Chapitre 5 Afficher du texte

"When ideas fail, words come in very handy."
- Goethe (1749-1832)

"Only two things are infinite, the universe and human stupidity, and I'm not sure about the former."
- Albert Einstein (1879-1955)

Objectifs du chapitre

- Les éléments <text> et <tspan>
- Formater et sélectionner du texte
- Attributs de présentation du texte
- Attributs de disposition du texte
- Fontes, Glyphes et fontes SVG
- L'élément <tref>
- Texte sur une courbe avec l'élément <textPath>
- Internationalisation
- Gérer les espaces

Aperçu

SVG nous permet avec l'objet 'text' d'afficher des titres, des légendes, des boutons et même des paragraphes complets de texte. Travailler avec le texte n'est pas toujours évident, mais à la fin de ce chapitre vous devriez avoir de bonnes connaissances vous permettant l'affichage de texte dans votre document SVG. Nous verrons aussi bien les aspects esthétiques que la programmation.

Beaucoup de choses sont spécifiques au texte, nous allons les découvrir et les mettre en pratique dans ce chapitre. Nous approfondirons avec le texte sur une courbe, comment faire référence à un contenu texte interne ou externe, les fontes, le formatage du texte et également nous anticiperons sur les spécifications SVG 2.0.

Les éléments <text> et <tspan>

Les éléments ‘text’ et ‘tspan' sont à la base de l'affichage de texte en SVG.

L'élément <text>

L'élément 'text' nécessite la donnée des attributs “x” et “y” qui définissent le point de départ de l'affichage de notre texte. Voici l'exemple classique “Hello world” en SVG.

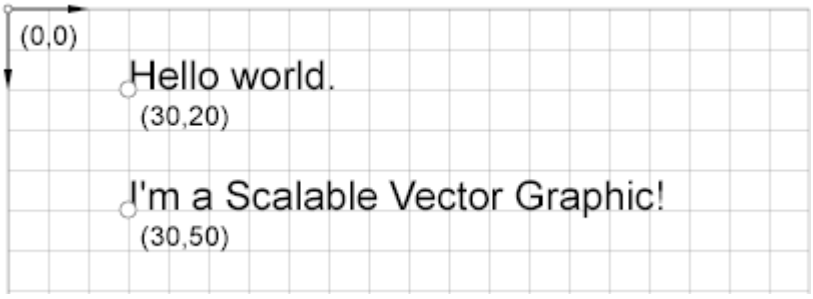

Figure 5-1. Hello world

```
<svg width="350" height="300">
      <text x="30" y="20"
            fill="black" font-size="10">
            Hello world.
      </text>
      <text x="30" y="50"
            fill="black" font-size="10">
            I&apos;m a Scalable Vector Graphic!
      </text>
</svg>
```

L'exemple “Hello world” utilise les attributs ‘x’ et ‘y’ pour positionner le texte dans le système de coordonnées SVG. Les attributs ‘dx’ et ‘dy’ peuvent être utilisés pour décaler le texte, ils sont optionnels.

Voyons la syntaxe de cet élément ‘text’.

Syntaxe

```
<text id="name"
        x="coordinate+"
        y="coordinate+"
        dx="lengths+"
        dy="lengths+"
        rotate="numbers+"
        textLength="length"
        lengthAdjust"spacing|spacingAndGlyphs"
        transform="transform properties"
        style-attribute="style-attribute">
        <!-- texte à afficher -->
</text>
```

Ce simple diagramme illustre les propriétés et la terminologie pour disposer le texte.

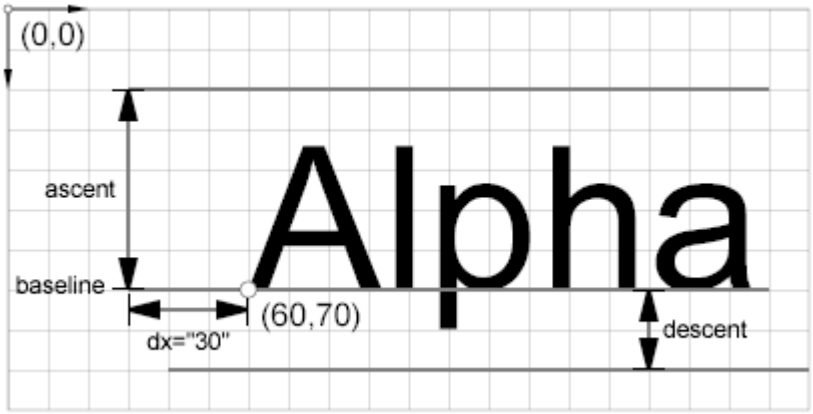

Figure 5-2. Disposition du texte

```
<?xml version="1.0"?>
<svg width="350" height="300">
      <text x="30" y="70" dx="30" font-size="50">Alpha</text>
</svg>
```

Sur ce schéma, ‘x’ est égal à ‘30’ et ‘y’ à ‘70’. Puis ‘dx’ décale le texte de ‘30’ unités ainsi le texte "Alpha" commence au point (60,70).

Si vous ne donnez pas de valeur pour ‘x’ ou ‘y’, ce seront les valeurs par défaut, 0 qui seront prises en compte pour positionner le texte. Si vous ne spécifiez pas de valeur pour 'viewBox', l'angle supérieur gauche de votre dessin étant (0,0), vous ne verrez certainement pas votre texte à l'écran.

L'élément <tspan>

L'élément ‘tspan’ ressemble à l'élément ‘text’, mais il doit être descendant d'un élément 'text'.

Comme nous le voyons sur cet exemple, des valeurs multiples peuvent être indiquées pour positionner chaque caractère du texte à afficher.

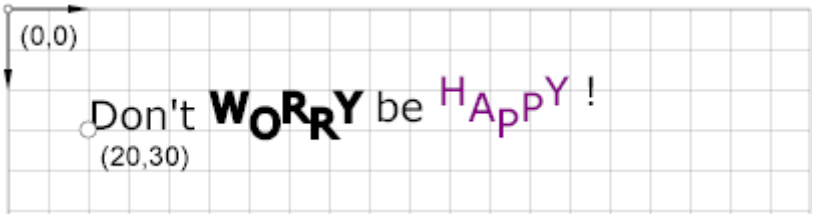

Figure 5-3. Positionnement lettre par lettre

```
<svg width="350" height="300">
      <text x="20" y="30"
            fill="black">
            Don't
            <tspan dx="0" dy="-2 4 -4 4 -4" stroke="black">WORRY</tspan>
            be
            <tspan dx="0" dy="-4 4 4 -4 -4" fill="purple">HAPPY</tspan>
            !
      </text>
</svg>
```

Quelques remarques sur cet exemple, les attributs ‘dx’ et ‘dy’ de l'élément ‘tspan’ le décale par rapport à la position générale de l'élément 'text' parent. Ici, ‘dy’ a pour valeur une liste de nombres qui affectent individuellement chaque lettre des mots ‘WORRY’ et ‘HAPPY’.

Voyons la syntaxe de cet élément 'tspan'

Syntaxe

```
<tspan id="name"
        x="coordinate+"
        y="coordinate+"
        dx="lengths+"
        dy="lengths+"
        rotate="numbers+"
        textLength="length"
        style-attribute="style-attribute">
        <!-- texte à afficher -->
</tspan>
```

Cette syntaxe de l'élément ‘tspan’ est semblable à celle de l'élément ‘text’. Le symbole ‘+’ qui suit le type de données à affecter aux attributs signifie que l'on peut affecter une liste de valeurs et non une seule.

Formatage et sélection de texte

Pour l'instant, l'élément ‘text’ ne formate pas automatiquement sur plusieurs lignes le texte dans les visualiseurs. Si votre texte dépasse les limites de votre zone d'affichage, il n'est pas rendu.

Les prochaines spécification SVG 2.0 devraient prendre en compte le formatage du texte. Pour le moment, nous devons disposer notre texte sur plusieurs lignes par le codage en utilisant des éléments ‘tspan’ ou ‘text’.

L'utilisateur peut sélectionner le texte affiché, mais il ne peut le faire que dans un seul élément 'text' à la fois. Il est donc préférable de créer nos lignes avec l'élément 'tspan' pour que la sélection de plusieurs lignes de texte soit possible.

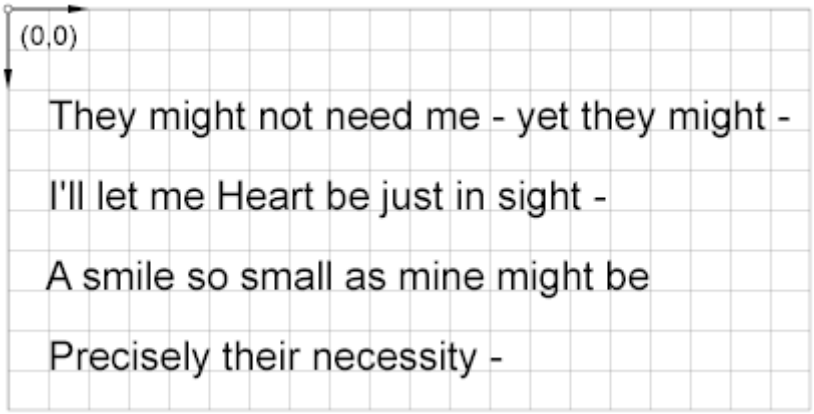

Figure 5-4. Plusieurs lignes de texte

```
<?xml version="1.0"?>
<svg width="350" height="300">
  <text x="10" y="30" fill="black" font-size="10">
    <tspan>They might not need me - yet they might -</tspan>
    <tspan x="10" dy="20">I'll let me Heart be just in sight -</tspan>
    <tspan x="10" dy="20">A smile so small as mine might be </tspan>
    <tspan x="10" dy="20">Precisely their necessity -</tspan>
  </text>
</svg>
```

Comme le poème de la figure 5-4 est dans un seul élément ‘text’, nous pouvons sélectionner, copier et coller toutes les lignes à la fois. Notons que ce poème est de Emily Dickinson.

Nous pouvons également utiliser ECMAScript pour un formatage automatique du texte au chargement du fichier par exemple. Vous pouvez voir cette application sur le site pilat.free.fr.

Propriétés de présentation

SVG permet de nombreuses possibilités pour contrôler le rendu du texte. Des attributs de présentation comme 'fill', 'stroke', 'underline', 'bold', 'italic', 'subscript' et 'superscript'.

Propriétés 'Fill' et 'Stroke'

Ces propriétés sont les mêmes que pour les autres objets graphiques.

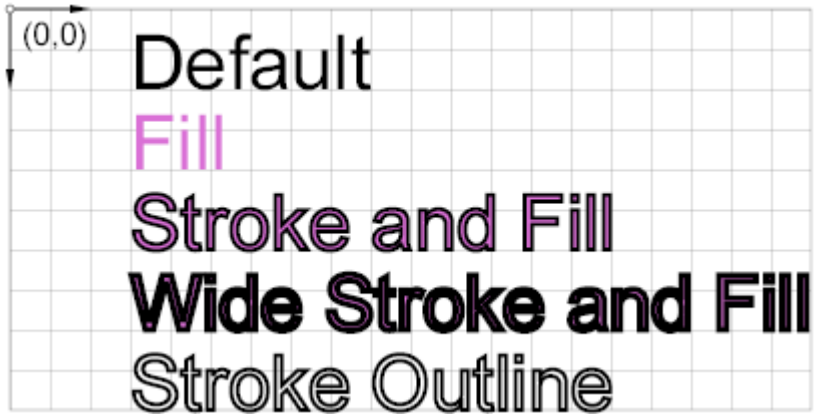

Figure 5-5. Attributs de présentation

```
<svg width="350" height="300">
      <text x="30" y="20"
            font-size="19">
            Default
      </text>
      <text x="30" y="40"
            fill="orchid" font-size="19">
            Fill
      </text>
      <text x="30" y="60"
            fill="orchid" stroke="black" font-size="19">
            Stroke and Fill
      </text>
      <text x="30" y="80"
            fill="orchid" stroke="black" stroke-width="1.5" font-size="19">
            Wide Stroke and Fill
      </text>
      <text x="30" y="100"
            fill="none" stroke="black" font-size="19">
            Stroke Outline
      </text>
</svg>
```

Nous vous encourageons à essayer ces différentes combinaisons. Il est très facile de personnaliser votre texte, ce ne sont que quelques propriétés du standard CSS appliquées aux éléments <text>. Mais nous pouvons faire beaucoup plus comme vous allez le voir.

Les attributs principaux sont 'font-weight', 'font-style' et 'text-decoration'. Les propriétés 'font-weight' et 'font-style' viennent des spécifications CSS à http://www.w3.org/TR/REC-CSS2/fonts.html.

'font-weight'

Valeur:	normal | bold | bolder | lighter | 100 | 200 | 300| 400 | 500 | 600 | 700 | 800 | 900 | inherit
Par défaut:	normal
S'applique à:	éléments 'text'
Transmissible:	Oui
Pourcentages:	N/A
Media:	Visuel
Animable:	Oui

'font-style'

Valeur:	normal | italic | oblique | inherit
Par défaut:	normal
S'applique à:	éléments 'text'
Transmissible:	oui
Pourcentages:	N/A
Media:	visuel
Animable:	oui

'text-decoration'

Valeur:	none | [underline || overline || line-through || blink] | inherit
Par défaut:	none
S'applique à:	éléments 'text'
Transmissible:	non
Pourcentages:	N/A
Media:	visuel
Animable:	oui

La propriété 'text-decoration' peut prendre les valeurs ‘underline’, ‘overline’, ‘line-through’ ou ‘blink’.

Appliquer 'bold', 'underlined' ou 'italic' en SVG peut être réalisé par les mêmes propriétés qu'en HTML. Toutefois, SVG permet l'utilisation d'attributs de présentation comme le montre cet exemple de la figure 5-6.

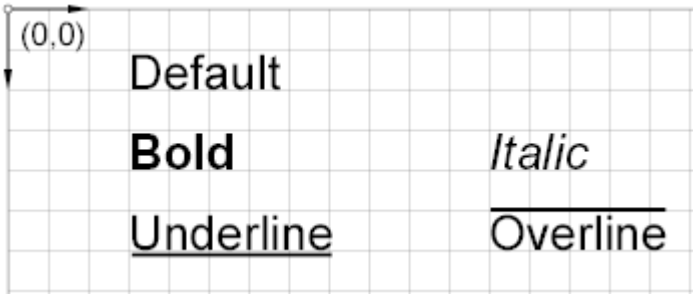

Figure 5-6. Attributs de présentation

```
<svg width="350" height="300">
   <text x="30" y="20"
             font-size="12">
             Default
       </text>
       <text x="30" y="40"
             font-weight="bold" font-size="12">
             Bold
       </text>
       <text x="120" y="40"
             font-style="italic" font-size="12">
             Italic
       </text>
       <text x="30" y="60"
             text-decoration="underline" font-size="12">
             Underline
       </text>
       <text x="120" y="60"
             text-decoration="overline" font-size="12">
```

```
            Overline
      </text>
</svg>
```

Comme vous le constatez, il est très facile d'ajouter des attributs de présentation au texte.

'text-rendering'

Cette propriété 'text-rendering' peut affecter très sérieusement la qualité du rendu de votre texte comme le montre cet exemple qui nous donne les différentes valeurs que peut prendre la propriété.

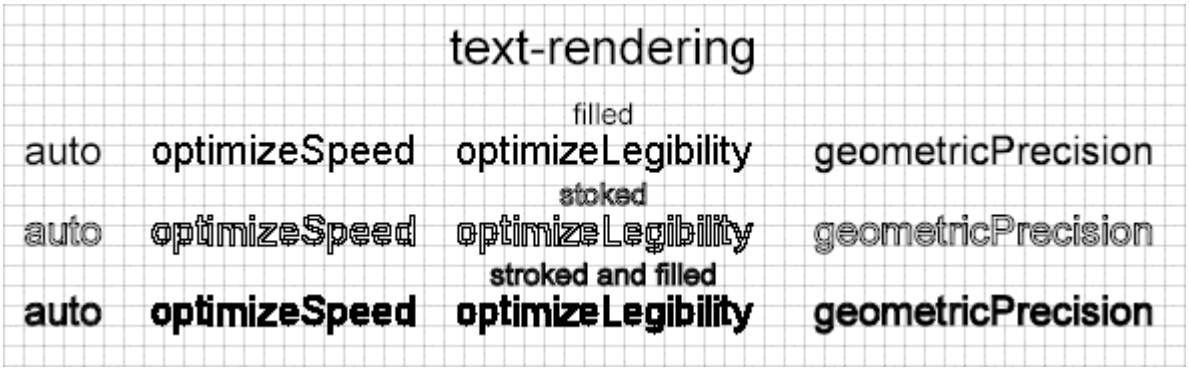

Figure 5-7. L'attribut 'text-rendering'

```
<svg width="600" height="200" viewBox="0 220 600 200">
<text    text-rendering="auto"    x="300"    y="250"    style="font-size:25;text-
anchor:middle">text-rendering</text>
<g transform="translate(0,-70)">
  <text text-rendering="auto" x="300" y="350" style="font-size:15;text-
anchor:middle">filled</text>
  <text text-rendering="auto" x="30" y="370" style="font-size:20;text-
anchor:middle">auto</text>
  <text text-rendering="optimizeSpeed" x="140" y="370" style="font-
size:20;text-anchor:middle">optimizeSpeed</text>
  <text text-rendering="optimizeLegibility" x="300" y="370" style="font-
size:20;text-anchor:middle">optimizeLegibility</text>
  <text text-rendering="geometricPrecision" x="490" y="370" style="font-
size:20;text-anchor:middle">geometricPrecision</text>
</g>
<g transform="translate(0,10)" stroke="black">
  <text text-rendering="auto" x="300" y="350" style="font-size:15;text-
anchor:middle">stroked and filled</text>
  <text text-rendering="auto" x="30" y="370" style="font-size:20;text-
anchor:middle">auto</text>
  <text text-rendering="optimizeSpeed" x="140" y="370" style="font-
size:20;text-anchor:middle">optimizeSpeed</text>
  <text text-rendering="optimizeLegibility" x="300" y="370" style="font-
size:20;text-anchor:middle">optimizeLegibility</text>
  <text text-rendering="geometricPrecision" x="490" y="370" style="font-
size:20;text-anchor:middle">geometricPrecision</text>
</g>
<g transform="translate(0,-30)" stroke="black" fill="none">
  <text text-rendering="auto" x="300" y="350" style="font-size:15;text-
anchor:middle">stoked</text>
  <text text-rendering="auto" x="30" y="370" style="font-size:20;text-
anchor:middle">auto</text>
  <text text-rendering="optimizeSpeed" x="140" y="370" style="font-
size:20;text-anchor:middle">optimizeSpeed</text>
  <text text-rendering="optimizeLegibility" x="300" y="370" style="font-
size:20;text-anchor:middle">optimizeLegibility</text>
  <text text-rendering="geometricPrecision" x="490" y="370" style="font-
size:20;text-anchor:middle">geometricPrecision</text>
</g>
```

```
</svg>
```

Disposition du texte et alignement

Nous avons à notre disposition de nombreuses propriétés spécifiques qui permettent le positionnement du texte dans notre document SVG.

Propriétés 'text-anchor' et 'baseline-shift'

La propriété ‘text-anchor’ permet de centrer le texte, de l'aligner à droite ou à gauche par rapport à la position de l'élément 'text'. La valeur par défaut est ‘start’, le texte est aligné à gauche si le texte s'affiche de gauche à droite (nous verrons plus loin le sens d'affichage du texte). Avec 'middle', le texte est centré à la position définie par les attributs 'x' et 'y' de l'élément 'text' ou 'tspan'.

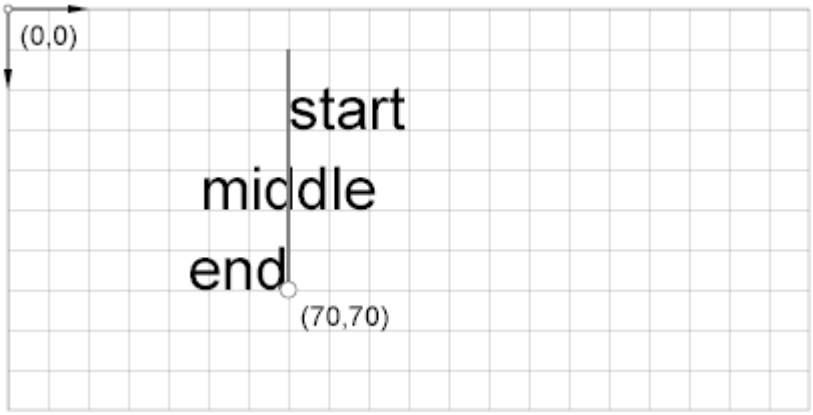

Figure 5-8. Alignement avec 'text-anchor'

```
<svg width="350" height="300">
  <text x="70" y="30"
      font-size="15">
      start
  </text>
  <text x="70" y="50"
      text-anchor="middle" font-size="15">
      middle
  </text>
  <text x="70" y="70"
      text-anchor="end" font-size="15">
      end
  </text>
</svg>
```

Dans cet exemple l'attribut 'text-anchor' permet l'alignement de tout ce qui est contenu dans l'élément 'text' à la position indiquée par 'x'.

Dans l'exemple suivant, nous utilisons 'tspan' pour centrer toutes les lignes du texte. Nous n'avons à gérer qu'une seule valeur de 'x' pour positionner tout notre texte.

Figure 5-9. 'tspan' et 'text-anchor'

```
<svg width="350" height="300">
  <!-- Circle -->
  <circle cx="120" cy="120" r="100"
      fill="darkred" stroke="black" stroke-width="3" />
  <!-- Advertisement Text -->
  <text x="120" y="60"
      fill="white" font-family="Verdana"
      text-anchor="middle">
      <tspan font-size="28">ALL</tspan>
      <tspan font-size="28" x="120" dy="35">Men's Pants</tspan>
      <tspan font-size="45" x="120" dy="55">
        <tspan font-size="30" baseline-shift="super">1</tspan>
        <tspan font-size="30">/</tspan>
        <tspan font-size="30" baseline-shift="sub">2</tspan> Off!
      </tspan>
      <tspan font-size="15" x="120" dy="45">(today only)</tspan>
  </text>
</svg>
```

La propriété ‘baseline-shift’ est utilisée pour positionner les trois éléments de “½”. La plupart des fontes possèdent l'information nécessaire à la mise en exposant ou en indice de caractères. Le caractère ‘1’ avec 'super' comme valeur pour 'baseline-shift' est un exposant, tandis que '2' avec 'sub' est un indice.

Remarquez également que nous devons rafraîchir la valeur de ‘x’ dans quelques éléments 'tspan', sinon le texte serait positionné par rapport à la fin de l'écriture du 'tspan' précédent. L'attribut ‘dy’ modifie la valeur de ‘y’ de départ. Après l'écriture d'un 'tspan', 'y' n'est pas actualisé automatiquement pour un passage à la ligne suivante.

Les propriétés 'letter-spacing', 'word-spacing' et 'kerning'

La valeur par défaut pour ‘letter-spacing’ et ‘word-spacing’ est ‘normal’. En leur donnant une valeur positive ou négative, nous augmentons ou diminuons l'espacement entre les lettres ('letter-spacing') et les mots ('word-spacing').

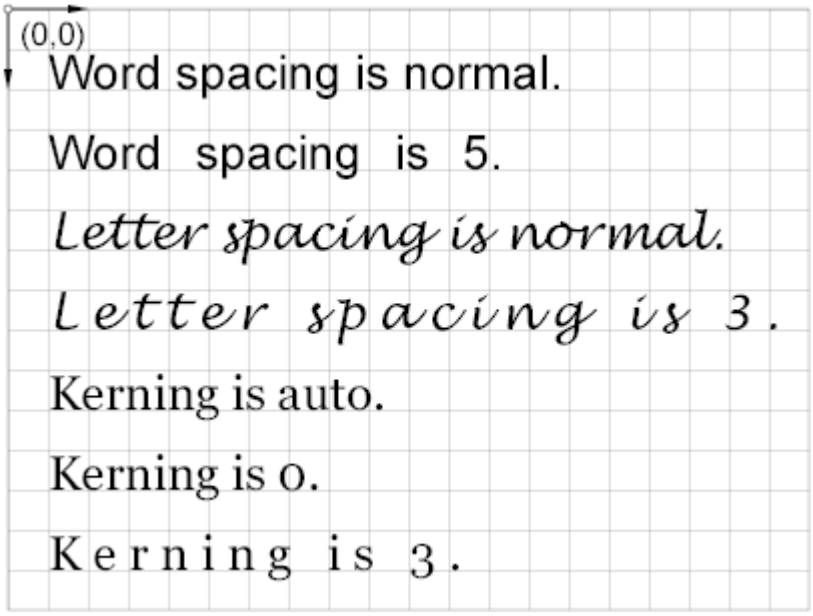

Figure 5-10. Spacing and kerning examples

```
<svg width="350" height="300">
  <text x="10" y="20" fill="black" font-family=""
      word-spacing="normal">Word spacing is normal.</text>
  <text x="10" y="40" fill="black" font-family=""
      word-spacing="5">Word spacing is 5.</text>
  <text x="10" y="60" fill="black" font-family="Lucida Handwriting"
      letter-spacing="normal">Letter spacing is normal.</text>
  <text x="10" y="80" fill="black" font-family="Lucida Handwriting"
      letter-spacing="3">Letter spacing is 3.</text>
  <text x="10" y="100" fill="black" font-family="Georgia"
      kerning="auto">Kerning is auto.</text>
  <text x="10" y="120" fill="black" font-family="Georgia"
      kerning="0">Kerning is 0.</text>
  <text x="10" y="140" fill="black" font-family="Georgia"
      kerning="3">Kerning is 3.</text>
</svg>
```

La syntaxe de ces propriétés:

'letter-spacing'

Valeur:	normal | <length> | inherit
Par défaut:	normal
S'applique à:	éléments 'text'
Transmissible:	oui
Pourcentages:	N/A
Media:	visuel
Animable:	oui

'word-spacing'

Valeur:	normal | <length> | inherit
Par défaut:	normal
S'applique à:	éléments 'text'
Transmissible:	oui
Pourcentages:	N/A
Media:	visuel

Animable: oui

SVG définit la propriété 'kerning' dans le but de pouvoir modifier l'espace entre lettres pour des fontes spécifiques. La même lettre dans dix fontes différentes peut avoir dix largeurs différentes. L'espace entre certaines lettres est défini dans une table 'fonts kern-pair table'.

La valeur par défaut de ‘kerning’ est ‘auto’, dans ce cas la table est utilisée pour espacer les caractères. En donnant une autre valeur, la table n'est plus utilisée et les lettres sont espacées de la valeur indiquée

'kerning'

Valeur: auto | <length> | inherit
Par défaut: auto
S'applique à: éléments 'text'
Transmissible: oui
Pourcentages: N/A
Media: visuel
Animable: oui

Les propriétés 'textLength' et 'lengthAdjust'

Dans cet exemple la phrase "The average person thinks he isn't." du Père Larry Lorenzoni est utilisée pour illustrer les propriétés ‘textLength’ et ‘lengthAdjust’. Vous pouvez avec ces propriétés forcer le texte à occuper une largeur précise.

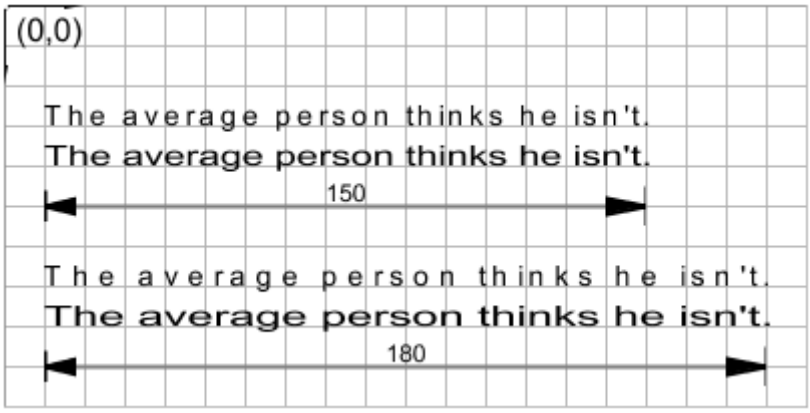

Figure 5-11. textLength and lengthAdjust attributes

Ce document SVG est rendu avec le visualiseur de Batik car celui d'Adobe, dans sa version 3, ne gère pas ces propriétés.

```
<svg width="350" height="300">
  <text x="10" y="30" fill="black" font-size="7" textLength="150"
      lengthAdjust="spacing">The average person thinks he isn&apos;t.</text>
  <text x="10" y="40" fill="black" font-size="7" textLength="150"
      lengthAdjust="spacingAndGlyphs">The average person thinks he
isn&apos;t.</text>
  <text x="10" y="70" fill="black" font-size="7" textLength="180"
      lengthAdjust="spacing">The average person thinks he isn&apos;t.</text>
<text x="10" y="80" fill="black" font-size="7" textLength="180"
```

```
      lengthAdjust="spacingAndGlyphs">The average person thinks he
isn&apos;t.</text>
</svg>
```

Avec ‘textLength’ nous indiquons la largeur que doit occuper le texte. La propriété ‘lengthAdjust’ peut avoir les valeurs ‘spacing’ ou ‘spacingAndGlyphs’. Comme vous le voyez, avec la valeur ‘spacingAndGlyphs’, les espaces entre caractères ou mots ainsi que les lettres elles-mêmes sont étirés pour occuper toute la place allouée. Avec 'spacing', seuls les espaces sont étirés.

Notons que la propriété ‘lengthAdjust’ ne peut être utilisée qu'avec l'élément ‘text’ et non avec 'tspan’.

La propriété 'rotate'

La propriété ‘rotate’ permet de faire tourner des lettres ou des lignes complètes de texte.

Avec une virgule ou un espace comme séparateur, nous pouvons affecter une liste de valeurs à l'attribut 'rotate' comme nous le voyons sur la figure 5-12.

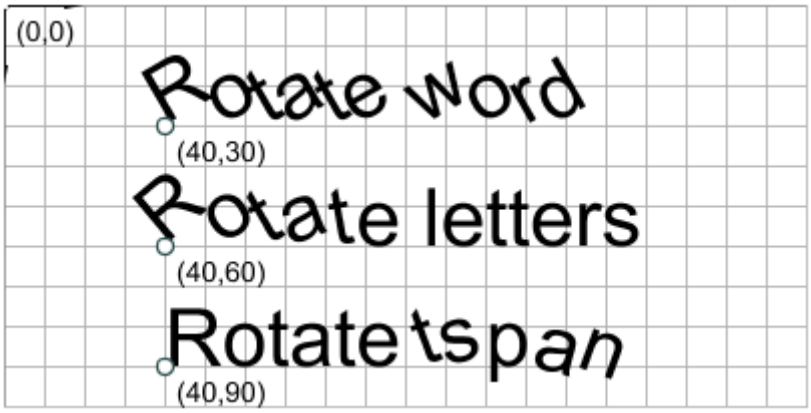

Figure 5-12. Rotate text

```
<svg width="350" height="300">
  <text x="40" y="30" fill="black"
      font-size="20" rotate="-30">Rotate word</text>
  <text x="40" y="60" fill="black"
      font-size="20" rotate="-40 -35 -30 -20 -10">Rotate letters</text>
  <text x="40" y="90" fill="black" font-size="20"
      rotate="0 0 0 0 0 0 0 -20 -15 0 15 20">
      <tspan>Rotate tspan</tspan>
  </text>
</svg>
```

Cette image est réalisée avec Batik. Le plugin d'Adobe dans sa version 3 ne supporte pas la rotation de chaînes entières avec un seul nombre comme paramètre. Dans la figure 5-12, remarquez que si nous n'entrons que deux nombres, seules les deux premières lettres subissent une rotation.

La propriété 'transform' peut s'appliquer à un élément 'text' mais elle ne pourra que s'appliquer globalement à tout le texte. Nous verrons ces transformations au prochain chapitre.

La propriété 'writing-mode'

L'attribut 'writing-mode' permet de donner la direction d'écriture du texte.

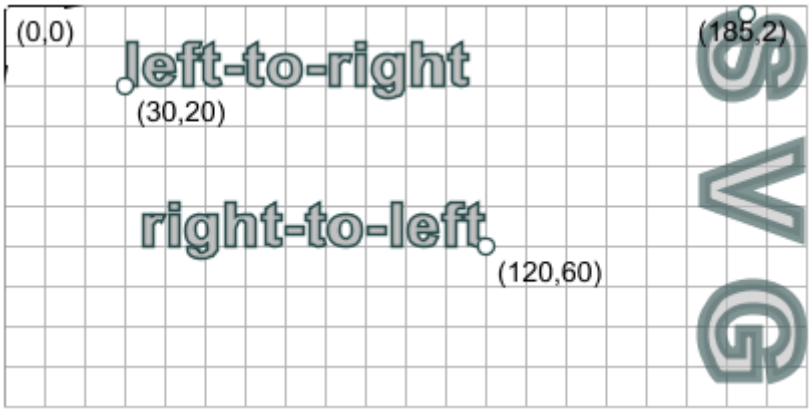

Figure 5-13. La propriété 'writing-mode'

```
<svg width="800" height="600" viewBox="0 0 400 300">
      <text x="30" y="20"
            fill="silver" stroke="darkslategrey" stroke-width=".7" opacity="1"
            font-weight="bold" font-size="15"
            writing-mode="lr-tb">left-to-right</text>
      <text x="120" y="60"
            fill="silver" stroke="darkslategrey" stroke-width=".7" opacity="1"
            font-weight="bold" font-size="15"
            writing-mode="rl-tb">right-to-left</text>
      <text x="185" y="2"
            fill="silver" stroke="darkslategrey" stroke-width="3" opacity="0.6"
            font-weight="bold" font-size="33"
            writing-mode="tb-rl">S V G</text>
</svg>
```

Nous pouvons utiliser cet attribut pour positionner du texte avec précision. Voici sa syntaxe

'writing-mode'

Valeur:	lr-tb | rl-tb | tb-rl | lr | rl | tb | inherit
Par défaut:	lr-tb
S'applique à:	éléments 'text'
Transmissible:	oui
Pourcentages:	N/A
Media:	visuel
Animable:	non

Les valeurs lr, rl et tb sont des raccourcis de lr-tb (gauche à droite et haut en bas), rl-tb (droite à gauche et haut en bas) et tb-rl respectivement.

La propriété 'glyph-orientation'

Une petite remarque sur le mot 'glyph', ce mot peut se traduire par caractère bien que le mot glyphe ait figuré au dictionnaire ('trait gravé en creux dans un ornement' dans le petit Larousse 1981) puis disparu. Le mot anglais 'character' désigne le codage lui-même de la

lettre ou du symbole. Nous devons donc prendre le mot caractère au sens de la représentation sur le papier, à l'écran ou pourquoi pas sur la pierre de la lettre ou du symbole.
Les propriétés 'glyph-orientation' permettent de faire tourner vos caractères d'un angle donné avec l'axe horizontal ou l'axe vertical. Si vous afficher un texte de haut en bas, en donnant à 'glyph-orientation-vertical' la valeur zéro, chaque caractère de votre texte sera aligné avec l'axe vertical comme le montre cette figure.

Figure 5-14. La propriété 'glyph-orientation-vertical"

Voici la syntaxe de ces propriétés:
Attention les seules valeurs autorisées pour l'angle sont 0, 90, 180 et 270 en degrés.

'glyph-orientation-vertical'

Valeur:	auto | <angle> | inherit
Par défaut:	auto
S'applique à:	éléments 'text'
Transmissible:	oui
Pourcentages:	N/A
Media:	visuel
Animable:	non

'glyph-orientation-horizontal'

Valeur:	<angle> | inherit
Par défaut:	0 degré
S'applique à:	éléments 'text'
Transmissible:	oui
Pourcentages:	N/A
Media:	visuel
Animable:	non

Les propriétés 'direction' et 'unicode-bidi'

De nombreuses langues s'écrivent dans des directions autres que gauche à droite, par exemple la langue arabe de droite à gauche ou la langue chinoise de haut en bas.

Si vous voulez que les caractères de votre texte soient pris dans l'ordre correct, vous devez utiliser les propriétés 'direction' et 'bi-directionality'.

Voici la syntaxe de ces propriétés.

'direction'

Valeur:	ltr | rtl | inherit
Par défaut:	ltr
S'applique à:	éléments 'text'
Transmissible:	oui
Pourcentages:	N/A
Media:	visuel
Animable:	non

'unicode-bidi'

Valeur:	normal | embed | bidi-override | inherit
Par défaut:	normal
S'applique à:	éléments 'text'
Transmissible:	Non
Pourcentages:	N/A
Media:	Visuel
Animable:	Non

Vous trouverez des exemples de ces propriétés sur notre site – www.learnsvg.com

Transmission des propriétés

La notion de transmission ou d'héritage des propriétés est très utile en SVG et plus généralement dans les langages basés sur XML. En fait la transmission des propriétés de style en SVG fonctionne exactement comme en HTML comme le montre cet exemple.

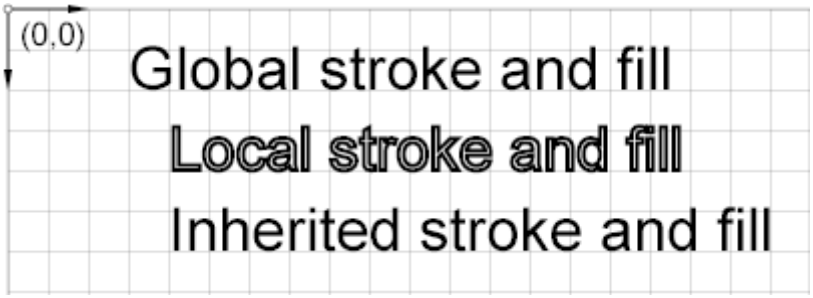

Figure 5-15. Transmission de 'stroke' et 'fill'

```
<svg width="350" height="300">
  <text x="30" y="20"
      stroke="none" fill="black" font-size="15">
      Global stroke and fill
      <tspan x="40" dy="20" stroke="black" fill="none">Local stroke and
fill</tspan>
      <tspan x="40" dy="20">Transmissible stroke and fill</tspan>
  </text>
</svg>
```

Remarquez que la propriété ‘fill’ de l'élément ‘text’ est remplacée par la valeur donnée dans le premier élément 'tspan', par contre elle s'applique dans le deuxième 'tspan' où aucune valeur n'a été donnée à 'fill'. Nous retrouvons les mêmes règles de transmission que nous avons vues aux chapitres précédents.

Ceci s'applique aussi à la propriété ‘rotate’. Les attributs 'x', 'y', 'dx', 'dy' et 'rotate' de l'élément 'text' peuvent prendre comme valeur une liste de nombres. Ces listes de nombres s'appliquent également aux descendants immédiats de l'élément 'text', à savoir les éléments 'tspan' inclus dans celui-ci.

Texte sur une courbe

L'élément **‘textPath’** permet de faire référence à un élément 'path'. Le texte sera alors disposé sur cette courbe. Ceci nous permet de disposer du texte de n'importe quelle manière.

Voici la syntaxe de cet élément:

```
<textPath   id="name"
            xlink:href="<uri>"
            startOffset="<length>"
            method="'align'|'stretch'"
            spacing="'auto'|'exact'"
            lengthAdjust="'spacing'|'spacingAndGlyphs'"
            style-attribute="style-attribute">
            <!-- texte à afficher -->
</textPath>
```

Comme SVG utilise la grammaire XML, nous pouvons faire référence à d'autres éléments en utilisant l'attribut “xlink:href”. L'élément 'textPath' aura l'attribut 'xlink:href' qui définira la courbe utilisée pour disposer notre texte.

Les autres attributs en quelques mots, 'lengthAdjust' a la même fonction que dans l'élément 'text' comme nous l'avons vu.
'startOffset' indique la position de départ de notre texte sur la courbe, il sera en général exprimé par un pourcentage, 0% et le texte commence au début de la courbe.
'method' définit la manière de représenter les caractères, avec 'align' valeur par défaut les caractères du texte seront simplement transformés pour être affichés, avec 'stretch' ils sont convertis en courbes et collent mieux à la courbe.
'spacing' définit l'écartement entre les caractères, avec 'exact' valeur par défaut, les règles habituelles sont appliquées et avec 'auto' les espacements sont adaptés à la courbe.

L'élément 'textPath' supporte également tous les attributs d'un contenu texte, en particulier le choix des fontes, leur taille, les attributs de présentation etc … et également 'text-anchor'.

Sur cet exemple de la figure 5-16, vous pouvez voir diverses dispositions du texte avec l'élément ‘textPath’.

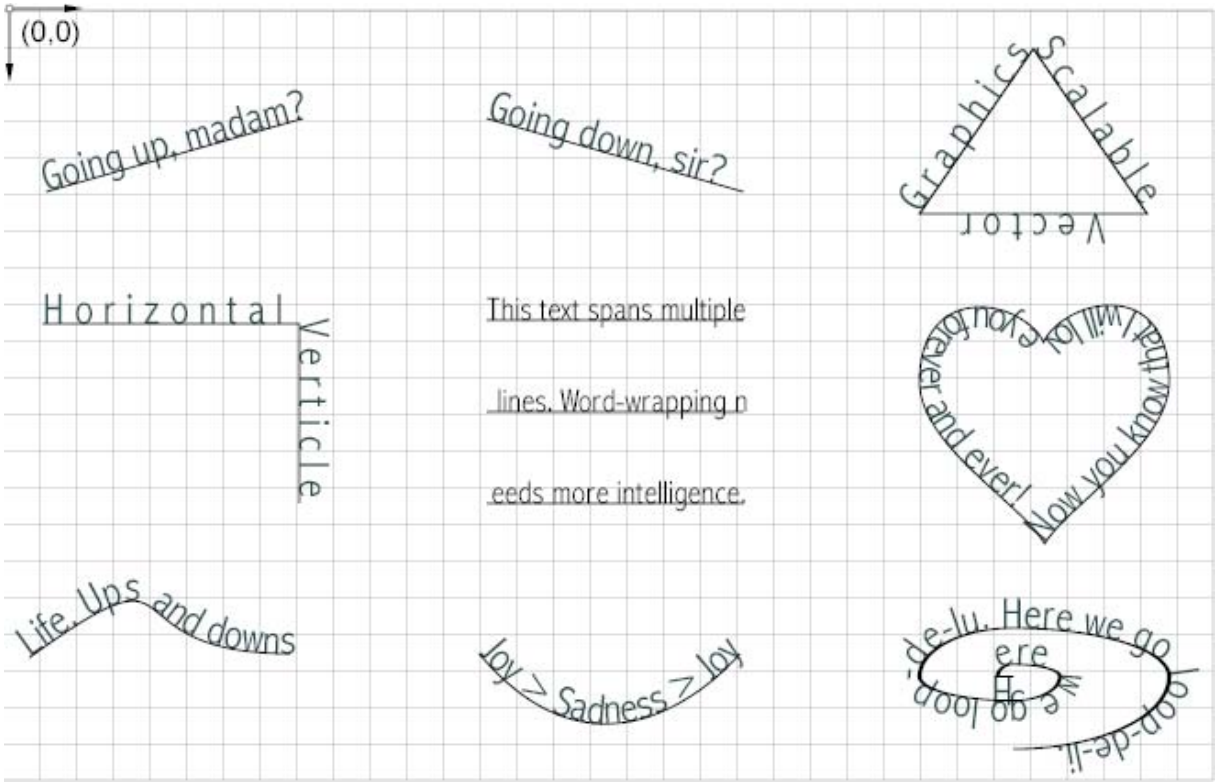

Figure 5-16. Texte sur des courbes

```
<svg width="800" height="600" viewBox="0 0 400 300">
  <!-- Paths →
  <path id="textPath01" d="M10 50 L80 30"
        fill="none" stroke="black" stroke-width="0.5"/>
  <path id="textPath02" d="M130 30 L200 50"
        fill="none" stroke="black" stroke-width="0.5"/>
<path id="textPath03" d="M279 11 L304.981 56 L253.019 56 z"
      transform="matrix(1.19319 0 0 1 -53.9002 4.77396e-015)"
       fill="none" stroke="black" stroke-width="0.5"/>
<path id="textPath04" d="M9 86 L79 86 L79 135"
       fill="none" stroke="black" stroke-width="0.5"/>
<path id="textPath05" d="M130 85 L200 85 M130 110 L200 110 M130 135 L200 135"
       fill="none" stroke="black" stroke-width="0.5"/>
<path id="textPath06"
      d="M282.057 146 C288.599 135.355 316 118.885 316 101.844
      C316 72.3472 87.46 79.366
      281.796 91.016 C275.78 79.0081 248 74.2157 248 101.844
      C248 119.3 276.338 135.443 282.057 146 z"
      fill="none" stroke="black" stroke-width="0.5"/>
<path id="textPath07" d="M5.5 177 C52.5 143 24.5 176 76.5 176"
       fill="none" stroke="black" stroke-width="0.5"/>
<path id="textPath08" d="M128 176 C164 219 199 176 199 176"
       fill="none" stroke="black" stroke-width="0.5"/>
<path id="textPath09"
      d="M278.7 188.8 C277.789 188.8 277.05 188.8 277.05 188.8
      C277.05 190.623 277.789 192.1 278.7 192.1
      C281.434 192.1 283.65 190.623 283.65 188.8
      C283.65 185.155 281.434  182.2 278.7 182.2
      C273.232 182.2 268.8 185.155 268.8 188.8
      C268.8 196.09 273.232 202 278.7 202
      C287.813 202 295.2 196.09 295.2 188.8
```

```
      C295.2 177.865 287.813 169 278.7 169"
      transform="matrix(2.57576 0 0 -1 -444.364 371.014)"
      fill="none" stroke="black" stroke-width="0.5"/>
 <!-- Text →
 <text x="50px" y="80px" visibility="visible"
      fill="darkslategrey" font-size="12" stroke="none">
      <textPath xlink:href="#textPath01">Going up, madam?</textPath>
      <textPath xlink:href="#textPath02">Going down, sir?</textPath>
      <textPath xlink:href="#textPath03">S c a l a b l e     V e
c t o r    G r a p h  i c s</textPath>
      <textPath xlink:href="#textPath04">H o r i z o n t a l V e r t i c l
e</textPath>
      <textPath xlink:href="#textPath05" font-size="9" fill="black">This text
spans multiple lines.  Word-wrapping needs more intelligence.</textPath>
      <textPath xlink:href="#textPath06">Now you know that I will love you
forever and ever!</textPath>
      <textPath xlink:href="#textPath07">Life. Ups and downs.</textPath>
      <textPath xlink:href="#textPath08">Joy &gt; Sadness &gt; Joy</textPath>
      <textPath xlink:href="#textPath09">Here we go loop-de-lu.  Here we go
loop-de-li.</textPath>
  </text>
</svg>
```

Les courbes référencées doivent avoir un attribut 'id' pour que nous puissions les utiliser comme référence. Ces courbes peuvent ne pas être dessinées si nous les mettons dans une section <defs>.

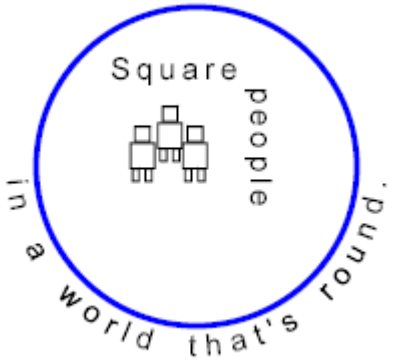

Figure 5-17. Autre exemple de texte sur des courbes

```
<svg width="350" height="300">
      <defs>
            <path id="textPath01" d="M90 80 L160 80 L160 160"/>
            <path id="textPath03"
                  d="M133 44 C88.8176 44 53 79.5937 53 123.5
                        C53 167.406 88.8176 203 133 203 C177.182
                        203 213 167.406 213 123.5
                        C213 79.5937 177.182 44 133 44 z"/>
            <g id="people">
                  <rect x="222" y="9" width="30" height="30"
                        stroke="black" stroke-width="5" fill="none"/>
                  <rect x="213" y="42" width="48" height="51"
                        stroke="black" stroke-width="5" fill="none"/>
                  <rect x="217" y="93" width="13" height="25"
                        stroke="black" stroke-width="5" fill="none"/>
```

```
                <rect x="243" y="93" width="13" height="25"
                      stroke="black" stroke-width="5" fill="none"/>
            </g>
      </defs>
      <use x="0" y="0" xlink:href="#textPath03"
           fill="none" stroke="blue" stroke-width="3"/>
      <text font-size="15" letter-spacing="3">
            <textPath xlink:href="#textPath01">
                  Square people
            </textPath>
      </text>
      <text font-size="15" baseline-shift="-15" letter-spacing="3"
            word-spacing="5" text-anchor="start" fill="black">
            <textPath xlink:href="#textPath03" startOffset="26%">
                  in a world that&apos;s round.
            </textPath>
      </text>
      <g transform="translate(58 95) scale(0.25 0.25)">
            <g transform="translate(-45 25)">
                  <use x="0" y="0" xlink:href="#people"/>

            </g>
            <g transform="translate(10 -15)">
                  <use x="0" y="0" xlink:href="#people"/>
            </g>
            <g transform="translate(60 25)">
                  <use x="0" y="0" xlink:href="#people"/>
            </g>
      </g>
</svg>
```

Revenons à l'attribut 'startOffset', dans notre dernier exemple il a la valeur de 26% pour le texte “in a world that’s round.”. La position du texte est liée à la manière de décrire la courbe, en particulier du choix du point de départ de la courbe. Remarquons que les formes de base ne peuvent servir de référence pour placer du texte, ici le cercle bleu utilise l'élément 'path' pour être défini. Si nous donnons à 'startOffset' la valeur 100% et que nous gardons avec text-anchor l'alignement à gauche, nous ne verrons aucun texte s'afficher!

Pour terminer, signalons l'intérêt de l'attribut 'startOffset' pour créer des animations, comme des textes flottant sur une courbe de Bézier ou défilant simplement. Nous verrons quelques exemples au chapitre 9.

Propriétés des polices

C'est un sujet intéressant et il y a tant de possibilités qu'un livre entier n'y suffirait pas. Aussi nous ne donnerons qu'un bref aperçu sur les polices et quelques propriétés élémentaires.

Taille de la police

Nous avons déjà utilisé dans nos exemples la propriété 'font-size'.
Voici un exemple de différentes tailles avec la même police.

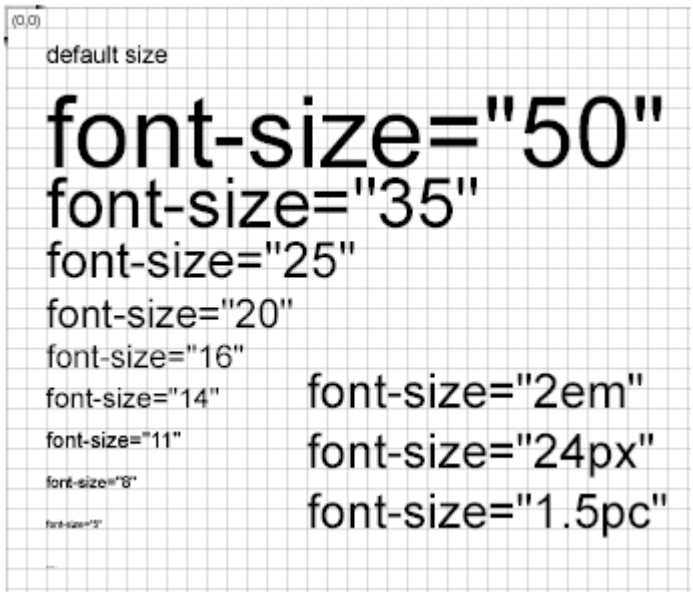

Figure 5-18. Différentes tailles de police

```
<svg width="350" height="300">
  <text x="20" y="28" fill="black">default size</text>
  <text x="20" y="80" fill="black"
      font-size="50" stroke="0">font-size="50"</text>
  <text x="20" y="110" fill="black"
      font-size="35" stroke="0">font-size="35"</text>
  <text x="20" y="135" fill="black" stroke="0"
      font-size="25">font-size="25"</text>
  <text x="20" y="160" fill="black" stroke="0"
      font-size="20">font-size="20"</text>
  <text x="20" y="180" fill="black" stroke="0"
      font-size="16">font-size="16"</text>
  <text x="20" y="200" fill="black" stroke="0"
      font-size="14">font-size="14"</text>
  <text x="20" y="220" fill="black" stroke="0"
      font-size="11">font-size="11"</text>
  <text x="20" y="240" fill="black" stroke="0"
      font-size="8">font-size="8"</text>
  <text x="20" y="260" fill="black" stroke="0"
      font-size="5">font-size="5"</text>
  <text x="20" y="280" fill="black"
      font-size="1">font-size="1"</text>
  <text x="150" y="200" fill="black" stroke="0"
      font-size="2em">font-size="2em"</text>
  <text x="150" y="230" fill="black" stroke="0"
      font-size="24px">font-size="24px"</text>
  <text x="150" y="260" fill="black" stroke="0"
      font-size="1.5pc">font-size="1.5pc"</text>
</svg>
```

La propriété 'font-size' peut être héritée d'éléments parents. Si vous voulez spécifier une unité, utilisez ‘em’ plutôt que ‘px’ pour ne pas être tributaire de l'environnement dans lequel votre document SVG sera visualisé.

Syntaxe de la propriété 'font-size'.

'font-size'

Valeur:	<absolute-size> | <relative-size> | <length> | <percentage> | inherit
Par défaut:	medium
S'applique à:	éléments 'text'
Transmissible:	oui
Pourcentages:	par rapport à la taille de l'élément parent
Media:	visuel
Animable:	oui

Polices et caractères (glyphes)

Il nous faut bien distinguer polices, lettres et caractères.

La représentation des lettres se fait à l'aide de caractères, une lettre peut faire appel à plusieurs caractères (par exemple la lettre et son accent). Mais les caractères ne représentent pas toujours des lettres, ils peuvent également représenter des symboles.

Aussi, la même lettre sera représentée par des caractères différents selon la police utilisée. PostScript et TrueType sont des formats vectoriels pour l'écriture des caractères. Les polices Verdana et Arial, créées par 'The Monotype company' sont parmi les plus connues sur le Web.

Le point du ‘i’ est un caractère et est utilisé avec d'autres pour représenter une lettre. Ces lettres forment une police de caractères.

Choix de la police

Voici un exemple des polices les plus communes en SVG.

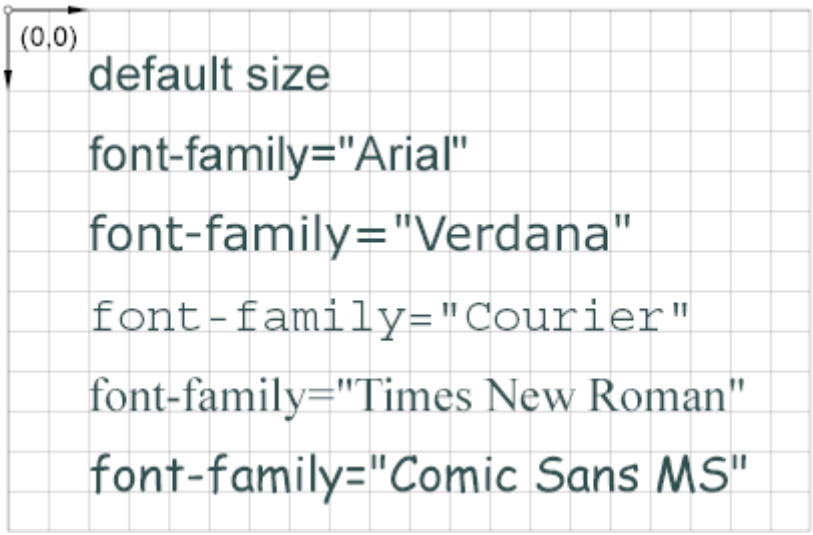

Figure 5-19. Quelques polices courantes

```
<svg width="350" height="300">
  <text x="20" y="20" fill="darkslategrey">default size</text>
  <text x="20" y="40" fill="darkslategrey"
      font-family="Arial"
      stroke="0">font-family="Arial"</text>
   <text x="20" y="60" fill="darkslategrey"
      font-family="Verdana"
      stroke="0">font-family="Verdana"</text>
```

```
  <text x="20" y="80" fill="darkslategrey"
      font-family="Courier"
      stroke="0">font-family="Courier"</text>
  <text x="20" y="100" fill="darkslategrey"
      font-family="Times New Roman"
      stroke="0">font-family="Times New Roman"</text>
  <text x="20" y="120" fill="darkslategrey"
      font-family="Comic Sans MS"
      stroke="0">font-family="Comic Sans MS"</text>
</svg>
```

Syntaxe de la propriété 'font-family'

'font-family'

Valeur:	[[<family-name> |<generic-family>],]* [<family-name> | <generic-family>] | inherit
Par défaut:	dépend du visualiseur et du navigateur
S'applique à:	éléments 'text'
Transmissible:	oui
Pourcentages:	N/A
Media:	visuel
Animable:	oui

Polices système

Des polices système sont des polices que l'on peut supposer être installées sur tout ordinateur. En les utilisant, nous supposons que nous n'avons pas à donner d'information sur ces polices dans notre document SVG. Parmi ces polices, nous trouvons: sans-serif, Arial, Times New Roman, Helvetica, Verdana, et Tahoma.

Tables et polices

Une police nécessite une information pour définir chaque caractère. Cette information est sous forme de table et permet d'avoir la même représentation quelque soit le support (écran, imprimante, …).

Polices SVG

Si vous souhaitez personnaliser vos polices et utiliser des polices plus exotiques, SVG vous permet d'inclure ces polices dans votre document. Ces polices sont nommées polices SVG.

Vous pouvez trouver un convertisseur 'SVGFont' de Steady State (http://www.steadystate.co.uk/svg/) qui convertit les polices en données utilisables par SVG. SVGFont a été incorporé à Batik ('Batik SVG Toolkit').

une autre possibilité pour créer des polices SVG est Illustrator 9 ou 10 qui peut incorporer automatiquement les polices utilisées à votre document SVG.

Polices incorporées

Quand vous créez des polices SVG, par exemple avec Illustrator 10, les caractères et la table sont encodés sur 64 bits et mis dans un élément <style>:

```
<style type="text/css">
      <![CDATA[
      .st0{font-family:'Verdana';}
      @font-face{font-family:'Verdana';src:url("data:; base64,(encoded data…)”)}
      ]]>
</style>
```

Cette police incorporée est utilisée en utilisant CSS comme ici:

```
<text>
  <tspan class="st0">Am I speaking with Verdana?</tspan>
</text>
```

Polices CEF externes

```
<style type="text/css">
      <![CDATA[
   .st0{font-family:'Verdana';}
      @font-face{font-family:'Verdana';src:url(79BA9154.cef)}
      ]]>
</style>
```

Le fichier CEF doit être localisé dans le même répertoire que le document SVG et vous l'utilisez de la même manière:

```
<text>
  <tspan class="st0">Am I speaking with Verdana?</tspan>
</text>
```

Référencer un contenu texte

Le contenu de votre texte peut être externe ou interne à votre document. L'élément <tref> permet de l'utiliser dans un élément 'text'

Il est plus avantageux d'utiliser un texte référencé que de copier de multiples fois le même texte dans votre document. Les modifications sont plus aisées et la taille de votre fichier moindre.

Texte référencé

Le texte référencé est utilisé avec l'attribut 'xlink:href' comme le montre cet exemple.

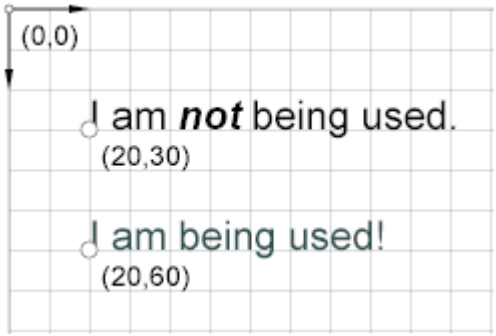

Figure 5-20. L'élément 'tref'

```
<svg width="350" height="300">
        <defs>
                <text id="UseMe" x="20" y="30">I <tspan font-style="italic"
font-weight="bolder">am</tspan> being used!
                </text>
        </defs>
        <text x="20" y="30" font-size="10">I am <tspan font-style="italic"
font-weight="bolder">not</tspan> being used.
        </text>
        <text x="20" y="60" font-size="10" fill="orange">
                <tref xlink:href="#UseMe" fill="darkslategrey" />
        </text>
</svg>
```

Ceci permet de réutiliser facilement un texte.

Internationalisation

SVG supporte l'internationalisation avec le codage Unicode et la gestion multilangues grâce à l'élément <switch>.

Encodage Unicode

Unicode est un standard ISO qui supporte tous les codes de caractères des différentes langues. Vous êtes sans doute familier avec un encodage tel que " " qui représente un espace en HTML. C'est un code de caractère Unicode. Unicode définit déjà les caractères utilisés par des centaines de langues à travers le monde.

SVG supporte aussi bien ce codage que n'importe quel autre. Il suffit de déclarer le codage retenu dans l'en-tête du document:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
```

Cette en-tête ne vous permettra pas l'affichage des lettres accentuées comme é, è à …

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

Celle-ci vous permettra de mettre directement les lettres accentuées dans votre contenu texte:

```
<text x="40" y="50" fill="black" font-size="12" font-family="Verdana">
Je suis très heureux de pouvoir utiliser des lettres accentuées
</text>
```

Restera le problème d'utiliser des caractères spécifiques au langage XML dans un contenu texte comme <, > … qui seraient interprétés comme des débuts ou fins de balises.

Ces lettres devront être codées comme ceci
Pour <, vous devrez utiliser < (décimal) ou < (hexadécimal) ou < (prédéfini)
Pour >, vous devrez utiliser > (décimal) ou > (hexadécimal) ou > (prédéfini)
Ce qui nous donnera:

```
<text x="40" y="50" fill="black" font-size="12" font-family="Verdana">
Je suis très heureux de pouvoir utiliser &lt; et &gt; dans mon texte
</text>
```

Nous avons mis dans l'annexe B des tables des caractères les plus utilisés, caractères alphabétiques ou symboles.

L'élément 'switch'

L'élément switch permet qu'à l'affichage du document, un choix soit fait sur le poste client, en fonction de la langue de l'utilisateur par exemple ou des possibilités du visualiseur, pour l'affichage de texte ou d'images.
Avec cet exemple, la bulle affichée sera dans la langue de l'utilisateur.

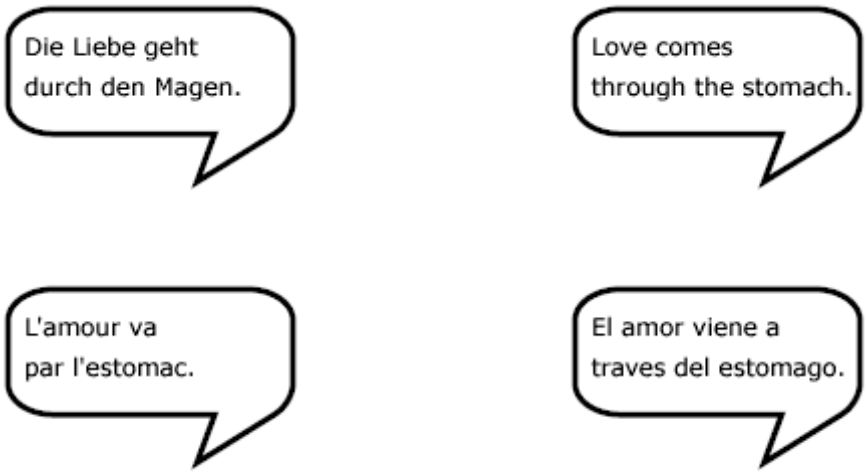

Figure 5-21. L'élément 'switch'

```
<svg width="350" height="300">
   <!-- Bulle -->
   <path transform="translate(-150 -80)"
       d="M314.669 164.186 C320.812 160.386 324 153.585 324 148.857
              L324 117.286 C324 108.844 313.907 102 301.457 102
              L204.543 102 C192.093 102 182 108.844 182 117.286
              L182 148.857 C182 157.299 192.093 164.143 204.543 164.143
              L285.024 164.143 L276.175 188 z"
              fill="white" stroke="black" stroke-width="3"/>
   <switch>
       <!-- Système en allemand -->
       <text systemLanguage="de" x="40" y="50"
              fill="black" font-size="12" font-family="Verdana">
         <tspan dy="-5">Die Liebe geht</tspan>
         <tspan x="40" dy="20">durch den Magen.</tspan>
       </text>
       <!-- Système en anglais -->
       <text systemLanguage="en" x="40" y="50"
              fill="black" font-size="12" font-family="Verdana">
         <tspan dy="-5">Love comes</tspan>
         <tspan x="40" dy="20">through the stomach.</tspan>
       </text>
       <!-- Système en français -->
       <text systemLanguage="fr" x="40" y="50"
              fill="black" font-size="12" font-family="Verdana">
         <tspan dy="-5">L'amour va</tspan>
         <tspan x="40" dy="20">par l'estomac.</tspan>
       </text>
```

```
      <!-- Système en espagnol -->
      <text systemLanguage="es" x="40" y="50"
            fill="black" font-size="12" font-family="Verdana">
        <tspan dy="-5">El amor viene a</tspan>
        <tspan x="40" dy="20">traves del estomago.</tspan>
      </text>
  </switch>
</svg>
```

Ceci montre les possibilités de SVG de produire des documents adaptés à l'utilisateur.
Vous trouverez des informations supplémentaires sur cet aspect à
document RFC: http://www.ietf.org/rfc/rfc3066.txt
Codes pour les différentes langues:
http://www.din.de/gremien/nas/nabd/iso3166ma/codlstp1/index.html

Gestion des espaces

Nous mettrons dans une même catégorie les espaces, les tabulations, les retours chariot dans un contenu texte. Nous pouvons gérer ces caractères de plusieurs manières suivant les spécifications XML 1.0 avec les valeurs données à “xml:space”.

Nous avons deux valeurs possibles pour la propriété 'xml:space', ‘default’ et ‘preserve’.

- **default** (également valeur par défaut pour xml:space) – Si nous indiquons xml:space="default", SVG supprimera les retours chariot, convertira les tabulations en espaces, supprimera les espaces de début et de fin et enfin réduira les espaces contigus à un seul
- **preserve** - Avec xml:space="preserve", SVG convertira les retours chariot et les tabulations en espaces et conservera tous les espaces, même contigus.

Ainsi avec xml:space="preserve", la chaîne "a b" (trois espaces entre "a" et "b") sera affichée avec trois espaces entre a et b, alors qu'avec xml:space="default" ou aucune indication, il ne sera affiché qu'un espace entre a et b.

Afficher des blocs de texte

Comme nous l'avons dit, il est particulièrement pénible de travailler avec de longs textes car nous n'avons pas de possibilité de positionner automatiquement le texte dans les spécifications SVG 1.0.

D'un autre côté XHTML est tout à fait adapté pour l'affichage de longs textes, pourquoi ne pas l'utiliser pour les blocs de texte? Ceci peut se faire en mixant les languages, c'est ce que nous verrons en détail au chapitre 11.

Sommaire

Maintenant que nous avons vu les basiques de SVG comme formes, structure, courbes et texte, nous allons nous intéresser à des choses encore plus excitantes comme les gradients, les filtres, l'animation et les scripts.

Cependant un chapitre un peu délicat nous attend, les systèmes de coordonnées et les transformations.

Les points forts de SVG

Résumons quelques uns des points forts de SVG que nous avons vu:

- Positionnement précis des objets
- Redimensionnement sans perte de qualité
- Fichiers textes éditables et modifiables dans un simple éditeur
- Taille réduite des fichiers
- Infinité de couleurs et de polices

Dans les prochains chapitres nous allons en voir d'autres:

- Effets natifs sur les images
- Animation
- Script pour générer, modifier le document suivant les requêtes de l'utilisateur
- Interactivité
- Son
- Compatibilité avec XML, XLink, SMIL, conformité à CSS, XSL, et DOM
- D'ores et déjà, de nombreux outils permettent de créer un contenu SVG