Chapitre 11 Mixer les langages XML

Aperçu

SVG est prévu pour être utilisé avec les technologies XML telles que DOM, XHTML, CSS, XSL, CML, MathML, SMIL et RDF. Cette image montre un exemple du bénéfice que nous pouvons en tirer, utiliser XHTML, SVG, et MathML dans un même document.

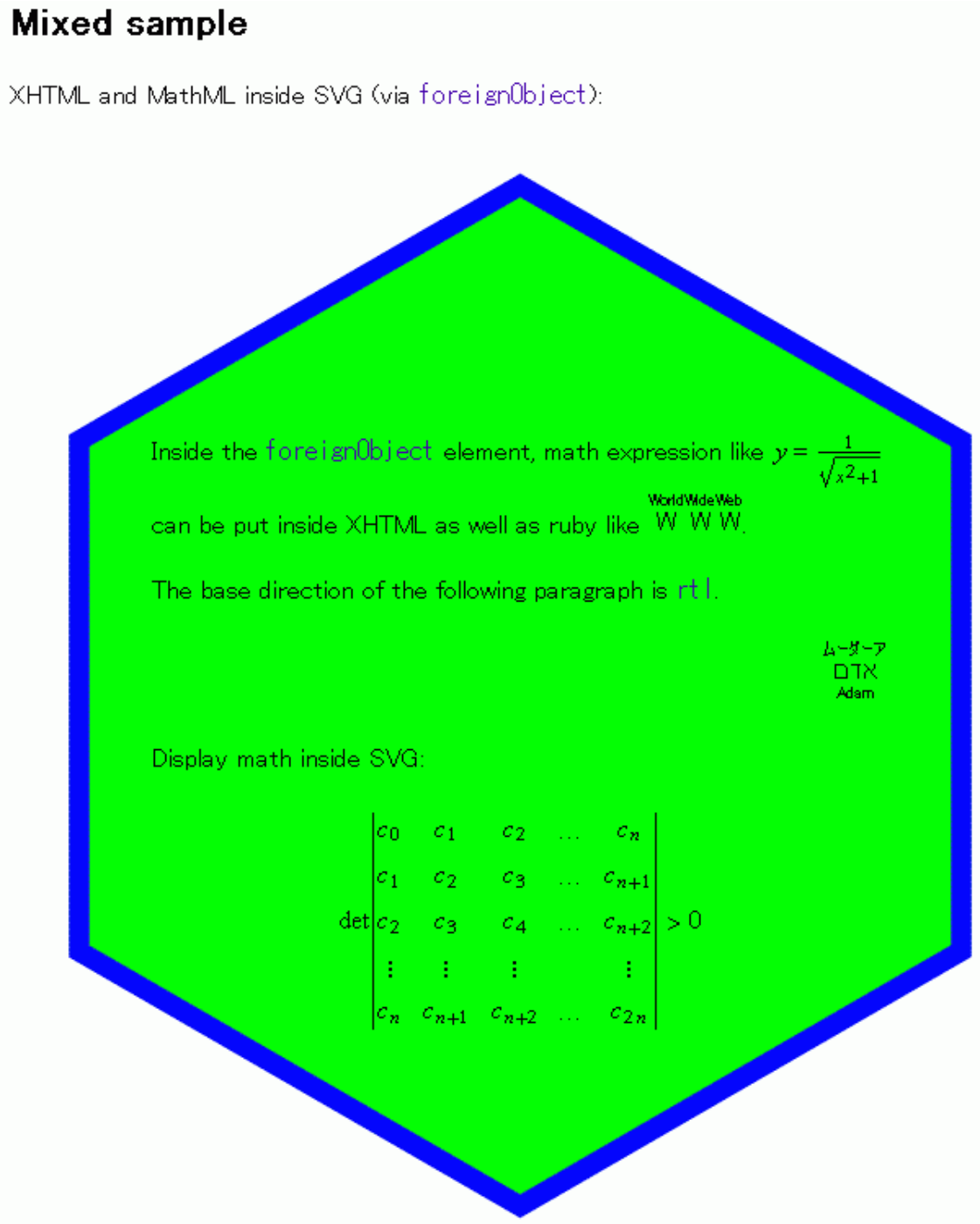

Figure 11-1. SVG, XHTML et MathML!

Espaces de noms et extensibilité

Les documents SVG utilisent la définition du type de document (Document Type Definition ou DTD) du W3C (www.w3c.org/Graphics/SVG). Cette définition précise les règles et la syntaxe du langage SVG. Pour mieux comprendre le fonctionnement de SVG, revenons à XML et son extensibilité.

XML est la grammaire à la base de toute une génération de langages basés sur les balises qui travaillent main dans la main avec les autres espaces de noms XML, tels que XSL pour le style, XSLT pour les transformations d'un langage à l'autre, XML Linking Language (Xlink), et bien d'autres.

Le génie de SVG tient autant à son vocabulaire pour décrire les graphiques vectoriels, les images et le textes qu'à sa conformité avec les recommandations XML 1.0, espaces de noms (Namespaces in XML Recommendation), HTML4, et XHTML. Il est également conforme aux spécifications Cascading Style Sheet niveau 2 (CSS2), XSL Transformations (XSLT) Version 1.0 et Document Object Model niveau 2 (DOM2).

SVG est aux graphiques ce qu'est XHTML au texte, CML est la description des molécules chimiques et MathML celle des équations mathématiques. Dans ce chapitre nous allons voir comment SVG peut être utilisé avec XHTML, CML et MathML avec l'aide de XSLT et d'espaces de noms multiples.

Le W3C travaille sur la coexistence de XHTML, MathML et SVG dans un même document en utilisant les espaces de noms XML.
Ceci est expliqué à http://www.w3.org/TR/XHTMLplusMathMLplusSVG/ .

SVG et XHTML

SVG excelle dans la description et le rendu de graphiques 2D. SVG sait faire des choses que fait XHTML et réciproquement XHTML peut faire des choses que fait SVG. Toutefois, la possibilité d'associer ces deux technologies complémentaires est un atout certain.

Grâce aux préfixes des espaces de noms ('namespace'), nous pouvons mêler des éléments qui font partie des spécifications SVG à des éléments qui appartiennent à XHTML. Le préfixe indique au navigateur quel langage utiliser pour interpréter ces éléments et les afficher.

XHTML en bref

Chacun connaît HTML, XHTML est une reformulation de HTML 4 comme une application XML 1.0.

Quels sont les objectifs de XHTML?

XHTML permet la représentation de contenus texte et d'images sur le Web tout en étant:

- extensible.
- compréhensible et simple à générer.

XHTML propose également des méthodes simples d'indexation pour les moteurs de recherche.

Afficher XHTML et SVG

Peu de navigateurs ont un support natif de XHTML. Toutefois Mozilla et Amaya permettent de rendre XHTML et SVG directement. Quelques autres gèrent XHTML comme X-Smiles (java) www.x-smiles.org de l'Université de technologie d'Helsinki.

L'utilisation d'extensions pour rendre XHTML dans une page Web a pour principal désavantage de nécessiter l'utilisations de balises spécifiques (comme <applet> pour les applets java, <embed> pour utiliser un plugin et <object> pour les extensions Microsoft). L'utilisation de ces balises limite le document à une configuration particulière ce qui n'est pas très pratique pour une publication sur le Web.

Cet exemple est un document XHTML contenant un fragment SVG.

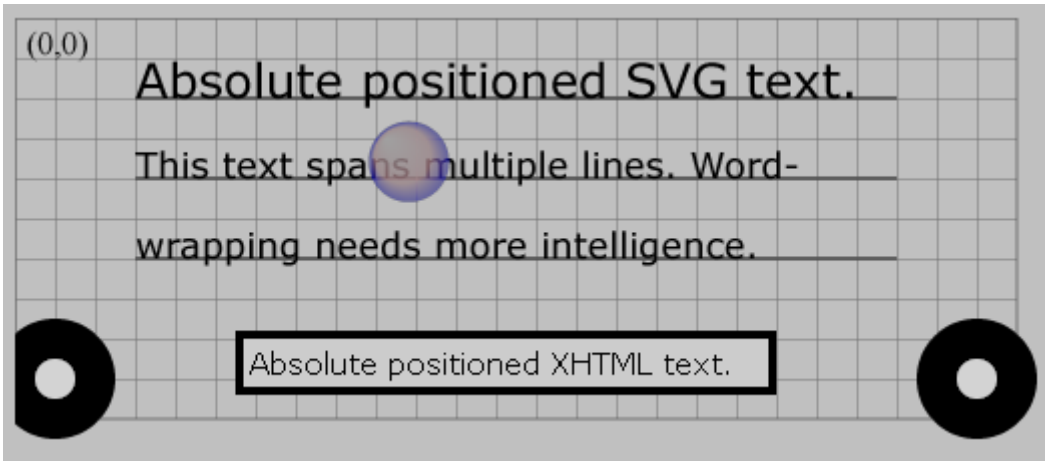

Figure 11-2. Document XHTML/SVG

L'un des avantages de cette approche est de pouvoir accéder et manipuler par script les objets XHTML et le DOM SVG. Voici le code de ce document:

```
<?xml version="1.0" standalone="no"?>
<html xmlns:iSvg="http://www.w3.org/2000/svg"
      xmlns="http://www.w3.org/1999/xhtml">
<head>

</head>
<body>

<div style="font-family:Verdana,Arial,san-serif;font-size:20px">Multi-Namespace
Document in Internet Explorer</div>
<p></p>
<div style="position:absolute; left:120; top:215; width:270px; height:25px;
padding:3px; border:solid; background-color: #CCCCCC; layer-background-color:
#CCCCCC;font-family:Verdana,Arial,san-serif;font-size:15px;">Absolute
positioned XHTML text.</div>

<iSvg:svg id="inlineSVG" width="800" height="600" viewBox="0 0 400 300"
xmlns:svg="http://www.w3.org/2000/svg">
```

```
<iSvg:defs>
      <iSvg:pattern id="gridPattern" width="10" height="10"
patternUnits="userSpaceOnUse">
            <iSvg:path d="M10 0 L0 0 L0 10"
style="fill:none;stroke:rgb(128,128,128);stroke-width:0.25"/>
      </iSvg:pattern>
      <!-- Pattern Gradient 01 -->
      <iSvg:radialGradient                                    id="bubblePattern01"
gradientUnits="objectBoundingBox" fx="20%" fy="20%" style="opacity:0.5;">
            <iSvg:stop offset="0%" style="stop-color:white;" />
            <iSvg:stop offset="70%" style="stop-color:pink;" />
            <iSvg:stop offset="100%" style="stop-color:blue;" />
      </iSvg:radialGradient>
</iSvg:defs>

      <iSvg:rect id="grid" width="100" height="100"
             style="stroke:rgb(128,128,128);stroke-
width:0.25;fill:url(#gridPattern)"/>
      <!-- grid -->
      <iSvg:text x="3" y="9" style="font-size:8">(0,0)</iSvg:text>
      <!-- example -->
      <iSvg:path  d="M30 20 L220 20
            M30 40 L220 40
            M30 60 L220 60"
            fill="none" stroke="black" stroke-width="0.5"/>

        <iSvg:text fill="black">
            <iSvg:tspan x="30" y="20" style="font-size:12;font-
      family:Verdana,serif;">Absolute positioned SVG text.</iSvg:tspan>
            <iSvg:tspan x="30" y="40" style="font-size:9;font-
      family:Verdana,serif;">This text spans multiple lines.  Word-
      </iSvg:tspan>
            <iSvg:tspan x="30" y="60" style="font-size:9;font-
      family:Verdana,serif;">wrapping needs more intelligence.</iSvg:tspan>
        </iSvg:text>

        <iSvg:circle cx="10" cy="90" r="10" stroke="black"
                stroke-width="10" fill="lightgray" />
        <iSvg:circle cx="240" cy="90" r="10" stroke="black"
                stroke-width="10" fill="lightgray" />
      <iSvg:circle id="bubble" cx="50" cy="50" r="10"
            style="fill:url(#bubblePattern01);opacity:0.5;">
            <iSvg:animateTransform attributeName="transform" type="translate"
                        values="0 0;40 40;0 -40;-20 0;-40 40;"
                        additive="sum" dur="20s" repeatDur="indefinite"/>
      </iSvg:circle>
</iSvg:svg>

</body>
</html>
```

Nous allons tester ce document dans différentes configurations.

Amaya

Amaya est un navigateur développé par le W3C pour tester et valider les spécifications à leurs différents stades d'élaboration. La version 6.1 supporte XHTML 1.0, CSS2, et une partie de SVG 1.0. Ces technologies Web sont gérées à travers des espaces de noms dans le document. Et de plus – Amaya n'est pas seulement un navigateur, il est aussi un éditeur spécialement

orienté vers les mathématiques. Nous pouvons ainsi créer et modifier des formules, des équations ou du texte.

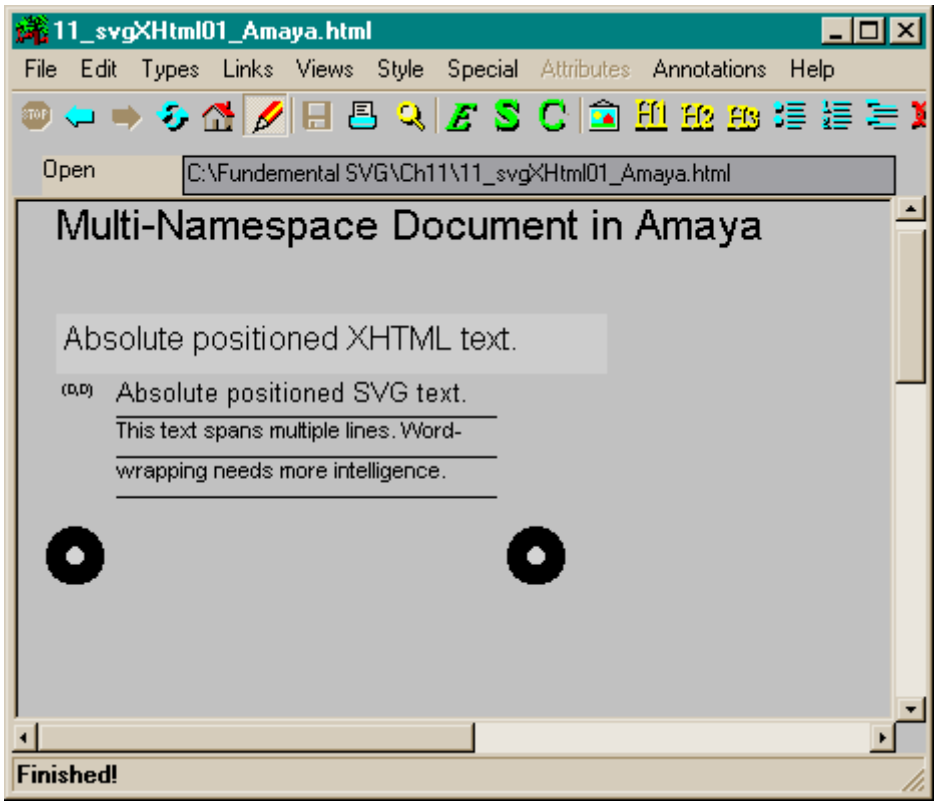

Figure 11-3. Le document dans Amaya

Le manque le plus cruel de cette version est le support du script pour agir sur le DOM. Nous avons des documents statiques, mais devons attendre pour un contenu interactif et dynamique.

Netscape et Mozilla

Netscape (version 7.0) et Mozilla 1.0 supportent les espaces de noms multiples. Mais pour SVG Netscape n'a pas de support natif et contrairement aux versions précédentes ne supporte pas le plugin d'Adobe.

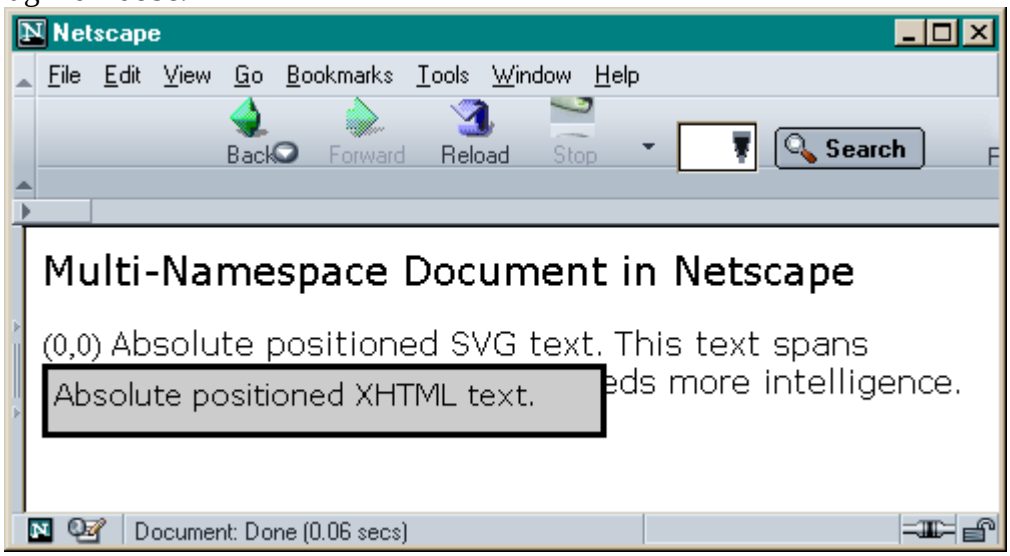

Figure 11-4. Document dans Netscape

Mozilla 1.0 n'a pas non plus de support natif de SVG. Espérons que la situation sera différente dans le futur. Le support de javascript/DOM ne pose pas de problème pour ces deux navigateurs.

Internet Explorer

Internet Explorer de Microsoft n'a pas non plus de support natif pour SVG. Toutefois, nous pouvons utiliser un composant externe grâce à une technologie propriétaire. Le plugin d'Adobe contient l'outil binaire nécessaire.

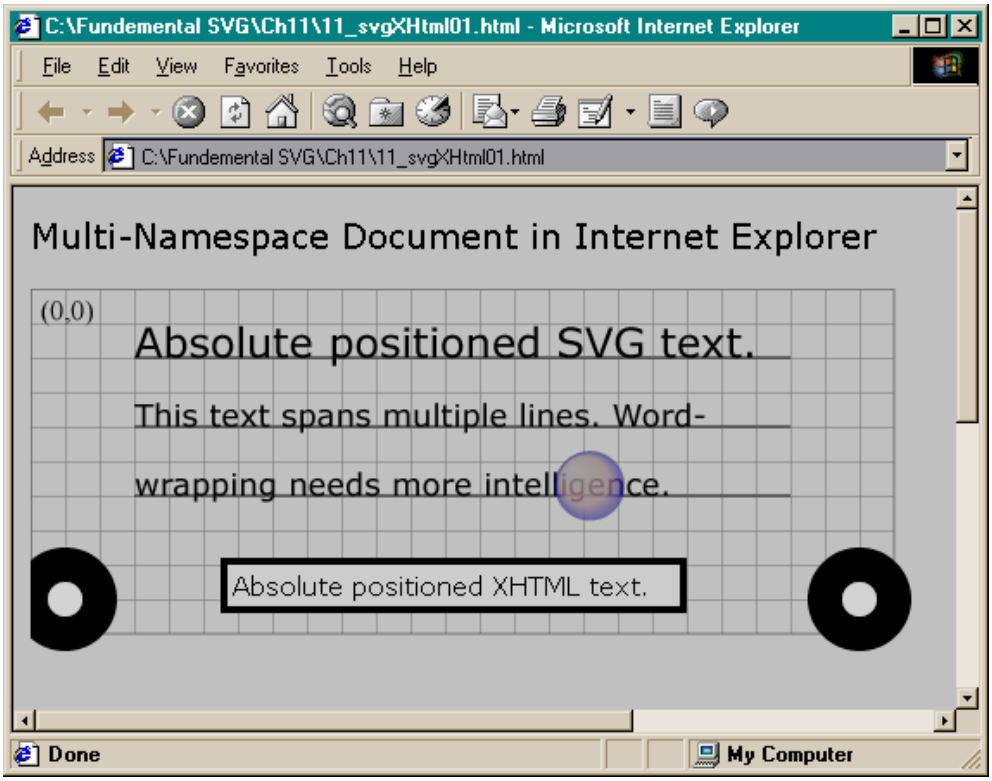

Figure 11-5. Document dans Internet Explorer

Pour Internet Explorer, nous devons cependant modifier notre code:

```
<?xml version="1.0" standalone="no"?>
<html xmlns:iSvg="http://www.w3.org/2000/svg"
      xmlns="http://www.w3.org/1999/xhtml">
<head>

<object id="AdobeSVG" classid="clsid:78156a80-c6a1-4bbf-8e6a-3cd390eeb4e2">
      <img src="rasterSVG.png" border="0"/>
</object>
<?import namespace="iSvg" implementation="#AdobeSVG"?>

</head>
<body>

<div style="font-family:Verdana,Arial,san-serif;font-size:20px">Multi-Namespace
Document in Internet Explorer</div>
<p></p>
<div style="position:absolute; left:120; top:215; width:270px; height:25px;
padding:3px; border:solid; background-color: #CCCCCC; layer-background-color:
#CCCCCC;font-family:Verdana,Arial,san-serif;font-size:15px;">Absolute
positioned XHTML text.</div>
```

```
<iSvg:svg id="inlineSVG" width="800" height="600" viewBox="0 0 400 300"
xmlns:svg="http://www.w3.org/2000/svg">

      … même code …

</iSvg:svg>

</body>
</html>
```

Cependant cet exemple modifié ne fonctionnera plus dans le navigateur de Netscape. Les modifications que nous avons faites rendent notre document non standard et spécifique à Internet Explorer. Si vous ouvrez cet exemple dans IE, vous remarquerez que certaines fonctionnalités utilisables avec le plugin ne le sont plus ici: zoom, pan ou sélection de texte.

Le Web sémantique

Un bénéfice majeur avec les langages basés sur XML est qu'ils peuvent réutiliser des contenus. Ceci est possible car les règles de grammaire et le vocabulaire sont suffisamment robustes pour référencer et interpréter des contenus définis dans d'autres contextes.

Ce qui nous amène au concept de Web sémantique et ses implications, SVG étant alors surtout un langage de description des graphiques de dimension 2. (voir pour plus d'information www.sciam.com/2001/0501issue/0501berners-lee.html).

Communication efficace et sensée en SVG

Pour mieux préciser la notion de Web sémantique, voyons cet adage qui dit que la moitié de ce que nous disons a un sens alors que l'autre moitié n'est là que pour nous faire comprendre la première moitié.

Ceci est vrai avec n'importe quel langage et particulièrement avec SVG. Les données qui nous sont communiquées ne sont souvent qu'une fraction du fichier. Le reste du document n'est là que pour communiquer quelques abstractions au visualiseur. En fait, un langage informatique est souvent plus abstrait et efficace quand il est dans un format binaire. Pourquoi SVG est-il écrit dans un format texte?

Mais nous pouvons en tirer une morale, ce n'est pas parce que quelque chose existe, qu'il doit être compréhensible.

Concepts SVG: la grammaire XML

La syntaxe est très importante en SVG. A travers tous ces exemples, nous retrouvons une similarité de structure. SVG utilise la grammaire XML, et nous avons des règles que nous devons respecter si nous voulons que notre document soit rendu correctement. Chaque langage a sa grammaire et ses règles qui définissent sa structure. XML peut être pensé comme la grammaire qui définit la structure du langage SVG. Si un document SVG ne suit pas ces règles, il ne sera pas formulé correctement et ne sera pas interprété et rendu par le visualiseur.

La première règle de la grammaire XML est que les balises ne peuvent pas être entrecroisées: “ <I></I> ” est correct alors que “<P><I></P></I>” ne l'est pas.
XML est sensible à la casse des caractères (minuscule-majuscule) aussi si nous écrivons “<text>tra la la </Text>”, notre document ne sera pas formulé correctement.

Beaucoup de navigateurs interprétent et affichent des documents HTML incorrects. Le navigateur, fait pour interpréter de tels documents, a fait de HTML un langage facile à utiliser mais le revers de la médaille est que ces navigateurs sont de très grosses applications pour interpréter un langage très pauvrement structuré. Les différences entre les navigateurs ont nécessité beaucoup d'acrobaties de programmation pour avoir un document qui puisse s'ouvrir dans les principaux navigateurs. Avec les langages basés sur XML tel que SVG nous arrivons à un haut degré de conformité aux standards.

Un autre bénéfice avec cette grammaire est que le contenu des balises peut être interprété. Les possibilités de filtrage, de recherche, d'internationalisation sont alors très grandes.

SVG et MathML

Pour un scientifique, il est amer de constater que le Web, créé pour les aider dans leur collaboration, reste très pauvre pour la représentation d'un élément essentiel de leur langage – les équations mathématiques.

Un contenu mathématique est soit écrit en caractères ASCII – ce qui n'est possible que pour des équations très simples – soit présenté sous des formes insatisfaisantes comme bitmap intégré au HTML ou fichier PDF. Dans ce cas ces équations sont des corps étrangers qui ne peuvent être:

- adaptés à la taille de caractère souhaitée par l'utilisateur.
- du même style que le texte voisin.
- recherchés dans la page.
- conçus avec les outils de création de pages Web.
- soumis à des scripts.

D'où une nouvelle idée pour mettre ces équations dans une page HTML. Le support des équations par HTML fut discuté en 1993 avec HTML+ et ajouté à HTML 3.2 en 1996. Mais les navigateurs ne suivirent pas cette évolution. Aussi en 1997 un groupe de travail est constitué par le W3C pour un langage mathématique (mathematical markup language – nommé ***MathML)*** – basé sur XML pour décrire des contenus mathématiques.

En avril 1998, les spécifications *MathML 1.01* deviennent une recommandation du W3C et en 2000 une nouvelle version est prête *MathML 2.0* .

Un des obstacles à l'adoption de ce standard est la popularité de T_EX de Donald Knuth – un système compréhensible et très largement utilisé pour la publication scientifique. Malheureusement, ce langage – bien que basé sur des balises – n'est pas facile à intégrer au modèle XML.

Heureusement, la situation semble évoluer rapidement et les récents efforts du W3C et des sociétés informatiques et de la communauté 'open source' semblent porter quelques fruits.

MathML en résumé

La réponse du W3C à une question primordiale:

Quels sont les buts de MathML ?

Le but principal de MathML est de permettre aux mathématiques d'être créées par les serveurs, reçues et affichées sur le Web, comme HTML le fait pour le texte.

De manière détaillée, MathML doit pouvoir:

- décrire un contenu mathématique de n'importe quel niveau pour l'enseignement et la communnication scientifique.
- faciliter la conversion de et vers d'autres formats de présentation ou sémantiques.
- faire passer des informations pour des applications ou des visualiseurs spécifiques.
- permettre l'extensibilité.
- faciliter l'impression et l'édition.
- être compréhensible (bien qu'il soit très verbeux), et simple à générer et interpréter.

La représentation correcte des équations mathématiques – l'écriture mathématique - est la tâche première de *MathML*. C'est ce que fait T_EX. D'un autre côté MathML doit assurer la sémantique, le sens mathématique des contenus. C'est plus que ne peut faire T_EX. Cet aspect permet l'échange de contenu mathématique entre applications et une possibilité de recherche sur les expressions mathématiques sur le Web.

Les créateurs de *MathML* ont décidé d'une double architecture avec deux jeux d'éléments XML pour la présentation (***presentation markup***) et le contenu (***content markup***).
Nous examinerons ici uniquement l'aspect présentation de *MathML*.

Voici la formule permettant de calculer la longueur d'un vecteur de composantes x et y.

$$r = \sqrt{x^2 + y^2}$$

Cette equation devient en *MathML*

```
<math xmlns='http://www.w3.org/1998/Math/MathML'>
 <mrow>
  <mi>r</mi>
  <mo>=</mo>
  <msqrt>
   <mrow>
    <msup>
     <mi>x</mi>
     <mn>2</mn>
    </msup>
    <mo>+</mo>
    <msup>
     <mi>y</mi>
     <mn>2</mn>
    </msup>
   </mrow>
  </msqrt>
</mrow>
```

```
</math>
```

Les éléments de présentation MathML les plus courants sont **<mi>**, **<mn>** et **<mo>**. Ce sont les seuls éléments qui peuvent contenir des caractères. Chaque nombre, variable ou opérateur doit apparaître dans un de ces éléments.

<mi> est surtout utilisé pour les noms de variables affichés en italique – comme *r, x, y* dans notre exemple.

<mn> contient des nombres affichés en police normale.

<mo> est utilisé pour les opérateurs. Ces opérateurs en MathML compte non seulement '+', '-' et '=', mais aussi les parenthèses, la ponctuation et les accents. La plupart des fonctions mathématiques comme 'sin' ou 'log' sont aussi mises dans l'élément <mo>.

Voici un autre exemple donnant l'angle du vecteur de tout à l'heure avec l'axe horizontal.

$$\psi = \arctan\frac{y}{x}$$

```
<math xmlns='http://www.w3.org/1998/Math/MathML'>
 <mrow>
  <mi>&psi;</mi>
  <mo>=</mo>
  <mo>arctan</mo>
  <mfrac>
   <mi>y</mi>
   <mi>x</mi>
  </mfrac>
 </mrow>
</math>
```

Nous devons pouvoir former des expressions avec ces éléments. MathML renferme des éléments de position, qui sont exclusivement des containers d'éléments, leurs seuls descendants sont des éléments et non des caractères.

<mrow> est le plus commun et important. Il peut avoir autant de descendants que l'on veut qui sont disposés sur une ligne horizontale.

<mfrac> ne peut avoir que deux descendants exactement. Le premier sera le numérateur et le second le dénominateur.

<msqrt> peut avoir un nombre quelconque de descendants qui sont placés sous le signe 'racine de '.

Les éléments **<msub>** et **<msup>** doivent avoir deux descendants le second étant l'indice ou l'exposant du premier. Pour les éléments avec à la fois indice et exposant, MathML permet une écriture contractée avec **<msubsup>** – trois descendants dans l'ordre base, indice et exposant.

$$\mathbf{T} = \begin{pmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

```
<math xmlns='http://www.w3.org/1998/Math/MathML'>
 <mrow>
  <mstyle mathvariant='bold' mathsize='normal'>
    <mi>T</mi>
  </mstyle>
  <mo>=</mo>
  <mrow><mo>(</mo>
   <mrow>
    <mtable>
     <mtr>
      <mtd>
       <mrow><msub><mi>a</mi><mrow><mn>11</mn></mrow></msub></mrow>
      </mtd>
      <mtd>
       <mrow><msub><mi>a</mi><mrow><mn>12</mn></mrow></msub></mrow>
      </mtd>
      <mtd>
       <mrow><msub><mi>t</mi><mi>x</mi></msub></mrow>
      </mtd>
     </mtr>
     <mtr>
      <mtd>
       <mrow><msub><mi>a</mi><mrow><mn>21</mn></mrow></msub></mrow>
      </mtd>
      <mtd>
       <mrow><msub><mi>a</mi><mrow><mn>22</mn></mrow></msub></mrow>
      </mtd>
      <mtd>
       <mrow><msub><mi>t</mi><mi>y</mi></msub></mrow>
      </mtd>
     </mtr>
     <mtr>
       <mtd><mn>0</mn></mtd>
       <mtd><mn>0</mn></mtd>
       <mtd><mn>1</mn></mtd>
     </mtr>
    </mtable>
   </mrow>
   <mo>)</mo>
  </mrow>
 </mrow>
</math>
```

Sur cet exemple, nous avons une structure pour les matrices, une structure très semblable aux tables en HTML.

<mtable> contient un nombre quelconque de lignes, les éléments **<mtr>**. Un élément **<mtr>** contient un nombre quelconque de cellules **<mtd>**, ces éléments **<mtd>** pouvant avoir un nombre quelconque de descendants.

Avec ces quelques éléments MathML, nous pouvons créer un grand nombre d'équations de différents types.

Afficher MathML et SVG

Peu de navigateurs ont un support natif de MathML, et même aucun pour la partie contenu de MathML. Mozilla et Amaya ont un bon support de la partie présentation de MathML, et pour

les autres navigateurs, de nombreuses extensions permettent de rendre un document MathML (en particulier, WebEQ et MathPlayer de Design Science etTechexplorer d'IBM).

Le principal désavantage d'utiliser une extension est que nous avons des balises spécifiques (comme <applet> pour les applets Java, <embed> pour les plugins ou <object> pour les extensions Microsoft). L'utilisation de ces balises limite le document à une seule configuration alors que les documents Web doivent être accessibles à n'importe quelle combinaison système d'exploitation - navigateur.

Pour sortir de ces contradictions le W3C présente une solution basée sur XSLT – la feuille de style *MathML* universelle. La feuille de style transforme le fichier XML en ajoutant les balises nécessaires au rendu dans le navigateur local du contenu MathML et passe ce document modifié au navigateur. Ceci fonctionne bien et sera utilisé pour notre exemple.

This polynomial is the equation of the quadratic Bézier curve - a simple parabolic curve

$$p(t) = (1-t)^2 \cdot p_1 + 2t \cdot (1-t) \cdot p_2 + t^2 \cdot p_3$$

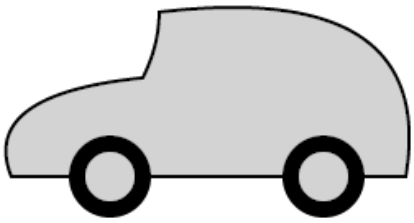

Figure 11-6. Document XHTML/MathML/SVG

Voici le code de cet exemple. C'est un document HTML avec des fragments SVG et XHTML

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet type="text/xsl"
      href="http://www.w3.org/Math/Group/XSL/pmathml.xsl"?>
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>MathML and SVG in Amaya 6.1</title>
 </head>

 <body>
 <h2 style="text-align:center">MathML and
     SVG Support in Amaya 6.1</h2>
 <p> This polynomial is the equation of the quadratic Bézier curve -
     a simple parabolic curve </p>

 <p style="text-align:center">
 <math xmlns='http://www.w3.org/1998/Math/MathML'>
 <mrow>
  <mi>p</mi><mo>(</mo><mi>t</mi><mo>)</mo><mo>=</mo>
  <msup>
   <mrow>
    <mo>(</mo><mn>1</mn><mo>&minus;</mo><mi>t</mi><mo>)</mo>
   </mrow>
   <mn>2</mn>
  </msup>
```

```
  <mo>&sdot;</mo>
  <msub> <mi>p</mi> <mn>1</mn> </msub>
  <mo>+</mo><mn>2</mn><mi>t</mi><mo>&sdot;</mo>
  <mo>(</mo><mn>1</mn><mo>&minus;</mo><mi>t</mi><mo>)</mo>
  <mo>&sdot;</mo>
  <msub> <mi>p</mi> <mn>2</mn> </msub>
  <mo>+</mo>
  <msup> <mi>t</mi> <mn>2</mn> </msup>
  <mo>&sdot;</mo>
  <msub> <mi>p</mi> <mn>3</mn> </msub>
 </mrow>
</math>
</p>

<svg  xmlns="http://www.w3.org/2000/svg">
    <path fill="lightgray" stroke="black" stroke-width="2"
          d="M 100,200
             Q 80,150 180,140
             Q 190,120 190,100
             Q 360,80 340,200
             Z" />
    <circle cx="160" cy="200" r="20" stroke="black"
            stroke-width="10" fill="lightgray" />
    <circle cx="290" cy="200" r="20" stroke="black"
            stroke-width="10" fill="lightgray" />
  </svg>
</body>
</html>
```

Nous allons tester ce document avec différents navigateurs.

Amaya

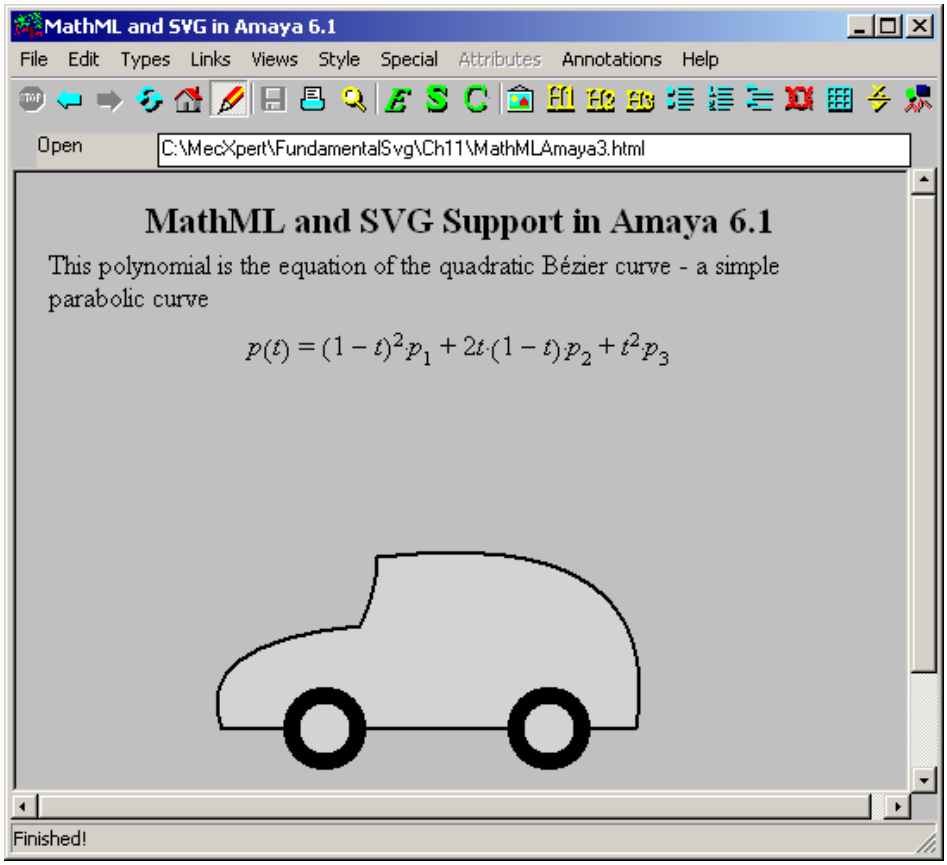

Figure 11-7. Document lu dans Amaya

Netscape and Mozilla

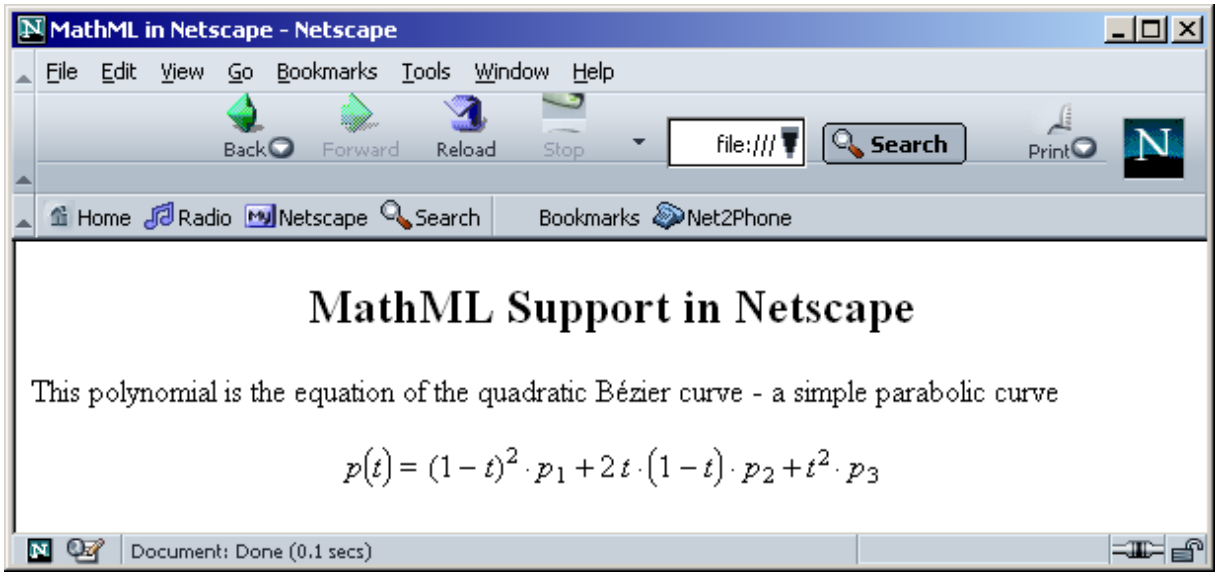

Figure 11-8. Document lu dans Netscape

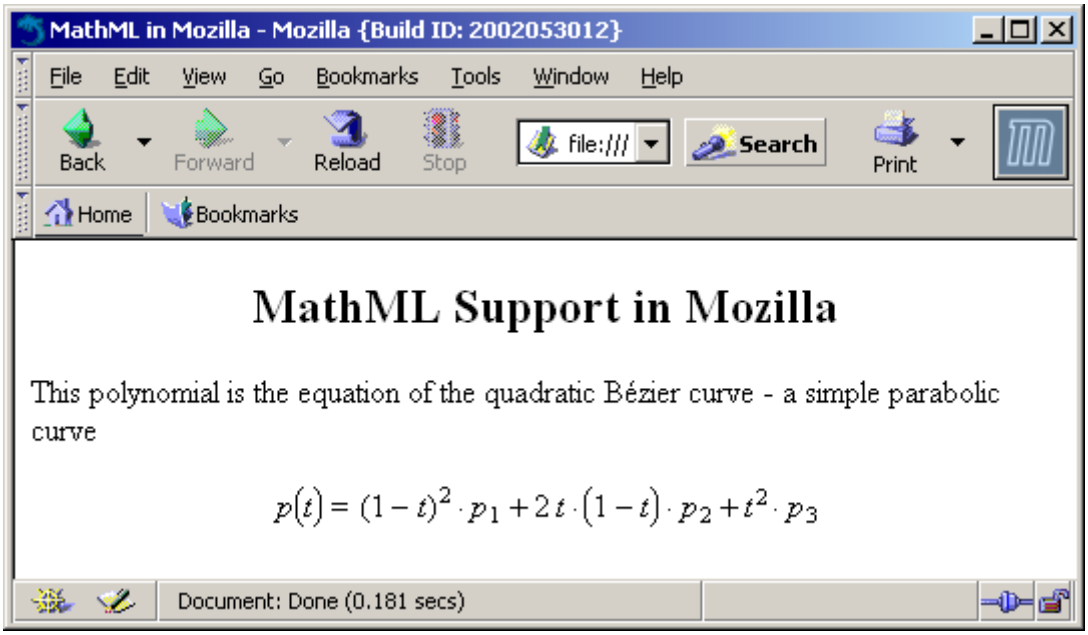

Figure 11-9. Document lu dans Mozilla

La partie MathML est rendue, mais non la partie SVG comme nous l'avons vu précédemment avec l'exemple XHTML et SVG.

Internet Explorer

Internet Explorer est le seul qui n'a de support natif, ni pour MathML, ni pour SVG. Nous utilisons des composants externes:

- Plugin d'Adobe SVGViewer 3.0
- MathPlayer de Design Science

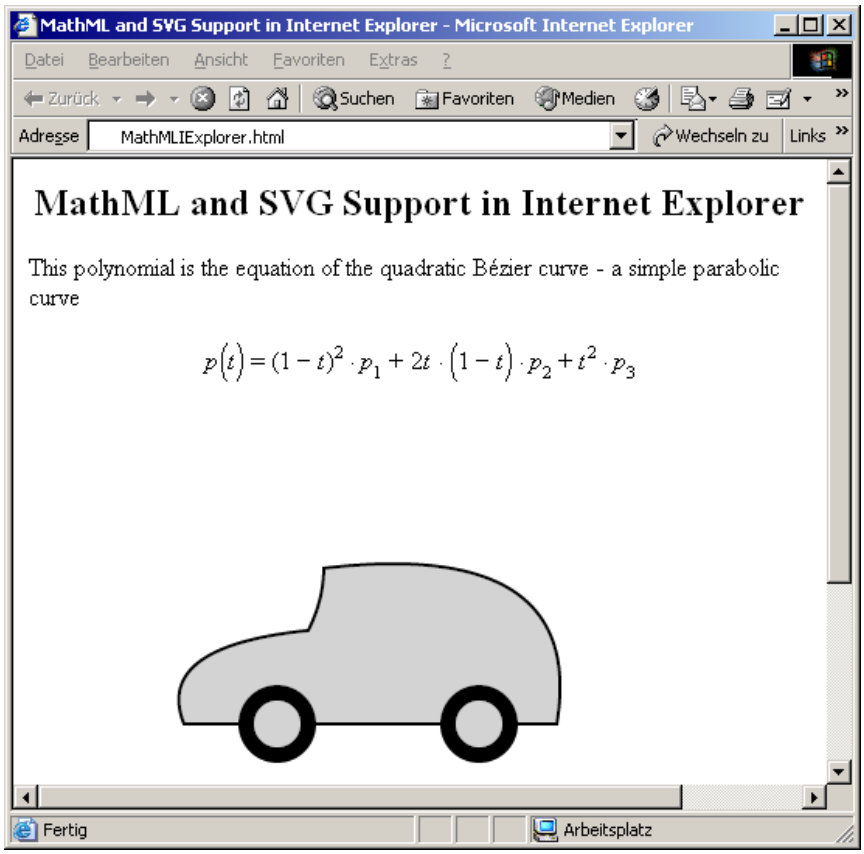

Figure 11-10. Document lu dans Internet Explorer

Nous devons modifier notre document pour l'utilisation des composants externes.

```
<html xmlns:g="http://www.w3.org/2000/svg"
      xmlns:m="http://www.w3.org/1998/Math/MathML">
 <head>
  <title>MathML and SVG Support in Internet Explorer</title>
  <object id="MathPlayer"
          classid="clsid:32f66a20-7614-11d4-bd11-00104bd3f987">
  </object>
  <object id="AdobeSVG"
          classid="clsid:78156a80-c6a1-4bbf-8e6a-3cd390eeb4e2">
  </object>
  <?import namespace="g" implementation="#AdobeSVG"?>
  <?import namespace="m" implementation="#MathPlayer"?>
 </head>
 <body>
  . . .
  <m:math>
   <m:mrow>
   . . .
   </m:mrow>
  </m:math>

  <g:svg width="600" height="400" >
   . . .
  </g:svg>
</body>
</html>
```

Avec ces modifications nous n'avons plus un document standard et non-propriétaire. Par contre nous avons un modèle DOM transparent pour un document XHTML/SVG/MathML interactif et dynamique.