COMP303
Internet Computing

# SVG Definitions

(Plus Pattern, Gradient and filter)

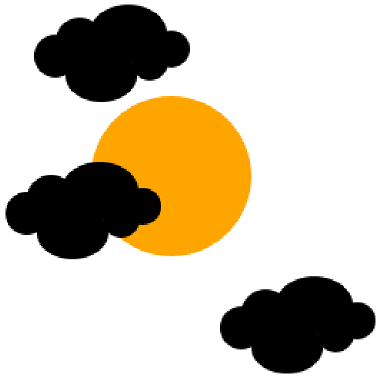
David Rossiter

## This Presentation

- There is a ‘definitions’ area of SVG in which something can be defined (once) and then used in the SVG (as many times as you want)
- These are some of the things you can define:
  - A definition of your own
  - Clipping paths
  - Gradients
  - Filters
  - Patterns

# Define an Object (01_bitmap_clip.svg)

- An object is defined once and used several times



```
<defs>  <g id="Cloud">
          <circle cx="24" cy="36" r="15"/>
          <circle cx="41" cy="26" r="17"/>
          <circle cx="90" cy="40" r="13"/>
          <circle cx="105" cy="31" r="13"/>
          <ellipse cx="75" cy="20" rx="27" ry="20"/>
          <ellipse cx="56" cy="50" rx="25" ry="18"/> </g> </defs>
<circle id="Sun" cx="125" cy="140" r="56" style="fill:orange"/>
<use id="SunCloud1" xlink:href="#Cloud" x="20" y="20" />
<use id="SunCloud2" xlink:href="#Cloud" x="0" y="130" />
<use id="SunCloud3" xlink:href="#Cloud" x="150" y="210" />
```

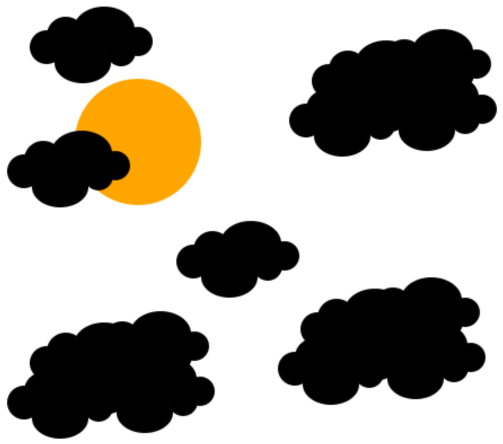
# Object Defined on Another Object

**(02_double_definition.svg)**

```
<defs> <g id="Cloud">
          <circle cx="24" cy="36" r="15"/>
          <circle cx="41" cy="26" r="17"/>
           <circle cx="90" cy="40" r="13"/>
          <circle cx="105" cy="31" r="13"/>
          <ellipse cx="75" cy="20" rx="27" ry="20"/>
          <ellipse cx="56" cy="50" rx="25" ry="18"/>  </g>
       <g id="SuperCloud">
          <use xlink:href="#Cloud" x="20" y="20" />
          <use xlink:href="#Cloud" x="70" y="10" />
          <use xlink:href="#Cloud" x="0" y="55" />
          <use xlink:href="#Cloud" x="75" y="50" />   </g>
</defs>
```

## Cont.. (02_double_definition.svg)

```
<circle id="Sun" cx="125" cy="140" r="56" style="fill:orange"/>

<use id="SmallCloud1" xlink:href="#Cloud" x="20" y="20" />
<use id="SmallCloud2" xlink:href="#Cloud" x="0" y="130" />
<use id="SmallCloud3" xlink:href="#Cloud" x="150" y="210" />

<use id="BigCloud1"
    xlink:href="#SuperCloud" x="250" y="30" />
<use id="BigCloud2"
    xlink:href="#SuperCloud" x="240" y="250" />
<use id="BigCloud3"
    xlink:href="#SuperCloud" x="0" y="280" />
```



# Bitmap and Clip Path (03_bitmap_clip.svg)

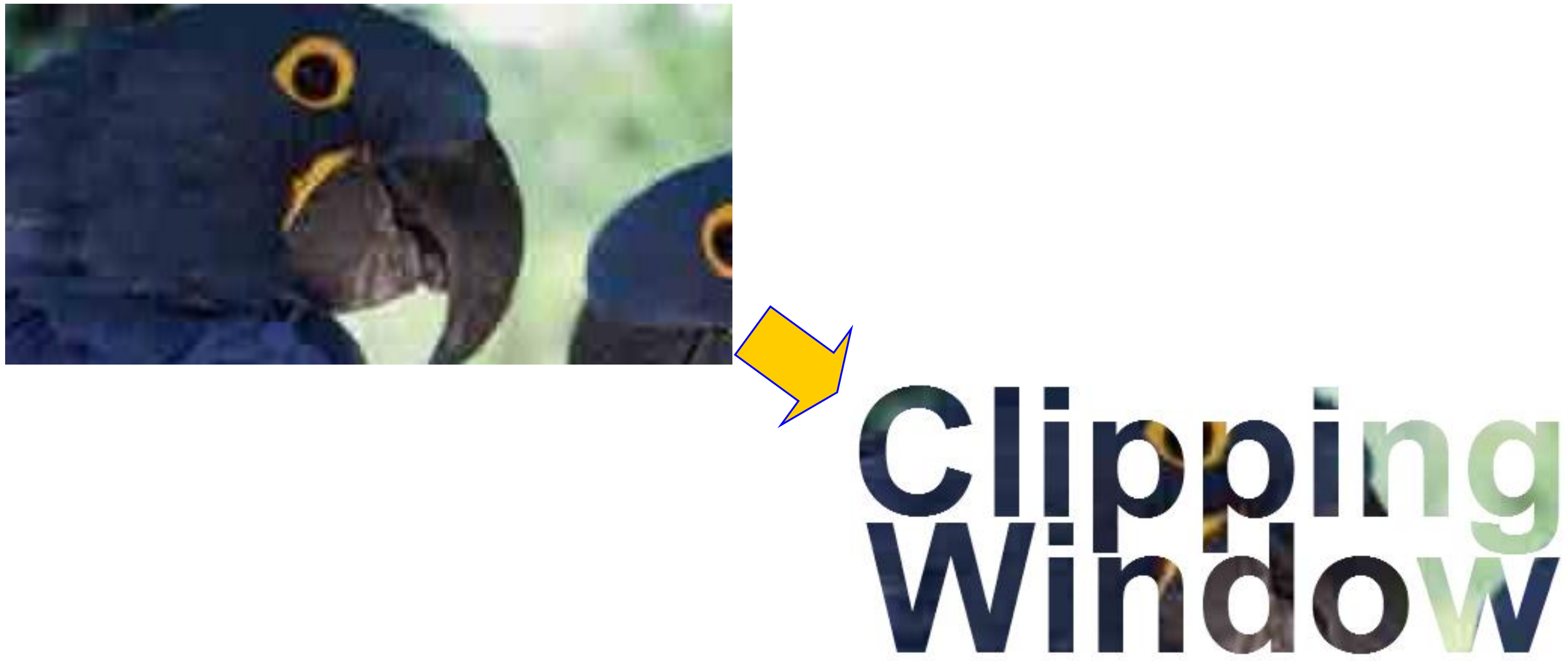
- A clip path is basically a ‘window’ through which the rest of the SVG can be shown

```
<style type="text/css">
    text { font-family: Arial;
           font-size: 120px;
           font-weight: bold;
             }
    rect { fill: black;
           fill-opacity: 1.0;
             }
</style>
```

```
<defs>
    <clipPath id="some_text" >
        <text x="0" y="130">
           Clipping</text>
        <text x="10" y="220">
           Window</text>
    </clipPath>   </defs>
```

```
<image xlink:href="image.jpg"
    style="clip-path:url(#some_text)"
    width="800" height="400"/>
```

## Bitmap and Clip Path (Cont.)



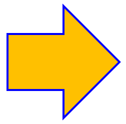
# Patterns (04_pattern.svg)

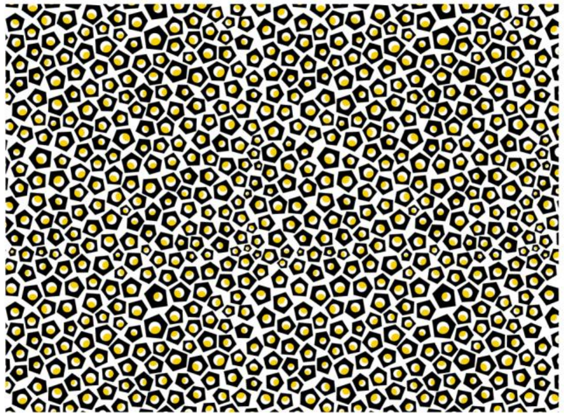
- A pattern fills an area by repeating an image many times

```
<defs> <pattern id="dotspattern" x="0" y="0"
            patternUnits="userSpaceOnUse"
            width="495px‶ height="495px">
          <image xlink:href="dots.png" x="0" y="0"
           width="495px" height="495px"/>   </pattern>    </defs>
<rect style="fill:url(#dotspattern)"
    width="950" height="700" x="50" y="50" />
```

Current user coordinate system)

*dots.png*



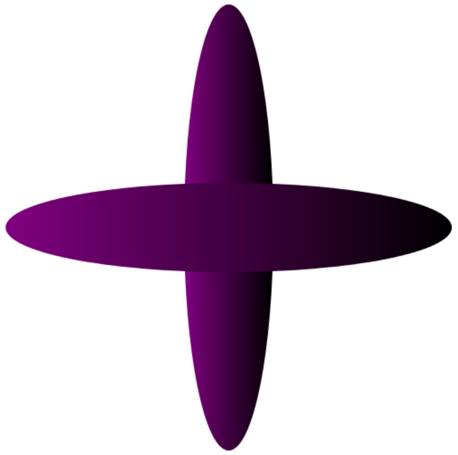


# Gradients (05_gradient.svg)

- Gradients can be visually very powerful

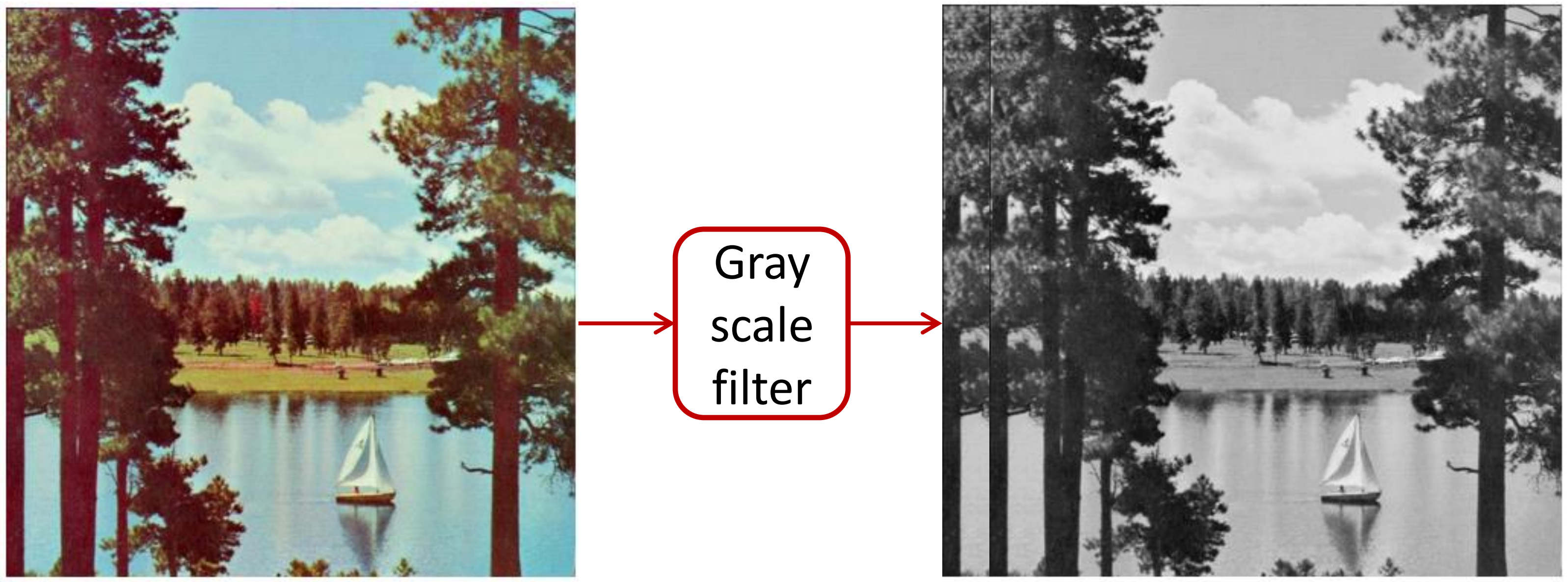
```
<defs>
   <linearGradient id="disc_gradient">
      <stop offset="0" style= "stop-color:purple"/>
      <stop offset="1" style= "stop-color:lilac"/>
   </linearGradient>
</defs>

<ellipse style="fill:url(#disc_gradient)"
      cx="400" cy="400" rx="50" ry="250" />
<ellipse style="fill:url(#disc_gradient)"
      cx="400" cy="400" rx="250" ry="50" />
```

Example creature made using gradients



# Image Filters

- Image filter applies effects on photos, and is supported by most image editors (e.g., Photoshop)
- Image filter is a mathematical function on an image, producing a new image (with effects) as output



*Original* *grayscale(100%)*

## Basic Ideas of Image Filters

- An image is represented by a 2D matrix (one for each color)
- A filter is a 2D matrix, which could have large or small sizes
- The filter matrix is applied to each pixel in the image matrix, the result is the sum of the products between the array cells
- Different operations between the 2D filter matrix and the image matrix result in different effect

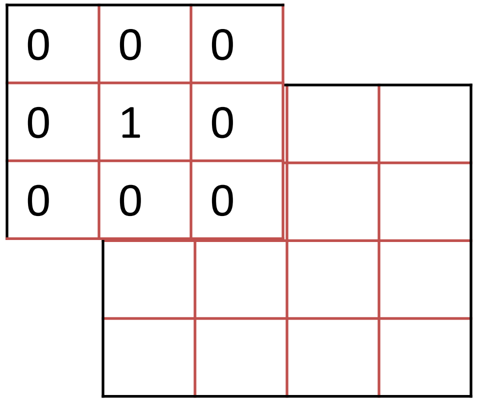
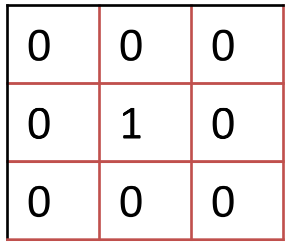


Image matrix



Filter matrix

1. cell’(0,0) = 1*cell(0,0) + 0 + 0 + 0

# Basic Ideas of Image Filters

- An image is represented by a 2D matrix (one for each color)
- A filter is a 2D matrix, which could have large or small sizes
- The filter matrix is applied to each pixel in the image matrix, the result is the sum of the products between the array cells
- Different operations between the 2D filter matrix and the image matrix result in different effect

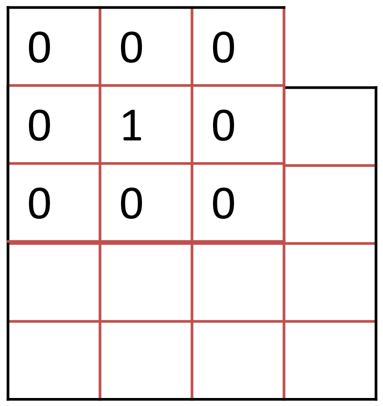


Image matrix

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

Filter matrix

1. cell’(0,0) = 0 + 0 … + 1*cell(0,0) + 0 … + 0
2. cell’(0,1) = 0 + 1*cell(0,1) + 0 + … + 0

What is the result matrix?

# Blurring

- An image can be blurred by averaging a pixel with its four neighbours

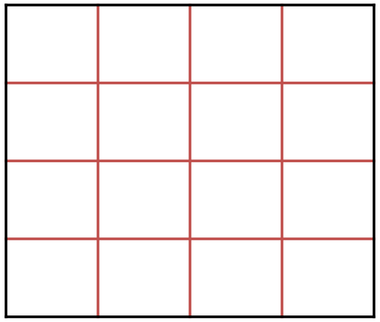


Image matrix

| | | |
|---|---|---|
| 0 | 0.2 | 0 |
| 0.2 | 0.2 | 0.2 |
| 0 | 0.2 | 0 |

Filter matrix

$$\text{cell}'(i,j) = 0.2*\text{cell}(i,j-1) + 0.2*\text{cell}(i-1,j) + 0.2*\text{cell}(i,j) + 0.2*\text{cell}(i+1,j) + 0.2*\text{cell}(i,j+1)$$

## SVG Filters

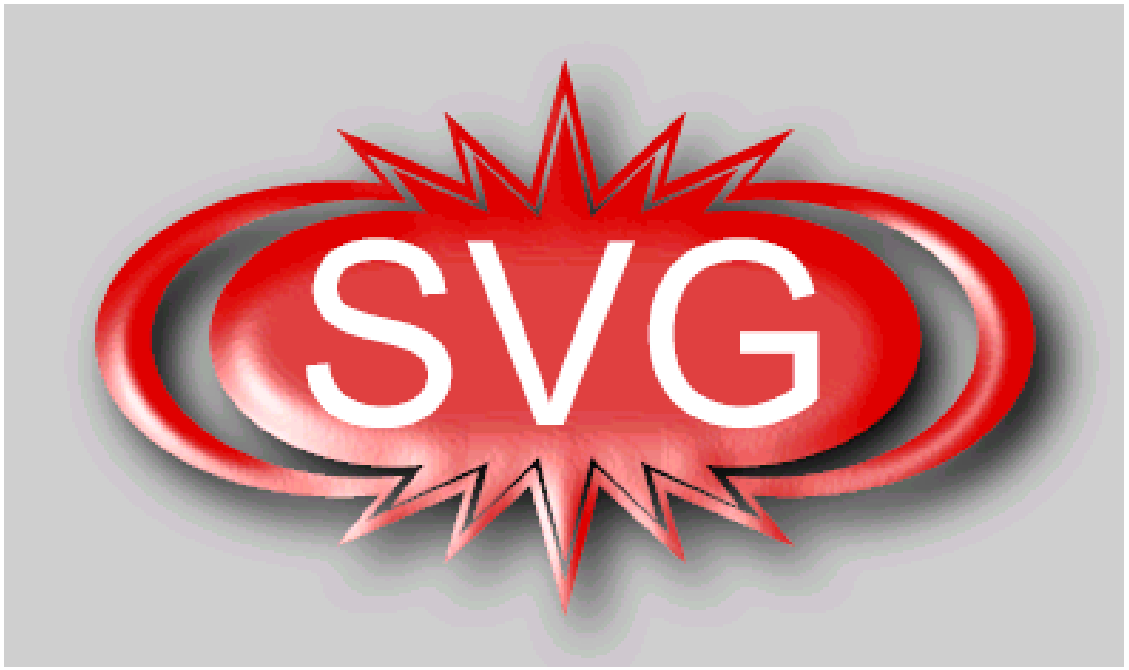
- SVG filters are supported by most browsers
  - W3C specifies 19 SVG filters (as of 2014)
- Since filtering is performed by the browser, it could be slow

## Defining Filters

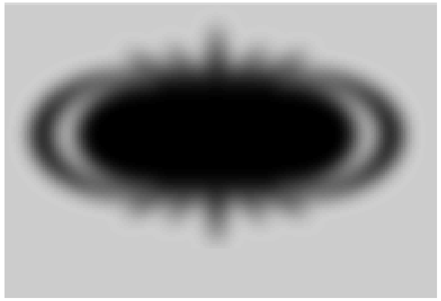
```
<svg>
   <defs>
      <filter id="cool_effect">
        <!-- Definition of filter goes here -->
      </filter>
   </defs>

   <text style="filter:url(#cool_effect)">
      In this example, the filter is
      applied to these words
   </text>
</svg>
```

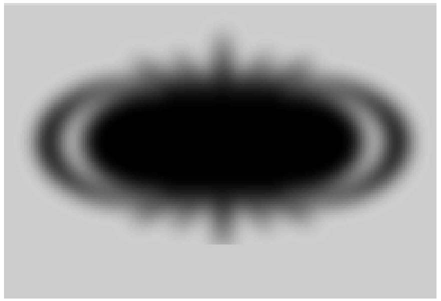
## Filters – Example (06_filter.svg)

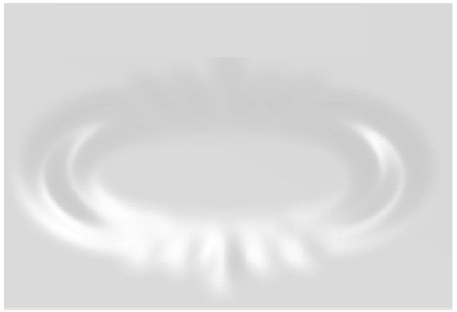
- Filters can be applied to an image in cascade, e.g., the following SVG uses a series of filters

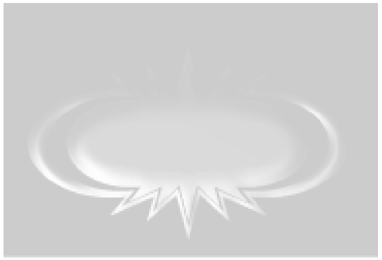


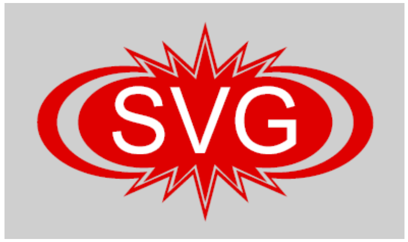
# Composition Stages


1. Gaussian Blur

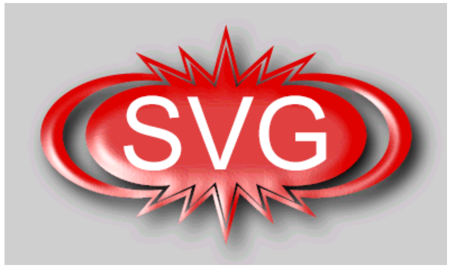

2. Offset (=move)


3. Specular Lighting (specular=point source)


4. Limited Specular Lighting



*Apply to some SVG*



Result

# Example - Complete Filter Code

```
<filter id= "MyFilter" >
<feGaussianBlur in="SourceAlpha" stdDeviation="4" result="blur"/>
<feOffset in="blur" dx="4" dy="4" result="offsetBlurredAlpha"/>

<feSpecularLighting in="blur" surfaceScale="5" specularConstant="0.9"
   specularExponent="20" lightColor="white" result="specularOut">
   <feDistantLight azimuth="135" elevation="30"/> </feSpecularLighting>

<feComposite in="specularOut" in2="SourceAlpha" operator="in"
   result="specularOut"/>

<feComposite in="SourceGraphic" in2="specularOut"
operator="arithmetic" k1="0" k2="1" k3="1" k4="0" result="litPaint"/>

<feMerge>
   <feMergeNode in="offsetBlurredAlpha"/>
   <feMergeNode in="litPaint"/>   </feMerge>   </filter>
```

Trace the relationship between "in" and "result" files

# Take Home Message

- Graphical editors like Photoshop and Inkscape are used to create large SVGs are filters (not to mention Instagram which was successful by providing filters)
- All powerful things are done by a tagging language (SVG)
- … …