Pengantar *Scalable Vector Graphics* (SVG)

Mohammad Athar Januar

athar@web.de
http://www.stud.uni-karlsruhe.de/~uwdn

BAB 1. KONSEP DASAR SVG DAN KEGUNAANNYA

1.1. Apakah SVG itu ?

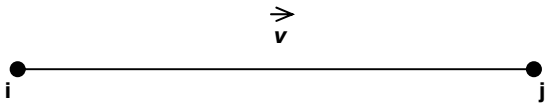
SVG adalah singkatan dari ***S**calable **V**ector **G**raphics* dan merupakan format file baru untuk menampilkan grafik dalam pengembangan web yang berbasis XML (e**X**tensible **M**arkup **L**anguage). Selain SVG, ada juga MathML (**Math**ematic **M**arkup **L**anguage) - berbasis XML- untuk menampilkan rumus-rumus matematika dan juga CML (**C**hemical **M**arkup **L**anguage) untuk kimia.

Gambar 1.1 Senyawa kafein dalam CML

1.2. Fungsi SVG

SVG berfungsi untuk menampilkan grafik 2 dimensional dalam kode XML.
Pada dasarnya, SVG dapat digunakan untuk membuat tiga jenis objek grafik, yaitu :
1. *path* (terdiri dari garis lurus dan kurva),
2. gambar,
3. teks.

SVG dapat mengkreasikan sebuah grafik yang terdiri dari banyak vektor yang berbeda-beda. Sebuah vektor pada dasarnya adalah garis yang menghubungkan dua titik.

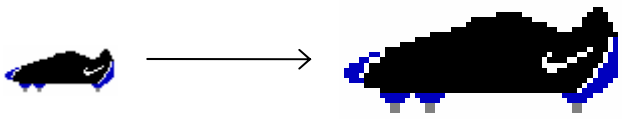


Gambar 1.2 Vektor *v* yang menghubungkan titik i dan j

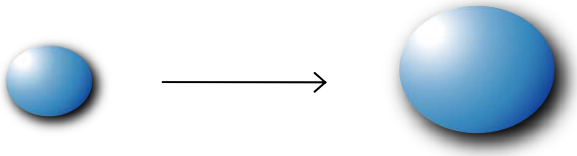
Teknologi baru ini bisa dikembangkan untuk membuat aplikasi-aplikasi web berbasis data yang selalu *update* (contoh : bursa saham, prakiraan cuaca, kurs mata uang) dan interaktif (contoh : *web based training*).

1.3. Kelebihan dan Kekurangannya

Kelebihan SVG yang paling utama adalah gambar tidak akan kehilangan kualitasnya apabila diperbesar atau diperkecil (*scalable*), karena dibuat berdasarkan metode vektor (*vector*), bukan pixel (seperti format grafik pada umumnya, GIF, JPEG dan PNG). Sehingga memungkinkan pengembang web dan juga designer untuk membuat grafik dengan mutu tinggi.



Gambar 1.3 Kualitas yang hilang pada file dengan format .gif apabila diperbesar



Gambar 1.4 Setelah diperbesar, mutu gambar .svg sama sekali tidak berkurang

Walaupun SVG berbasis vektor (dalam artian semua objek dibangun dengan prinsip vektor), SVG ternyata juga dapat dikreasikan untuk efek bayangan, gradasi warna atau juga pencahayaan, Selain itu, animasi juga dapat dikembangkan SVG, sesuatu yang tidak dimiliki oleh GIF, JPEG dan PNG. Hal ini dimungkinkan dengan integrasi DOM (**D**ocument **O**bject **M**odell). Jadi, grafik SVG dapat dianimasikan melalui perintah skript.

Keuntungan SVG lainnya adalah, pengkodeannya yang lumayan ‘familiar’ karena berakar XML.

Pengembang web tidak akan perlu susah payah untuk memulai belajar bahasa baru, karena pada dasarnya SVG adalah pengimplementasian objek vektor dalam web dengan menggunakan XML.

Informasi (vektor) yang disimpan SVG berbentuk teks (dalam XML), bukan *binary code*, ini memiliki keunggulan dalam kecepatan proses *download* karena kecilnya kapasitas file.

Sementara itu, kekurangan SVG terletak pada belumnya semua browser internet dapat membaca data SVG. Untuk itu harus diinstal dulu *plug-in*, yaitu *SVG-Viewer*, *SVG-Viewer* teraktual dikembangkan oleh ADOBE. Browser *Croczilla*, variasi dari *Mozilla-0.9*, sudah mendukung SVG, tanpa perlu menginstal *SVG-Viewer* terlebih dahulu. *Croczilla* sendiri masih dalam tahap awal pengembangannya dan baru hanya bisa membaca kode SVG yang relatif primitif.

1.4. *Tools* Pendukung

Bekerja sama dengan W3C, dan juga perusahaan-perusahaan terdepan di bidang software, ADOBE menjadi pendukung utama pengembangan SVG. Setelah *SVG-Viewer*, perusahaan ini membuat *tools* yang menggenerasikan gambar langsung ke kode SVG, yaitu ADOBE Illustrator, selain itu ada juga ADOBE GoLive untuk mengedit kode SVG. Sementara itu, COREL juga memproduksi software pendukung SVG.

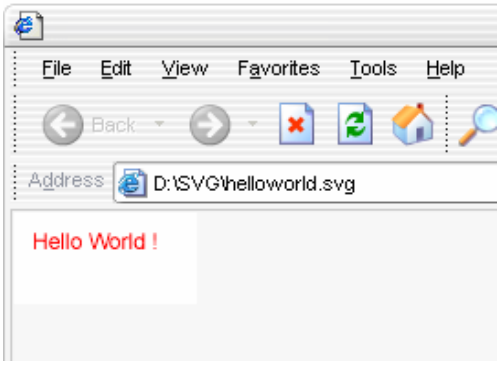
BAB 2. STRUKTUR SVG

2.1. "*Hello World*" Versi SVG

Dengan menggunakan *notepad*, file SVG sudah bisa dibuat. *"Hello World"* dalam SVG memiliki kode seperti di bawah ini :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD svg 20000303 Stylable//EN"
"http://www.w3.org/TR/2000/03/WD-SVG-20000303/DTD/svg-20000303 stylable.dtd">
<svg width="100px" height="50px">
<text style="fill:red" x ="10" y="20">Hello World !</text>
</svg>
```

Setelah disimpan dalam format .svg, file tersebut dapat dibuka oleh browser (dalam contoh berikut menggunakan *Internet Explorer*). Tampilannya akan terlihat sebagai berikut :



Gambar 2.1 Program "*Hello World*" Pertama Dalam SVG

Selanjutnya kita akan mempelajari kode SVG ini.

Baris pertama menjelaskan versi XML yang digunakan. *Tag* kedua `<DOCTYPE>` menyediakan referensi untuk DTD (**D**ocument **T**ype **D**efinition). DTD ini mendefinisikan semua *tag* XML yang akan digunakan untuk membuat file SVG, seperti atribut *tag*, element, dan memspesifikasikan bentuk dokumen dan struktur yang berlaku. Baris ketiga masih termasuk dalam *tag* `<DOCTYPE>` , menginformasikan alamat *http* (**H**yper**T**ext **T**ransfer **P**rotocol), akan tetapi alamat dituju ke *SVG-Viewer*.

DTD yang akan digunakan sangat tergantung pada *SVG-Viewer*. Sebagai contoh, untuk *SVG-Viewer 1.0*, *tag* `<DOCTYPE>` akan berkode :

```
<!DOCTYPE svg PUBLIC "-//W3C/DTD svg 20000303 Stylable/EN"
"http://www.w3.org/TR/2000/03/WD-SVG-20000303/DTD/svg-20000303-stylable.dtd">
```

Sementara untuk versi 2.0 :

```
<!DOCTYPE svg PUBLIC "-//W3C/DTD svg 20001102 Stylable/EN"
"http://www.w3.org/TR/2000/CR-SVG-20001102/DTD/svg-20001102.dtd">
```

Dan tentu saja untuk versi terbaru *SVG-Viewer* ada *tag* `<DOCTYPE>` lainnya.
Isi *tag* ini adalah tanggal dan status dari standar SVG yang digunakan. Seperti 20000303 berarti DTD ini mempunyai dokumen tanggal 03.03.2000.

Selanjutnya *tag* `<svg>`, di sini kita sudah bisa mendefinisikan atribut yang diinginkan. Di contoh, kita mendefinisikan ukuran SVG, yaitu `width="100px" height="50px"`, berarti lebarnya 100 pixel dan tingginya 50 pixel. Selain satuan pixel, bisa juga digunakan ukuran cm (centimeter), mm (milimeter), bahkan in (inchi) sebagai satuan absolut.

Kemudian *tag* `<text>`, *tag* ini memiliki atribut style yang kita definisikan `"fill:red"`. Dengan menggunakan atribut ini kita ingin agar tulisan yang ditampilkan berisi (*fill*) warna merah. Selain itu ada koordinat `x="10" y="20"`, atribut ini menginformasikan di mana akan dimulai teks tersebut ditampilkan.

2.2. *Syntax* SVG

Sebelum kita memulai 'bermain' dengan kode SVG, sangat penting sekali untuk mengetahui 'aturan mainnya' (syntax) terlebih dahulu.
- SVG sangat memperhatikan sistem penulisan. Semua *tag*, atribut dan nilai atribut ditulis dengan huruf kecil
- Semua *tag* harus ditutup. Untuk *tag*, seperti `<text>`, yang diluarnya dapat ditulis sesuatu, akan ditutup dengan *tag* pasangannya `</text>`. Sementara itu untuk *tag* yang diluarnya tidak dapat ditulis apa-apa akan ditutup dengan `</>`, seperti `<rect.../>`.
- Komentar memiliki kode yang sama seperti HTML `<!--` dan `-->`.
- Untuk memposisikan sebuah elemen digunakan atribut x dan y, bukan top atau left seperti HTML.
- Semua atribut dimulai dan diakhiri dengan tanda kutip " ... ".

2.3. *Tags* SVG

Tag SVG dibedakan menjadi 4 macam :
1. Element grafik (*graphics elements*) :
   `<path>` `<text>` `<rect>`
   `<circle>` `<ellipse>` `<line>`
   `<polyline>` `<polygon>` `<image>`
   `<use>`

2. Element container (*container elements*) :
   `<svg>` `<g>` `<defs>`
   `<symbol>` `<clipPath>` `<mask>`
   `<pattern>` `<marker>` `<a>`
   `<switch>`

3. Element referensi untuk grafik (*graphics referencing elements*):
   `<use>` `<image>`

4. Element untuk mendesign teks (*text content elements*):
   `<text>` `<tspan>` `<texPath>`
   `<tref>` `<altGlyph>`

Sedangkan contoh *tags* yang penting :

`<text>` : Mendefinisikan sebuah teks,
`<rect>` : Mendefinisikan sebuah persegi,
`<path>` : Mendefinisikan sebuah *path*,
`<filter>` : Mendefinisikan filter atau bahkan kombinasinya.
`<g>` : Membuat sebuah grup yang terdiri dari elemen-elemen.
`<a>` : Mendefinisikan sebuah *hyperlink*,
`<line>` : Mendefinisikan sebuah garis

BAB 3. *SHAPES*

3.1. *Tags* untuk *Shapes*

Tags yang tersedia untuk menggambar sebagai berikut :
1. <rect> : untuk persegi empat (***rect****angle*),
2. <circle> : untuk lingkaran,
3. <ellipse> : untuk elips,
4. <line> : untuk garis,
5. <polyline> : untuk garis banyak,
6. <polygone> : untuk poligon.

Sebenarnya, untuk *SVG-Viewer* teraktual (versi 3.0) sudah didefinisikan lagi berbagai macam *tags* yang lainnya, seperti untuk spiral, bintang, segilima (*pentagon*), dsb.

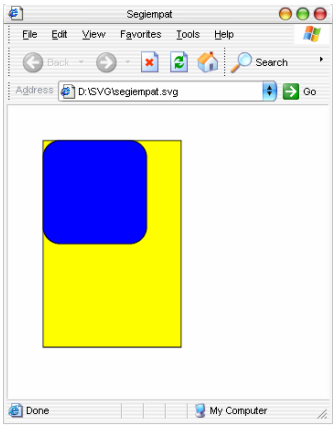
3.2. Segiempat (rectangle)

Dengan menggunakan *tag* `<rect>` kita akan memulai menggambar SVG.

Contoh kode :
```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
"http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="800" height="600">
<title>Segiempat</title>
<g id="contoh_rect" stroke="black" stroke-width="1pt">
<!-- Segiempat berwarna kuning -->
<rect x="1cm" y="1cm" width="4cm" height="6cm" fill="yellow" />
<!-- Segiempat berwarna biru -->
<rect x="1cm" y="1cm" width="3cm" height="3cm" rx=".5cm" ry=".5cm" fill="blue" />
</g>
</svg>
```

Simpan file tersebut dengan format *.svg*. Sekedar informasi, ada juga file *SVG* dengan tipe *.svgz*, ini hanyalah format SVG yang dikompres.

Tampilannya :



Gambar 3.1 Segiempat dengan SVG

Apabila file tersebut tidak menampilkan apa-apa, ini berarti *browser* yang Anda gunakan belum dilengkapi *SVG-Viewer*. Untuk menginstalnya, silahkan ke
http://www.adobe.com/svg/viewer/install/main.html
dan cari *SVG-Viewer* tersebut.

Penjelasan untuk masing-masing atribut :
- `x` dan `y` : untuk mendefinisikan koordinat, dimana segiempat tersebut akan dimulai digambar.
- `width` dan `height` : atribut wajib untuk ditulis, menjelaskan berapa panjang (*width*) dan tinggi (*height*) segiempat tersebut.
- `rx` dan `ry` : atribut tambahan, bila ingin menggambar segiempat dengan pojok-pojok yang melengkung (*rounded rectangles*).

Dengan merubah-rubah kode di atas, dengan sendirinya Anda akan lebih mengerti fungsi masing-masing atribut.

3.3. Lingkaran

Untuk lingkaran kita menggunakan *tag* `<circle>`.

Contoh kode :
```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
"http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="800" height="600">
<title>Lingkaran</title>
<g id="contoh_circle">
<!-- Lingkaran berwarna kuning -->
<circle cx="100" cy="100" r="60" fill="yellow" />
<!-- Lingkaran berwarna merah -->
<circle cx="100" cy="100" r="30" fill="red" />
</g>
</svg>
```

Tampilannya :

Gambar 3.2 Lingkaran dengan SVG

Penjelasan untuk masing-masing atribut :

- `cx` dan `cy` : untuk mendefinisikan koordinat titik pusat lingkaran.
- `r` : untuk mendefinisikan jari-jari (**r**adius) lingkaran.

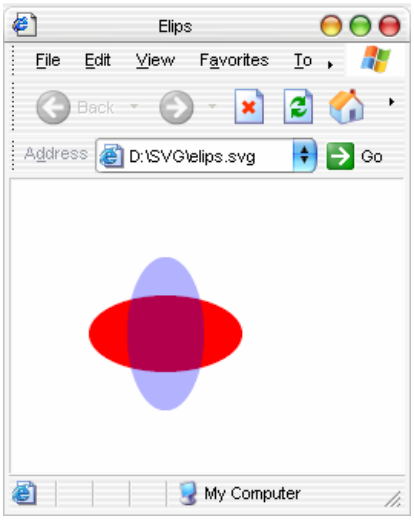
3.4. Elips

Elips adalah lingkaran yang memiliki 2 jari-jari yang berbeda, untuk itu dibutuhkan *tag* baru yang memiliki atribut tambahan.

Contoh kode :

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
"http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="800" height="600">
<title>Elips</title>
<g id="contoh_ellipse">
<!-- Elips berwarna merah -->
<ellipse cx="100" cy="100" rx="50" ry="25" fill="red" />
<!-- Elips berwarna biru -->
<ellipse cx="100" cy="100" rx="25" ry="50" fill="blue" fill-opacity=".3" />
</g>
</svg>
```

Tampilannya :



Gambar 3.3 Elips dengan SVG

Penjelasan untuk masing-masing atribut :

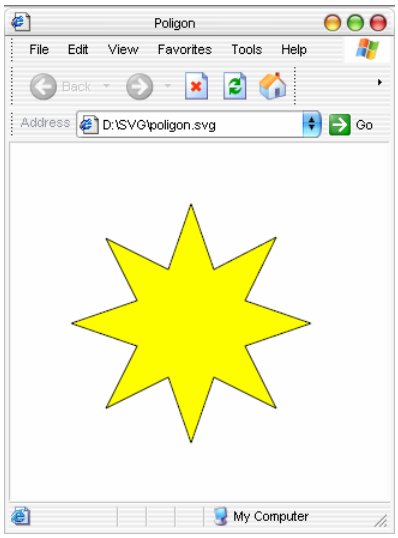
- `rx` dan `ry` : untuk mendefinisikan radius horizontal (sumbu **x**) dan radius vertikal (sumbu **y**) elips.
- `opacity` : untuk mendefinisikan tingkat transparan sebuah objek (nilainya dari 0.0 sampai 1.0).

3.5. Poligon

Contoh kode :

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
"http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="800" height="600">
<title>Poligon</title>
<g id="contoh_polygon">
<!-- Poligon berwarna kuning dengan pinggiran berwarna hitam -->
<polygon stroke="black" fill="yellow"
points="150, 50 169,105 221, 78
195,131 250,150 195,169
221,221 169,195 150,250
131,195 79,221 105,169
50,150 105,131 79, 79
131,105" />
</g>
</svg>
```

Tampilannya :



Gambar 3.4 Poligon dengan SVG

Penjelasan untuk masing-masing atribut :

- `stroke` : untuk mendefinisikan warna garis pinggir poligon.
- `points` : untuk mendefinisikan koordinat titik-titik untuk ditarik sebuah garis diantaranya.

BAB 4. *PATHS*

4.1. Definisi *Path*

Di samping bentuk standar yang telah kita pelajari di bab 3, SVG juga bisa digunakan untuk mengkreasikan berbagai macam bentuk seperti segitiga, setengah lingkaran dan bentuk-bentuk tak beraturan lainnya.

Path memiliki konsep menghubungkan titik ke titik lainnya. Konsep ini dapat diperluas untuk menggambar kurva-kurva atau form-form yang sangat kompleks. *Path* juga dapat digunakan untuk membuat animasi dan bahkan untuk mengkreasikan teks.

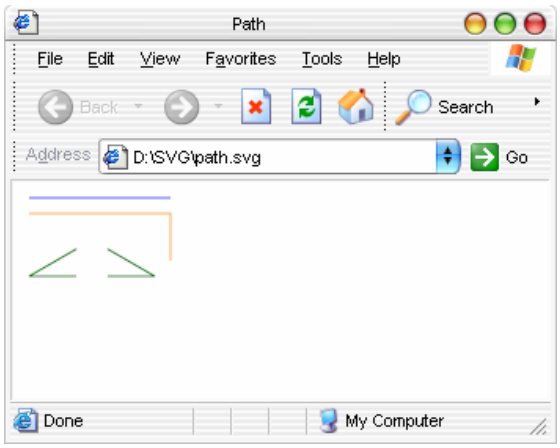
4.2. *Path* Pertama

Sekarang kita akan membuat program *path* pertama.

Contoh kode :

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
"http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="800" height="600">
<title>Path</title>
<g id="contoh_path" fill="none">
<!-- Garis berwarna biru -->
<path d ="M10,10 L100,10" stroke ="blue" />
<!—Garis berwarna oranye -->
<path d ="M10,20 L100,20 L100,50" stroke ="orange" />
<!-- Garis berwarna hijau -->
<path d ="M40,60 L10,60 L40,42.68 M60,60 L90,60 L60,42.68" stroke ="green"/>
</g>
</svg>
```

Tampilannya :



Gambar 4.1 *Paths* dengan SVG

Penjelasan dari kode :

```
<!-- Garis berwarna hijau -->
<path d ="M40,60 L10,60 L40,42.68 M60,60 L90,60 L60,42.68" stroke ="green"/>
```

`M40, 60` : menggerakkan pensil ke titik (40, 60).
`L10, 60` : menarik sebuah garis ke titik (10, 60).
`L40, 42.68` : menarik sebuah garis ke titik (40, 42.68).
`M60, 60` : (membuat *path* baru) menggerakkan pensil ke titik (60, 60).
`L90, 60` : menarik sebuah garis ke titik (90, 60).
`L60, 42.68` : menarik sebuah garis ke titik (60, 42.68).
`M[x][y]` : mendefinisikan koordinat titik awal (***m****oveto*).
`L[x][y]` : menggambar garis ke titik `[x][y]` (***l****ineto*).

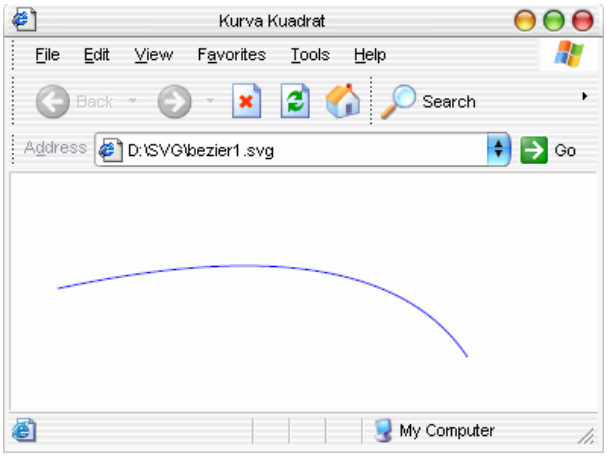
4.3. Kurva BÉZIER Kuadrat

Kurva BÉZIER tersederhana adalah kurva kuadratnya. Hal yang perlu dispesifikasikan adalah koordinat titik awal, titik akhir dan titik kontrol (*control point*).

Contoh kode :

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
"http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="800" height="600">
<title>Kurva Kuadrat</title>
<g id="contoh_bezier" fill="none">
<!-- Kurva berwarna biru -->
<path d ="M30,75 Q240,30 300,120" stroke ="blue" />
</g>
</svg>
```

Tampilannya :



Gambar 4.2 Kurva BÉZIER kuadrat dalam SVG

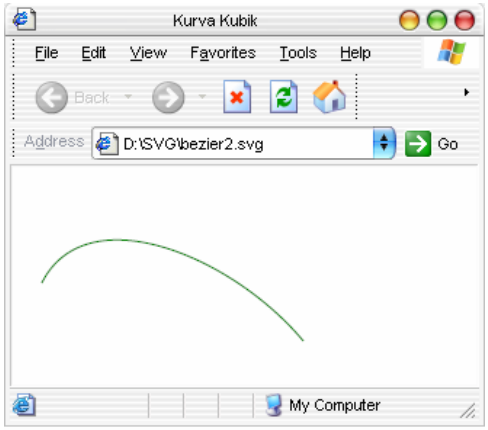
Penjelasan dari kode :
`M30,75` : menggerakkan pensil ke titik (30, 75).
`Q240,30` : mendefinisikan koordinat titik kontrol (240, 30).
`300,120` : menggambar kurva sampai titik (300, 120).
Jadi kurva digambar dari titik (30, 75) sampai titik (300, 120) dengan titik kontrol (240, 30). `Q` sendiri digunakan untuk menggambar kurva BÉZIER kuadrat (***q****uadrate*).

4.4. Kurva BÉZIER Kubik

Perbedaan antara kurva BÉZIER kubik dan kuadrat terletak pada banyaknya titik kontrol (*control point*). Kurva BÉZIER kuadrat, seperti sudah dibahas sebelumnya (bab 4.3), hanya memiliki satu titik kontrol, sedangkan kubik mempunyai dua titik untuk setiap titik akhir.
Akan tetapi, pada dasarnya kedua kurva ini memiliki prinsip pengkodean yang sama.

Contoh kode :
```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
"http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="800" height="600">
<title>Kurva Kubik</title>
<g id="contoh_bezier" fill="none">
<!-- Kurva berwarna hijau -->
<path d ="M20,80 C50,20 150,60 200,120" stroke ="green" />
</g>
</svg>
```

Tampilannya :



Gambar 4.3 Kurva BÉZIER kubik dalam SVG

Penjelasan dari kode :
`M20,80` : menggerakkan pensil ke titik (20, 80).
`C50,20` : mendefinisikan koordinat titik kontrol pertama (50, 20).
`150,60` : titik kontrol kedua (150, 500).
`200,120` : menggambar kurva sampai titik (200, 120).
Jadi kurva digambar dari titik (20, 80) sampai titik (200, 120) dengan dua titik kontrol (50, 20) dan (150,60). `C` berfungsi untuk menggambar kurva BÉZIER kubik (***c****ubic*).

4.5. Referensi Kurva BÉZIER Kuadrat dan Kubik

Jadi, untuk menggambar kurva kuadrat digunakan perintah `Q` yang memiliki argumen `[x1][y1][x][y]`. Berarti kita harus mendefinisikan titik kontrol `[x1][y1]` dan melanjutkan penggambaran kurva dari titik awal hingga titik akhir `[x][y]`.

Sementara itu untuk kubik digunakan perintah `C` yang memiliki argumen `[x1][y1][x2][y2][x][y]`. Seperti yang telah dijelasakan sebelumnya, kurva BÉZIER kubik memiliki dua titik kontrol, dalam hal ini titik pertama `[x1][y1]` dan kedua `[x2][y2]`.

BAB 5. ANIMASI

5.1. *Tags* untuk Animasi

Selain digunakan untuk mengkreasikan form dan kurva, SVG juga bisa membuat animasi. Dan ini merupakan salah satu keunggulan SVG.

Tags yang digunakan untuk animasi :
1. `<animate>` : untuk memulai penganimasian.
2. `<animateMotion>` : untuk animasi gerakan sepanjang sebuah *path*.
3. `<animateColor>` : untuk animasi perubahan warna.
4. `<animateTransform>` : untuk animasi sebuah transformasi.
5. `<mpath>` : untuk referensi sebuah *path* dari gerakan `<animateMotion>`.

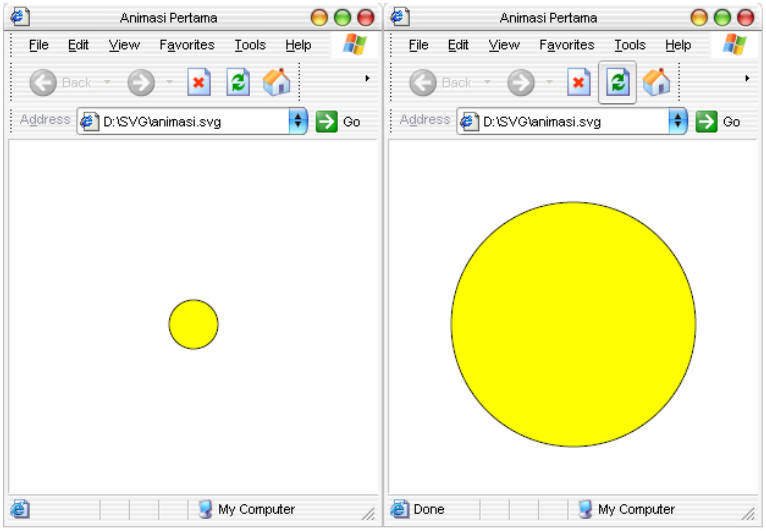
Contoh-contoh yang dapat dianimasikan dari sebuah ‘aktor’:
- ukuran,
- kelihatan,
- posisi dan gerakan,
- warna,
- transformasi.

5.2. Animasi Pertama

Sekarang kita akan membuat animasi sebuah lingkaran yang akan membesar dari detik ke-0 hingga ke-3 (berdurasi 3 detik).

Contoh kode :
```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
"http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="300" height="300">
<title>Animasi Pertama</title>
<g id="contoh_animation">
<!-- Lingkaran berwarna kuning dengan radius 20-->
<circle cx="150" cy="150" r="20" style="fill:yellow; stroke:black">
<!-- Animasi mulai dari detik ke-0 berdurasi 3 detik, memperbesar radius lingkaran
dari 20 hingga 100 -->
<animate attributeName="r" begin="0s" dur="3s" from="20" to="100" fill="freeze" />
</circle>
</g>
</svg>
```

Tampilannya :



Gambar 5.1 Animasi lingkaran dari radius 20 pixel hingga 100 pixel berdurasi 3 detik.

Penjelasan dari kode :

```
<animate attributeName="r" begin="0s" dur="3s" from="20" to="100" fill="freeze" />
```

Sebenarnya hanya dengan membaca kode di atas, kita sudah dapat bayangan dari animasi tersebut. Pertama dengan `attributeName="r"` kita ingin r (radius) dianimasikan dengan mulai di detik ke-0 dan berdurasi 3 detik (`begin="0s" dur="3s"`). Kemudian ukurannya kita definisikan dari 20 pixel hingga 100 pixel (`from="20" to="100"`). Terakhir dengan `fill="freeze"`, kita ingin keadaan objek lingkaran setelah durasi menjadi 'beku' (*freeze*).

5.3. Komponen Dasar Sebuah Animasi

Hampir semua *tags* animasi mengharuskan penulisan atribut : `attributeName` , karena dengan ini kita menspesifikasikan karakter 'aktor'(dalam hal ini atribut sebuah objek) apa saja yang akan kita animasikan. Selain itu, ada beberapa hal yang harus diperhatikan dalam pembuatan animasi. Yaitu :

- karakter aktor yang akan dianimasikan,
- durasi waktu,
- keadaan (skenario) selama animasi,
- keadaan (skenario) setelah animasi,
- start baru untuk animasi,
- pengulangan animasi.

BAB 6. MASA DEPAN SVG

Melihat realita sekarang, SVG adalah teknologi yang sedikit kurang populer. Teknologi seperti *Flash* sudah terlampau kuat untuk dikalahkan oleh SVG.

Dengan segala kelebihan dan kekurangan, SVG memberikan sebuah alternatif baru untuk penganimasian objek dan juga pembuatan web yang dinamis dan interaktif. Mengingat umur SVG yang relatif muda (lahir tahun 1999) tentu saja masih banyak penambalan di sana-sini untuk menuju ke arah yang memadai dan belum saatnya untuk mengatakan SVG tidak akan berkembang. Pada dasarnya, SVG adalah teknologi baru yang sangat menguntungkan dari segi keinteraktifan dan kualiatasnya. Jadi, lebih baik kita tunggu saja perkembangannya.

Referensi

[Helm Spona, 2001] "Das Einstiegerseminar", *SVG Webgrafiken mit XML*, Verlag moderne industrie Buch AG & Co., 2001.
[Iris Fibinger, 2002] "SVG-Scalable Vector Graphics", *Praxiswegweiser und Referenz für den neuen Vektorgrafikstandard*, Markt+Technik Verlag, 2002.
[J. David Eisenberg, 2002] "SVG Essentials", *Producing Scalable Vector Graphics with XML*, O'Reilly & Associates Inc, 2002.
[Kurt Cagle, 2002] "SVG Programming", *The Graphical Web*, Appres, 2002.

Biografi Penulis

Mohammad Athar Januar. Angkatan 1999 SMUN 78 Jakarta. Sekarang sedang kuliah *computer sciences* program diplom di universitas Karlsruhe, Jerman. Magang di sebuah institut kampus, sebagai asisten peneliti. Hobby olahraga, dengerin musik, nonton film dan pokoknya yang asik-asik. Selain itu juga aktif menyumbangkan pikiran dan ide di perusahaan IT consulting CIC, Bogor.