>
    <use xlink:href="#Ball01" x="0" y="0" transform="scale(0.5)"/>
    <use xlink:href="#Ball01" x="0" y="0" transform="scale(1)"/>
    <!--VECTOR-->
    <text x="205" y="180" stroke="none" writing-mode="lr">VECTOR</text>
    <line x1="165" y1="16" x2="165" y2="132" stroke="black" stroke-width="1.5"
          marker-end="url(#Marker01)"/>
    <line x1="165" y1="16" x2="272" y2="16" stroke="black" stroke-width="1.5"
          marker-end="url(#Marker01)"/>

    <!--GRAPHICS-->
    <text x="385" y="180" stroke="none" writing-mode="lr">GRAPHICS</text>
    <image xlink:href="3202948.jpg" x="340" y="0" width="140" height="160"
           opacity="0.5"/>
  </g>
</svg>
```

Et c'est tout!

Si vous regardez rapidement ce code, vous reconnaissez quelques mots comme circle, text, line, image, style, stroke, fill et opacity. Il y en a d'autres tels que pattern, gradient, marker, path, use, xlink:href, cx, cy, stop, patternUnits et gradientUnits. Tous ces termes viendront en temps utile dans ce livre avec des exemples, leur syntaxe et des utilisations plus avancées.

Voyons en détail quelques parties de ce code.

Les lignes fléchées

Ces lignes symbolisent l'aspect vectoriel de SVG.

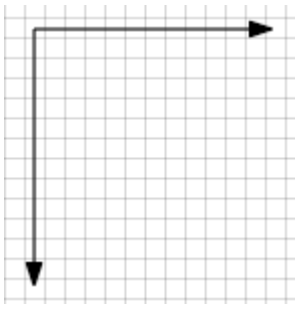

Figure 2-33. Lignes avec 'marker'

```
<!--VECTOR-->
<line x1="165" y1="16" x2="165" y2="132"
      stroke="black" stroke-width="1.5"
      marker-end="url(#Marker01)"/>
<line x1="165" y1="16" x2="272" y2="16"
      stroke="black" stroke-width="1.5"
      marker-end="url(#Marker01)"/>
```

L'attribut ‘marker-end’ a pour valeur “url(#Marker01)”. Il fait référence à un élément 'marker' qui est défini dans la section 'defs', cette section permet de définir des objets graphiques qui ne seront pas dessinés directement mais utilisés par d'autres éléments du graphique.

```
<defs>
  <marker id="Marker01" viewBox="0 0 10 10" refX="0" refY="5"
        markerUnits="strokeWidth" markerWidth="10" markerHeight="6"
        orient="auto">
        <path d="M 0 0 L 15 5 L 0 10 z" />
  </marker>
</defs>
```

Une flèche sera dessinée à la fin de chaque élément 'line' ou 'path' qui l'utilisera.

Le rectangle utilisé comme fond

Ce rectangle définit le fond de l'image, un fond quadrillé.

```
<!--FOND-->
<rect id="Background" x="0" y="0" width="100%" height="100%"
      fill="url(#Pat01)" stroke-width="0.5" stroke="#000000" >
   <desc>Background Pattern</desc>
</rect>
```

Remarquez que la propriété ‘fill’ a pour valeur “url(#Pat01)”. C'est une référence à un élément ‘pattern’ lui-aussi décrit dans la section 'defs'.

```
<pattern id="Pat01" width="10" height="10" patternUnits="userSpaceOnUse">
  <rect width="10" height="10"
       fill="#FFFFFF" stroke="#000000" stroke-width="0.1"/>
</pattern>
```

Les éléments 'pattern' seront abordés dans le prochain chapitre et leur étude approfondie dans le chapitre 7.

Les cercles illustrant le redimensionnement

Nous utilisons l'élément ‘use’ pour afficher des cercles de différentes tailles.

```
<!--SCALABLE-->
<use xlink:href="#Ball01" x="0" y="0" transform="scale(0.1)"/>
<use xlink:href="#Ball01" x="0" y="0" transform="scale(0.25)"/>
<use xlink:href="#Ball01" x="0" y="0" transform="scale(0.5)"/>
<use xlink:href="#Ball01" x="0" y="0" transform="scale(1)"/>
```

Une fois encore, nous faisons référence à un élément 'circle' défini dans la section 'defs'. Comme l'élément 'use' accepte 'xlink:href' comme attribut nous l'utilisons alors que pour les objets précédents nous avons utilisé ‘url()’ comme valeur pour 'fill' et 'marker-end'.

```
<defs>
  <circle id="Ball01" cx="100" cy="100" r="40" fill="url(#Grad01)"/>
</defs>
```

Les transformations seront vues en détail dans le chapitre 6.

L'élément 'Text'

Nous utilisons l'élément 'text' pour afficher la phrase “SCALABLE VECTOR GRAPHICS”.

```
<!--SCALABLE-->
<text
      x="30"
      y="180"
      stroke="none"
      writing-mode="lr">SCALABLE</text>
<!--VECTOR-->
<text
      x="205"
      y="180"
      stroke="none"
      writing-mode="lr">VECTOR</text>
<!--GRAPHICS-->
<text
      x="385"
      y="180"
      stroke="none"
      writing-mode="lr">GRAPHICS</text>
```

Tous les attributs de cet élément 'text' seront vus au chapitre 5.

Sommaire

Dans ce chapitre nous avons vu quelques concepts et termes avec les formes de base et les images bitmap.

Dans le prochain chapitre nous allons examiner avec attention la structure d'un document SVG.