SCALABLE VECTOR GRAPHICS

Esercitazione N.1 -- FONTI:

Scalable Vector Graphics (SVG) Tutorial

Ivan Herman, W3C Head of Offices,

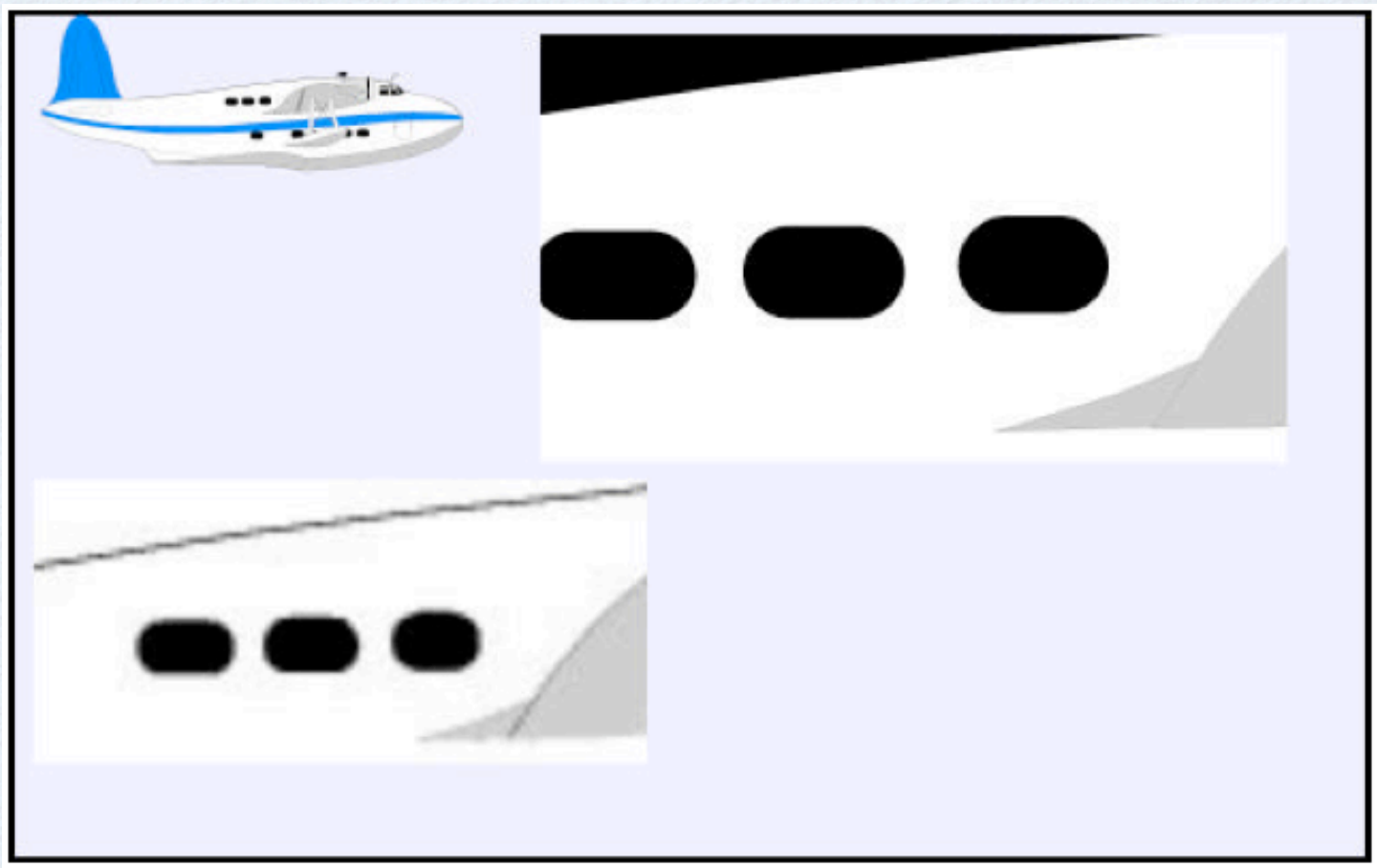
SVG Tutorial

David Duce*, Ivan Herman+, Bob Hopgood*

*Oxford Brookes University, +World Wide Web Consortium

- Images are not Computer Graphics
- The Oxford Dictionary of Computing has the following definition of computer graphics: the creation, manipulation of, analysis of, and interaction with pictorial representations of objects and data using computers.
- A digital image is a 2-dimensional regular grid of pixels.

- The ability to interact with it is limited.
- Being an array of pixels, most of the information that existed in the original object is lost.



- The image formats all share many disadvantages that are serious obstacles to the development and adoption of new technologies on the Web.
- Some of the major problems are listed below.

- **Bandwidth** Images are large. Improvements in network bandwidth have helped to hide this.
- **Flexibility**. Images inherently have a fixed resolution. Colour, resolution, aspect ratio and bandwidth often differ significantly between devices.

- **Hyperlinking**. Hyperlinking is a fundamental requirement on the Web. However, to link to different places, dependent on where the user clicks on an image, is not simple.
- **Animation and Interaction**. Many applications profit from the use of animation and interaction (cartography, CAD, remote teaching, etc). Interaction is limited to the use of image maps.
- **Separation** of Style from Content.

- Using an image format for vector graphics has some major major drawbacks:

- **Image size:**
- **Fixed resolution**
- **Binary format**
- **Minimal animation**
- **No inherent hyperlinking**

- <text x=”20” y=”20”>Abracadabra</text>
- The **text element** has a start and end tag written as <text> and </text> and the content of the element is the string Abracadabra. The text element has two **attributes**, x and y. These are defined as part of the start tag.

SVG is an application

- This has the benefit that the overall syntactic structure of SVG is known and parsers exist to handle it.
- It also means that SVG can benefit from the other activities within W3C concerned with the XML Family of standards.

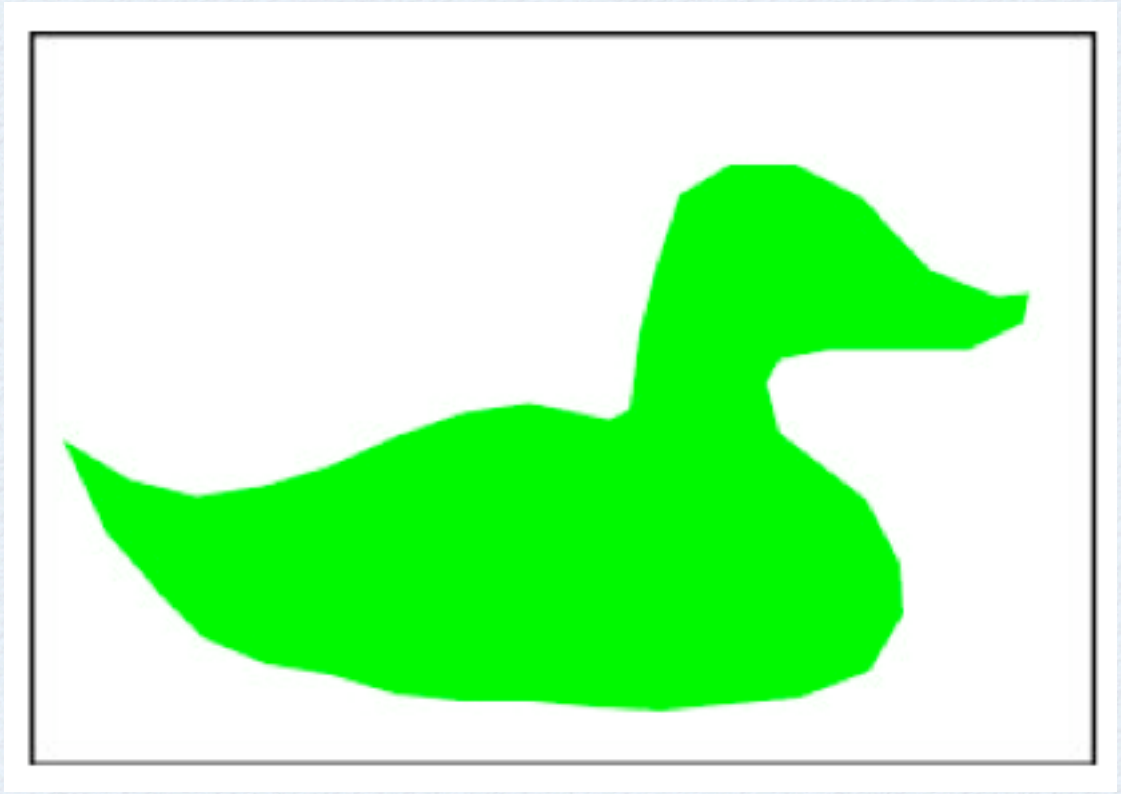
- **Being an XML application, several rules have to be obeyed:**
- There can only be **one outer element** (the root element) that encloses the complete drawing definition.
- **Every start** tag must have a **correctly nested end** tag
- The form of the **start and end tags must be identical**. If the start tag is upper case so must be the end tag.

- Attributes must be enclosed in quotes (either single or double) If the content of the element is null, a shorthand can be used:

- <rect x=”10” y=”10” width=”50” height=”30”></rect>

- <rect x=”10” y=”10” width=”50” height=”30” />

```
<?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="320" height="220">
<rect width="320" height="220" fill="white" stroke="black" />
<g transform="translate(10 10)">
<g stroke="none" fill="lime">
<path d="M 0 112
L 20 124 L 40 129 L 60 126 L 80 120 L 100 111 L 120 104
L 140 101 L 164 106 L 170 103 L 173 80 L 178 60 L 185 39
L 200 30 L 220 30 L 240 40 L 260 61 L 280 69 L 290 68
L 288 77 L 272 85 L 250 85 L 230 85 L 215 88 L 211 95
L 215 110 L 228 120 L 241 130 L 251 149 L 252 164 L 242 181
L 221 189 L 200 191 L 180 193 L 160 192 L 140 190 L 120 190
L 100 188 L 80 182 L 61 179 L 42 171 L 30 159 L 13 140Z"/>
</g>
</g>
</svg>
```



SVG has a hierarchical structure

- The **g element** in the example groups a set of elements.
- In this case there is just a single path element but normally there would be a **sequence** of drawing elements making up an object.
- **Attributes** can be defined on the g element that **apply** to the whole group.
- The hierarchical structure in SVG is similar to the **scene graph** approach used in systems like **OpenInventor**, PostScript and most graphics editors

- The simplest way to use SVG is to open an SVG file with an **SVG-enabled web browser** (either via an SVG plug-in or providing local support).

- An **SVG diagram** can be incorporated into a web page defined in HTML.

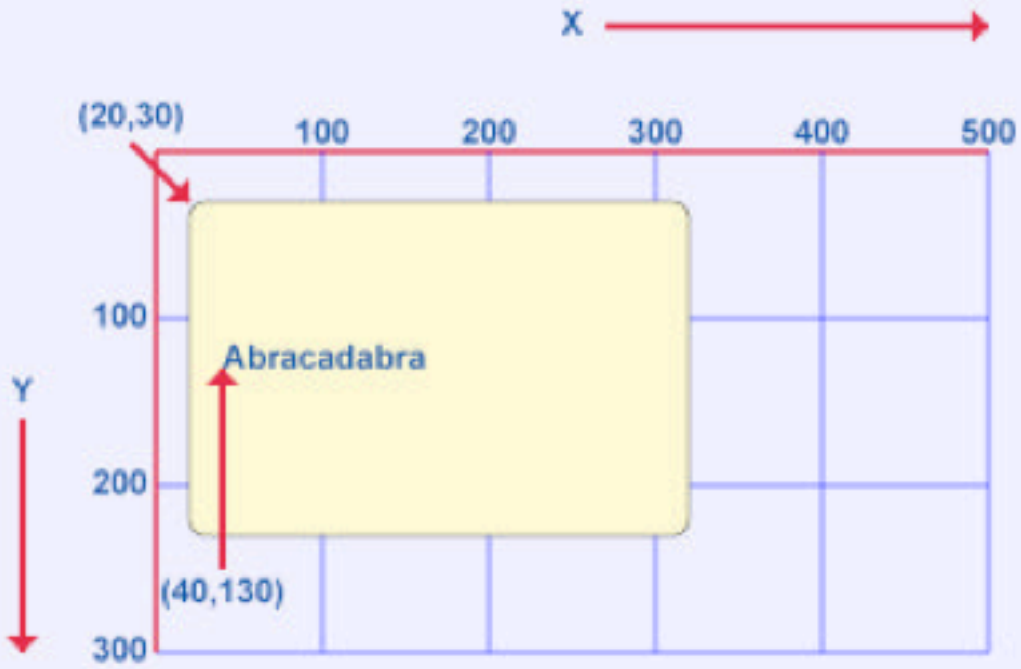
- The SVG document is defined and stored in a **file with '.svg'** as the file extension.

```
<p>This can be shown in the following diagram:</p>
<object width="320" height="220" data="myfirstsvg.svg" type="image/svg+xml">
Please download Adobe Plug-in to see SVG diagram </object>
```

Rectangles and Text

- It is difficult to talk about either coordinates or rendering in a vacuum
- so we first need to specify two SVG drawing elements so that we can illustrate the points being made.

```
<rect x="20" y="30" width="300" height="200" rx="10" ry="10" style="fill:yellow;stroke:black" />
<text x="40" y="130" style="fill:black;stroke:none">Abracadabra</text>
```




```
<rect x="20" y="30" width="300" height="200" rx="10" ry="10" />
<text x="40" y="130" >Abracadabra</text>
```

COORDINATES

- All graphics elements are rendered conceptually on to an **SVG infinite canvas**.
- A **viewport** can be established that defines a finite rectangular region within the canvas.
- Rendering uses the **painter's model**; elements later in the document are rendered on top of the preceding ones.
- The **viewport** is specified by attributes of the svg element.
- **Explicit width** and **height** attributes can be set
- An alternative is to use the **viewBox** attribute which specifies the lower and upper bounds of the viewport in both the X and Y directions

A complete SVG document containing the drawing defined above could be:

```
<svg viewBox="0 0 500 300">
<rect x="20" y="30" width="300" height="200" rx="10" ry="10" style="fill:yellow;stroke:black" />
<text x="40" y="130" style="fill:black;stroke:none">Abracadabra</text>
</svg>
```

```
<p> <object width="500" height="300" data="figure.svg" type="image/svg+xml">
<img src="figure.png" alt="Alternative PNG image" width="500" height="300"/>
</object>
</p>
```

- **The two approaches** (width/height and viewport) **are subtly different**.
- using width and height, no units have been specified so pixels are the assumed coordinate system.
- The **viewport** required is 320 pixels wide and 220 pixels high.

- The local user coordinate system for the duck is also set to be 320 by 220 with one pixel equal to one local user coordinate.
- In the second case, the local user coordinate is set to 500 wide and 300 high and **this is to be mapped to fit in the viewport**.
- A small viewport would have the mapping from user coordinate to pixels different from a large viewport.

COORDINATES

- In SVG, if no units are specified the assumption is that the **coordinates are defined in the local coordinate system**

- However, in defining the **viewport size** and in specifying the drawing, the **complete set of units defined in CSS** are available to the user

- (em, ex, px, pt, pc, cm, mm, in, %).

- If the drawing is to be displayed as part of a Web page, a **complex negotiation** takes place between the SVG plug-in and the browser
- taking into account any **constraints** on inserting the drawing in the Web page **or** by **the styling** applied to the page as a whole.
- As a result of this negotiation, **part of the image could be clipped, scaled or distorted**, depending on how the constraints are resolved.
- The user can control the effect somewhat through a **preserveAspectRatio** attribute and by specifying whether all the drawing must be visible or whether some parts can be obscured.

RENDERING MODEL

- The rendering model used by SVG is the **painter's model**
- the painter starts at the first element to be rendered and paints an area defined by the element.
- The artist then paints the second element and so on.
- If the second element is painted in the area occupied by the first element than it will obscure the first element unless the paint being applied is semi-transparent.
- Both the interior and the edge have to be painted. **In SVG, the interior is painted followed by the edge**

RENDERING ATTRIBUTES

- The **separation of style and content** has been an issue in text processing and computer graphics for many years
- The task of constructing a document using LaTex was reduced (as Lamport puts it) to a "logical design" task.
- The LaTeX system provided **typographic design**, through particular **style files**, and the document's author provided the **logical design**.
- **A whole class of documents (for example the papers in a journal) can thus be given a uniform appearance;**

Advantages of this approach include:

- Easy maintenance
- House style
- Clarity
- Adaptation to the end-user
- Design control

SVG Drawing Elements

- Path
- Text
- Basic Shapes

PATH

- The **path element** defines a shape that can be **open** or **closed**.
- A **path** consists of **a sequence of path segments** and in many cases this is a single path segment in which case path and path segment are synonymous.
- Each path segment consists of a **sequence of commands** where the first defines a new current position and the remainder define a line or curve from the current position to some new position which becomes the current position

The following path is a triangle:

<path d=”M 0 0 L 100 0 L50 100 Z”>

A path with two path segments would have the form:

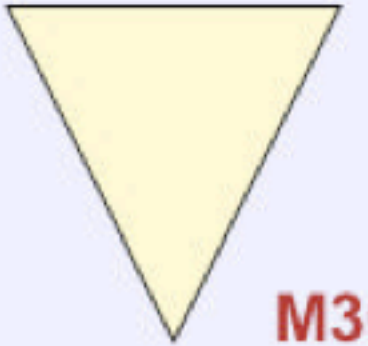
<path d=”M 0 0 L 100 0 L50 100 Z M300,300 L400,300 L350,400 Z”>

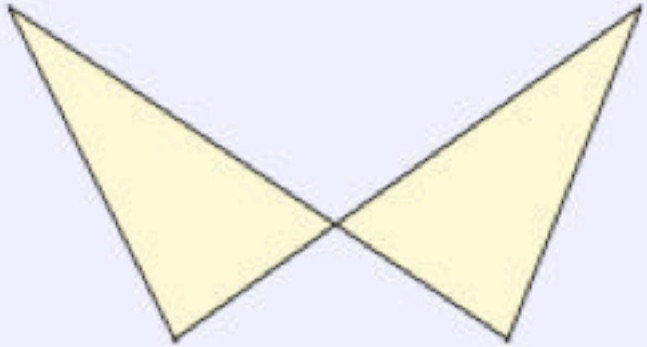
Command	Meaning	Parameters
M	Establish origin at point specified	Two parameters giving absolute (x,y) current position
L	Straight line path from current position to point specified	Two parameters giving absolute (x,y) position of the line end point which becomes the current position.
H	Horizontal line path from current position to point specified	Single parameter giving absolute X-coordinate of the line end point. The Y-coordinate is the same as that of the previous current position. The new point becomes the current position.
V	Vertical line path from current position to point specified	Single parameter giving absolute Y-coordinate of the line end point. The X-coordinate is the same as that of the previous current position. The new point becomes the current position.
Z	Straight line back to original Move origin	No parameters.

Points and Lines

M150,50 L200,100 H250 V50 h50 v-40 l50,50

Areas

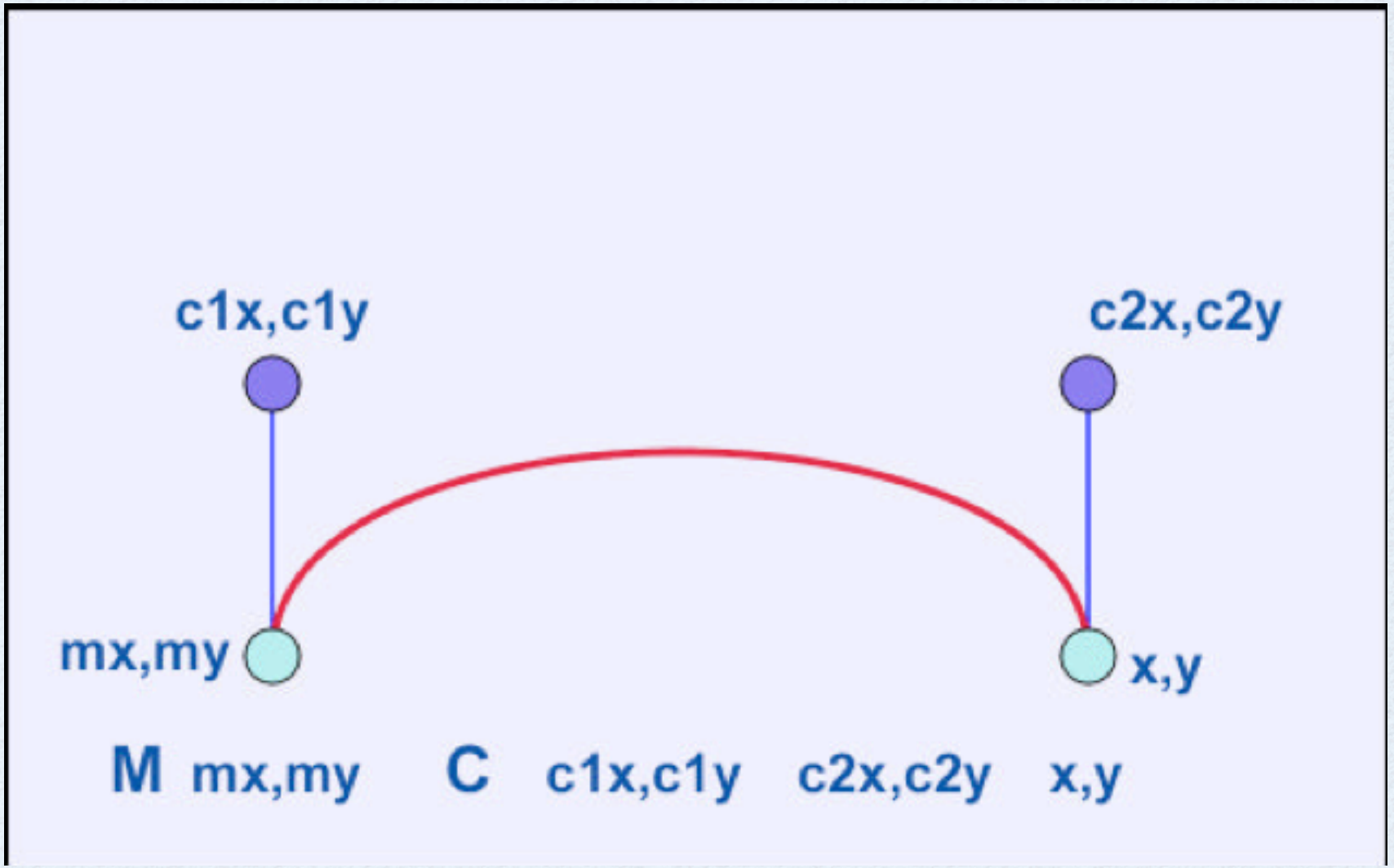




M300,140 L450,240 L490,140 L350,240 Z

M50,170 L150,170 L100,270 Z

PATH CUBIC BEZIER



CUBIC BEZIER EXAMPLES