COMP4021
Internet Computing

# Introduction to SVG

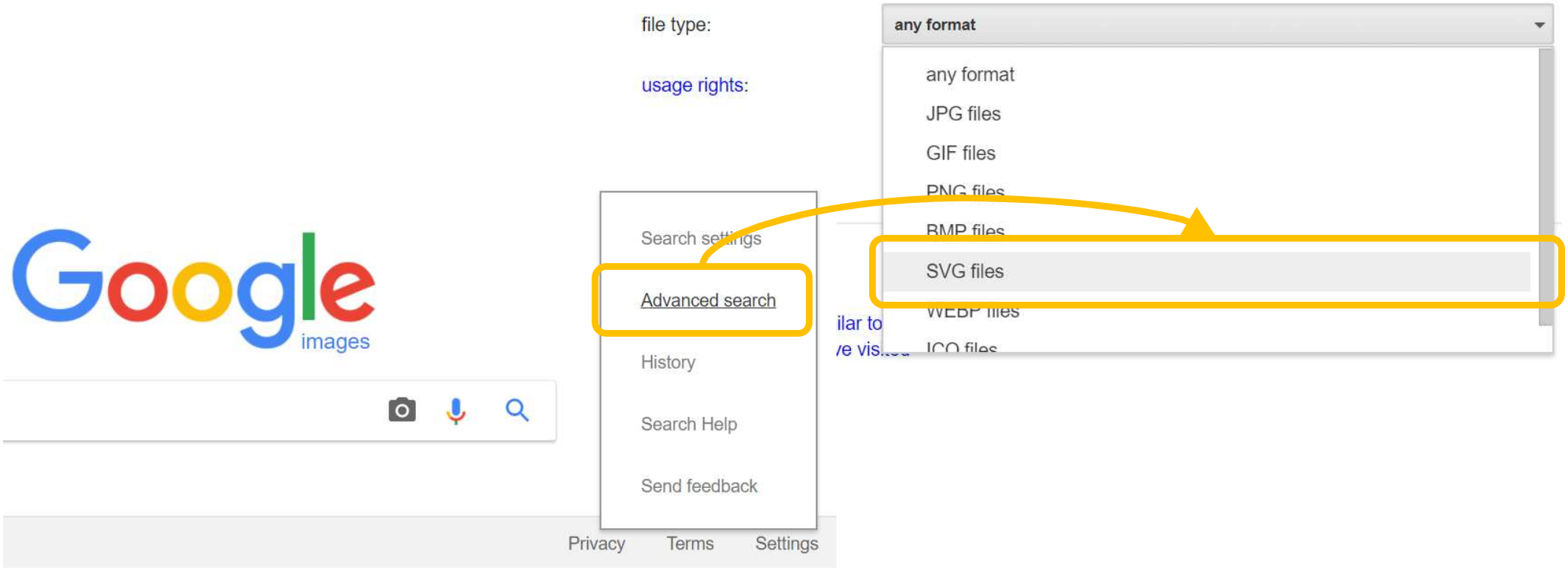
Gibson Lam and David Rossiter

# SVG

- SVG is a vector graphics language for web pages
- You can use it to make logos, figures and charts easily
- In this presentation, we will look at how to create SVG content and the many different elements in SVG

# SVG Images on the Web

- Before we look at how you create SVG, let's see what are available on the web
- You can look for SVG images in Google by changing the search settings



Google

mcdonalds

All Maps Images News Videos More Settings Tools View saved SafeSearch

Size Color Type Time SVG files Usage rights More tools Clear

joker | starbucks | walmart | burger king | wendy's | kfc | subway | jollibee | japan | egypt | america









# Standalone or Embedded SVG

- SVG can be:
  - Used as a standalone file
    - Filename ends with `.svg`
  - Embedded inside an HTML file
    - Filename ends with `.htm` or `.html`

- Most examples in this discussion are standalone SVG files

# A Standalone SVG – HKUST.svg

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg PUBLIC
"-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/
svg11.dtd">
<svg xmlns="http://www.w3.org/2000/svg"
     version="1.1"
     width="390" height="600"
     viewBox="0 0 390 600">

...SVG content...

</svg>
```

# Embedding SVG in a Webpage

```
<!DOCTYPE html>
<html xmlns='http://www.w3.org/1999/xhtml'>
<head>
    <title>HKUST Logo</title>
</head>
<body>
    <h1>HKUST Logo</h1>
    <svg xmlns="http://www.w3.org/2000/svg"
         version="1.1"
         width="185" height="300"
         viewBox="0 0 390 600">

    ...SVG content...

    </svg>
</html>
```

HKUST Logo

# Starting to Use SVG

- To start using SVG, you make a simple SVG ‘drawing’ area, like this:

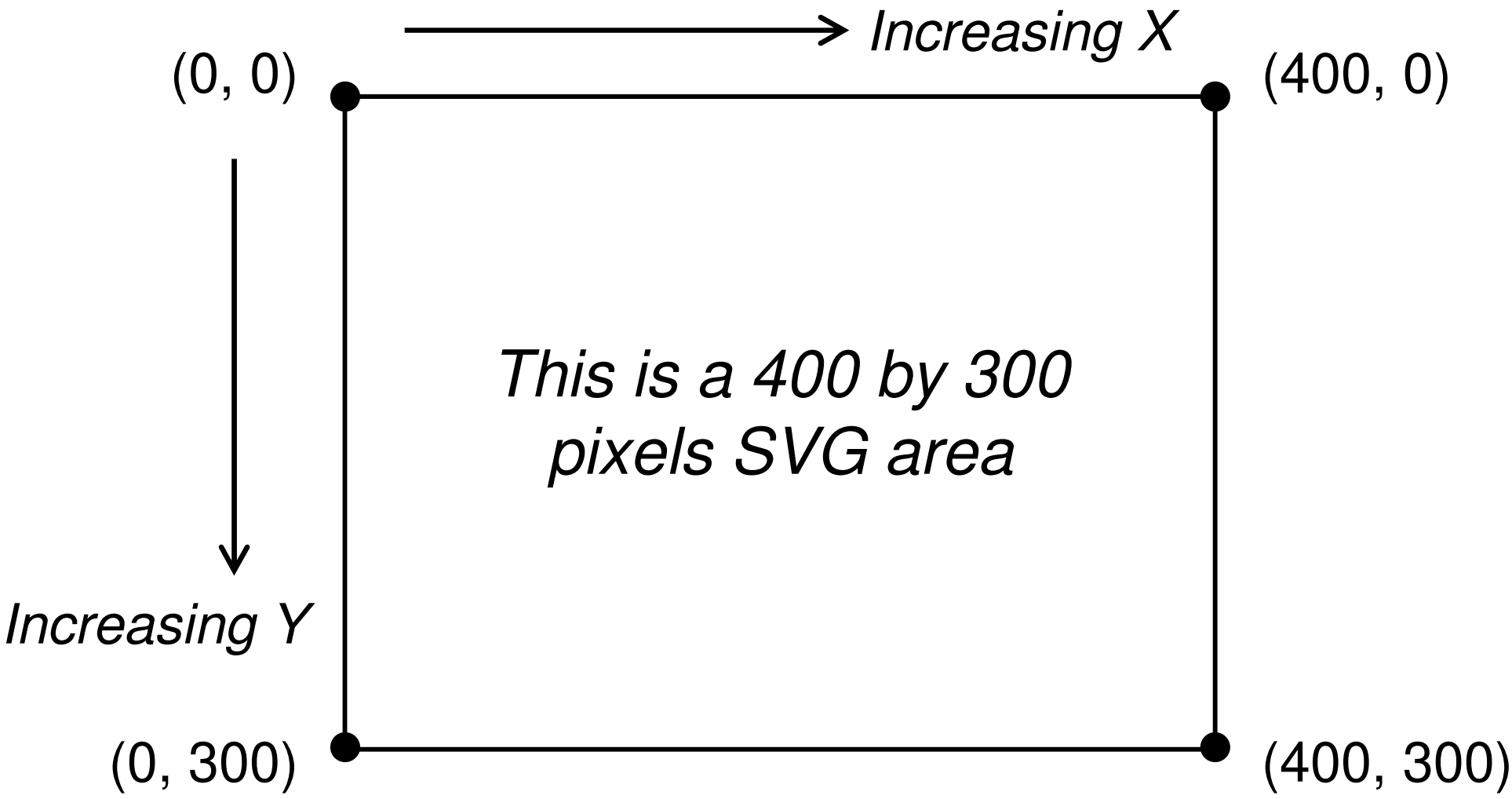
```
<svg xmlns="http://www.w3.org/2000/svg"
     width="400" height="300">

...SVG content...

</svg>
```

*Creating an SVG area with a size of 400 by 300 pixels*

- You then add text and shapes into it

# The SVG Coordinate System

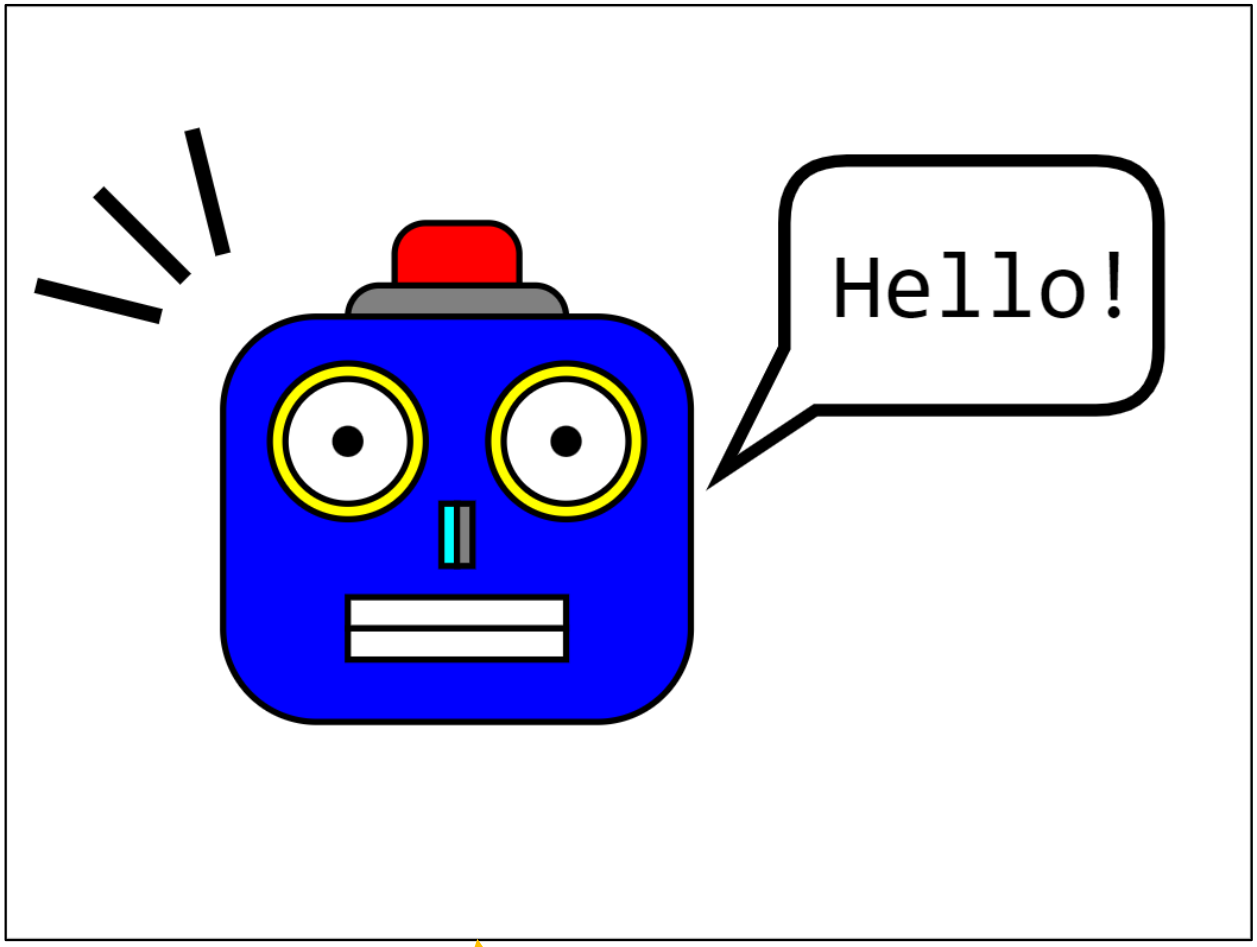
- Here is the coordinate system in a 400 by 300 pixels SVG area:



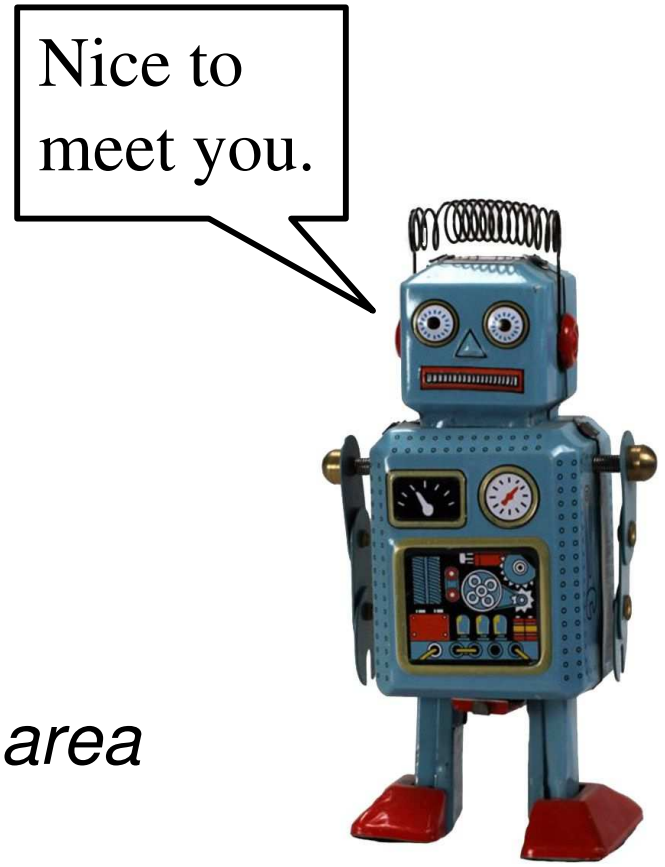
# SVG Drawings

- You can add many kinds of shapes into an SVG area
- We will talk about the more commonly used ones:
  - Lines
  - Rectangles
  - Circles
  - Paths
  - Text

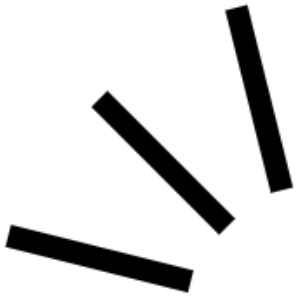
# Let's Draw a Robot Head!




*400 by 300 pixels SVG area*

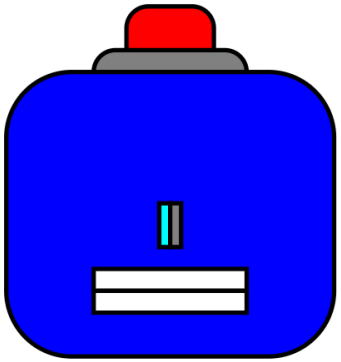


# The Robot Parts

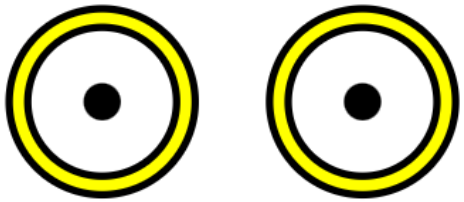


- You can use lines to draw these:

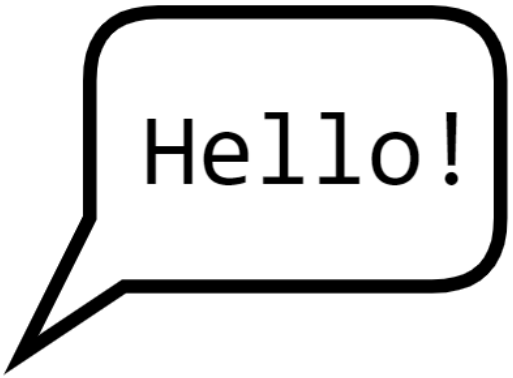
rectangles to draw these:



circles to draw these:



a path and a piece of text to draw this:

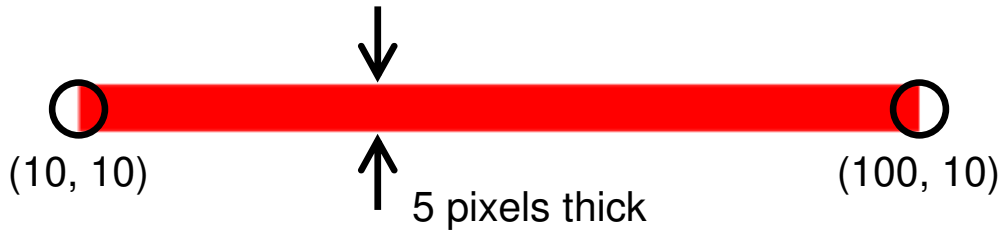


# Using Lines

- You can use the SVG `<line>` element to draw a line between two points (x1, y1) and (x2, y2)
- For example, you can draw a red line from (10, 10) to (100, 10) using this code:

```
<line x1="10" y1="10" x2="100" y2="10"
      stroke="red" stroke-width="5" />
```

- See details in the next slide



# Drawing a Line

- This is the first point (10, 10)

- This is the second point (100, 10)

```
<line x1="10" y1="10" x2="100" y2="10"
      stroke="red" stroke-width="5" />
```

- This is the colour and width of the line
- This closes the `<line>` tag, which is required for SVG

# Closing Tags

- In HTML, you do not need a closing tag for elements that do not enclose any content such as `<img>` and `<br>`
- SVG works differently so that all elements must have a closing tag
- If the element does not enclose anything, the tag can be closed by a '/' at the end

`<line ...`*attributes*`... />`

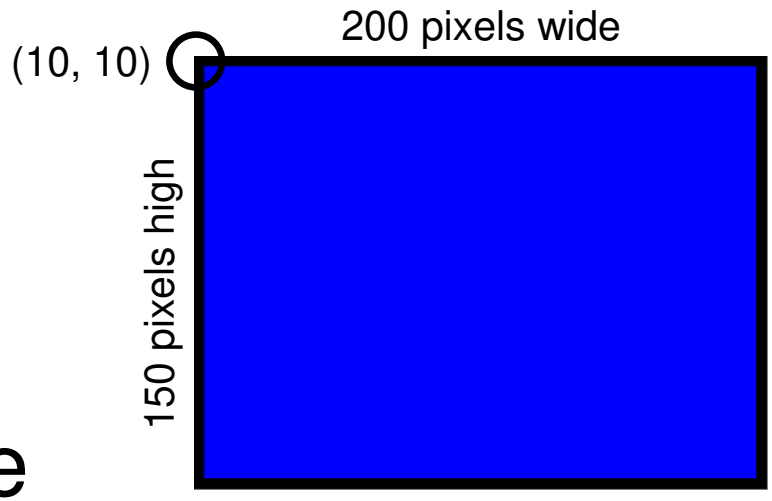
*Close the tag*

# The Robot Lines

- You can draw as many lines as you want
- You can then use this code to draw the three lines near the corner of the robot:

```
<line x1="60" y1="40" x2="70" y2="80"
      stroke="black" stroke-width="5" />
<line x1="30" y1="60" x2="58" y2="88"
      stroke="black" stroke-width="5" />
<line x1="10" y1="90" x2="50" y2="100"
      stroke="black" stroke-width="5" />
```

# Using Rectangles




- Creating a rectangle in SVG is very similar to creating a line
- You can use the `<rect>` element to draw a blue rectangle at (10, 10) with a size of (200, 150), e.g.:

```
<rect x="10" y="10"
      width="200" height="150"
      stroke="black" stroke-width="4"
      fill="blue" />
```

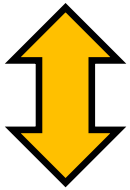
*A black outline for the rectangle*

*Fill the rectangle with blue*

# Using CSS

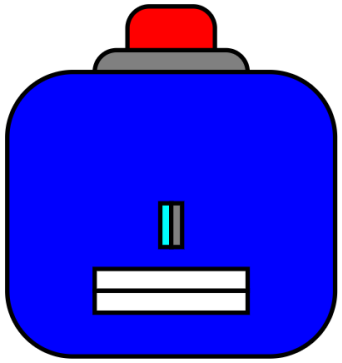
- You can use the `style` attribute to specify the visual styles, e.g.:

```
<rect ... stroke="black" stroke-width="4"
          fill="blue" />
```



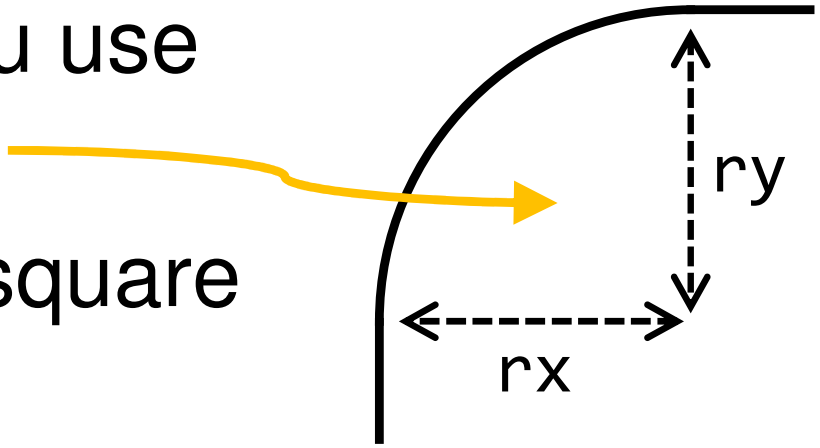
```
<rect ... style="stroke: black;
           stroke-width: 4; fill:blue" />
```

- You can also use the `class` attribute, style sheets and CSS selectors if you want to

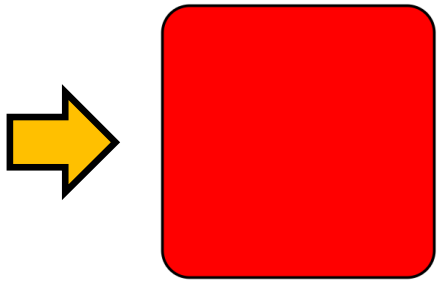


# Making Rounded Corners

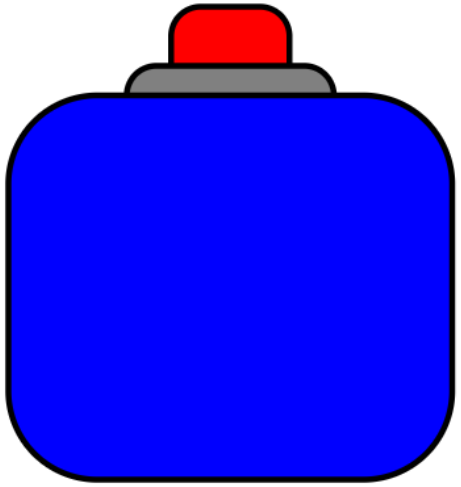
- If you look at the target robot head, it has rounded corners
- To do that in `<rect>`, you use the `rx` and `ry` attributes
- Here is an example red square with rounded corners:




```
<rect x="10" y="10"
      width="100" height="100"
      stroke="black" fill="red"
      rx="10" ry="10" />
```

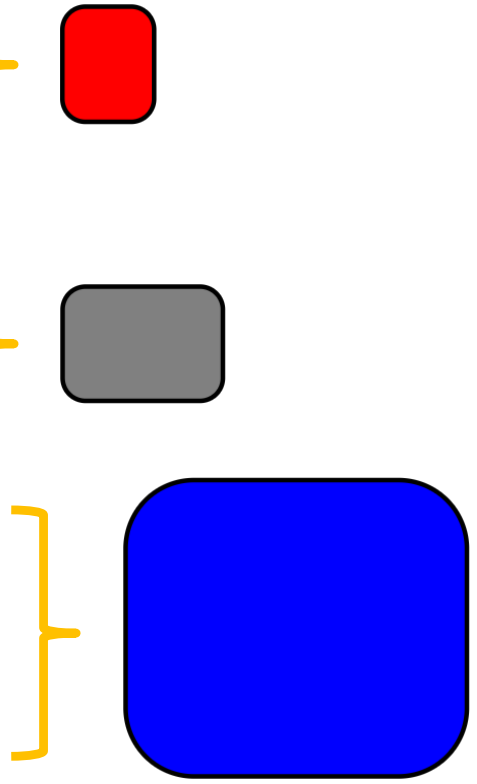


# The Robot's Face



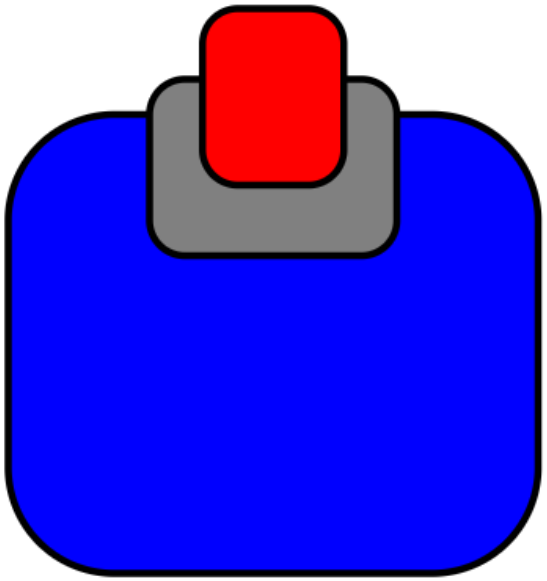
- You can use the following three rectangles to make the face

```
<rect x="125" y="70" width="40" height="50"
      stroke="black" stroke-width="2"
      fill="red"
      rx="10" ry="10" />
<rect x="110" y="90" width="70" height="50"
      stroke="black" stroke-width="2"
      fill="gray"
      rx="10" ry="10" />
<rect x="70" y="100" width="150" height="130"
      stroke="black" stroke-width="2"
      fill="blue"
      rx="30" ry="30" />
```



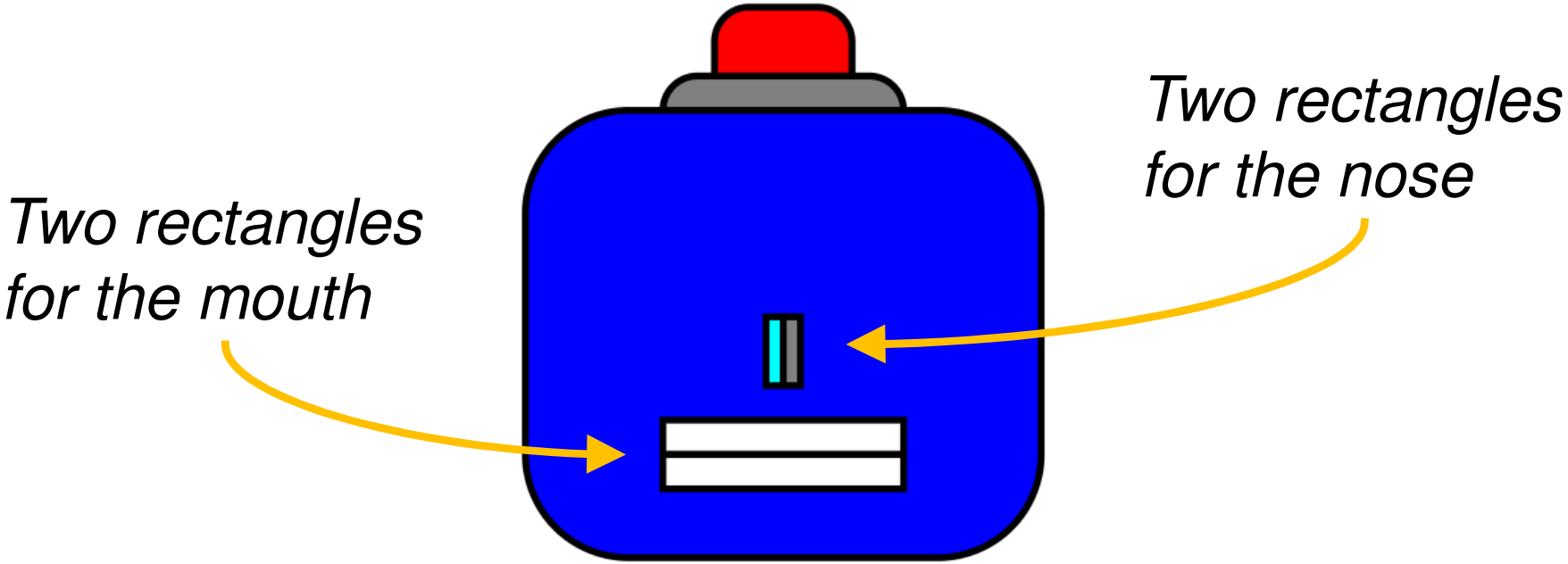
# Drawing Order

- In the previous slide, the three rectangles overlap each other
- The drawing order of the rectangles follows their order inside the SVG file, i.e. the first element in the file is drawn first, then the second element, and so on
- If the three rectangles are put in the opposite order, they will look like this:

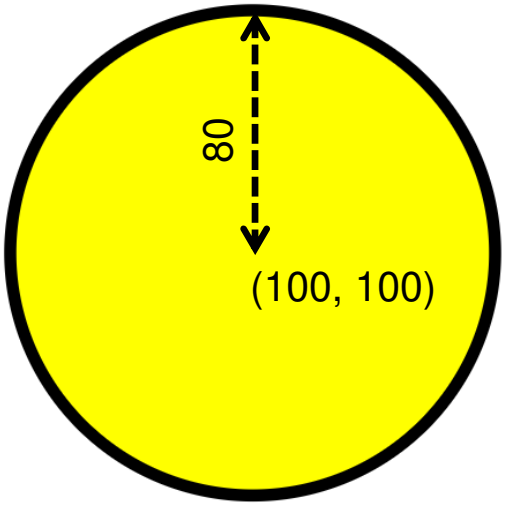


# The Robot's Nose and Mouth

- After creating the robot's face, you can use four more rectangles to create a nose and a mouth in the face



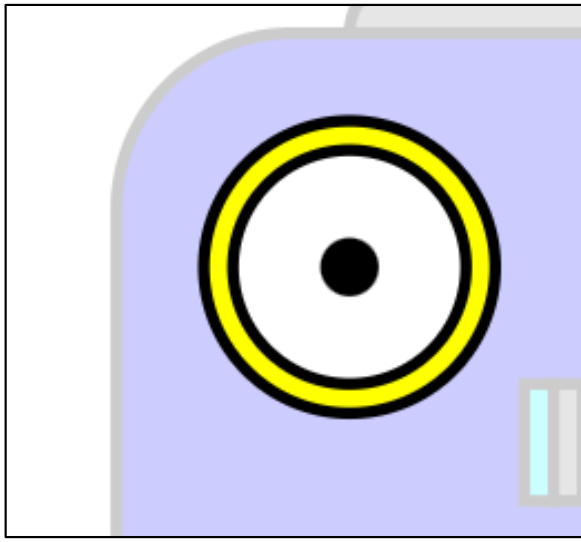
# Using Circles




- You can draw a circle using the `<circle>` element
- Here is an example that draws a yellow circle having a radius of 80 and centred at (100, 100)

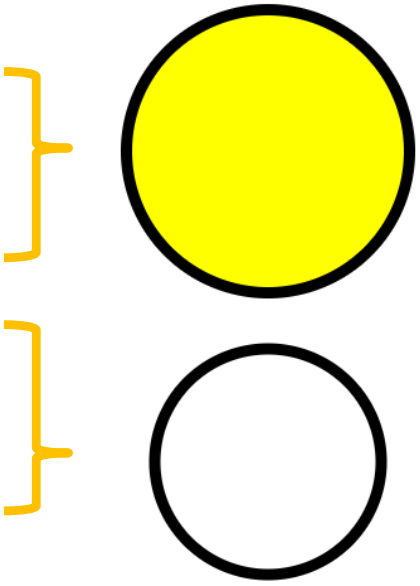
```
<circle cx="100" cy="100" r="80"
        stroke="black" stroke-width="4"
        fill="yellow" />
```

# The Robot's Eyes

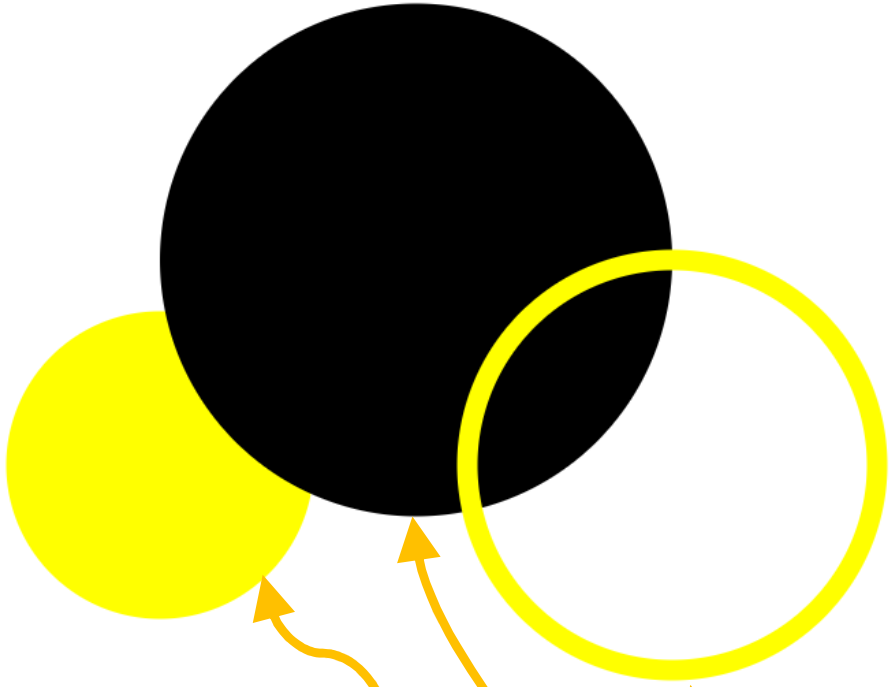


- You can easily create one eye of the robot using three circles:

```
<circle cx="110" cy="140" r="25"
        stroke="black" stroke-width="2"
        fill="yellow" />
<circle cx="110" cy="140" r="20"
        stroke="black" stroke-width="2"
        fill="white" />
<circle cx="110" cy="140" r="5"
        fill="black" />
```



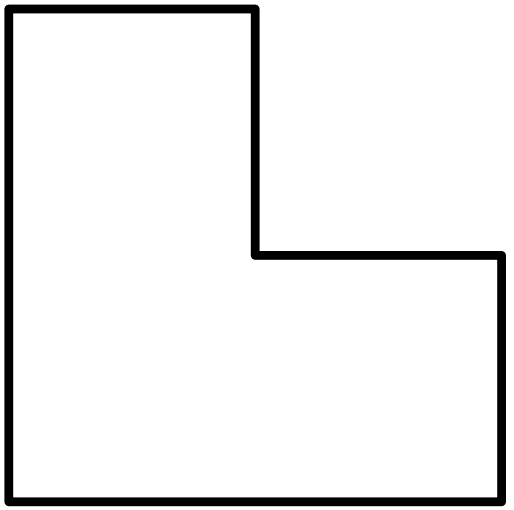
# Outline and Fill Colours



- If you do not specify the `stroke` and `stroke-width` attributes, the shapes will not have any outline
- If you do not specify the `fill` attribute, the shape will be filled with black
- If you want a hollow shape, you need to specify the `stroke` attribute and set the `fill` attribute to `none`

# Using Paths

- Sometimes you have shapes that cannot be made using basic shapes such as rectangles and circles
- For example, you would not be able to make an L-shape shown on the right using only two rectangles



- Using the `<path>` element allows you to build your own shape

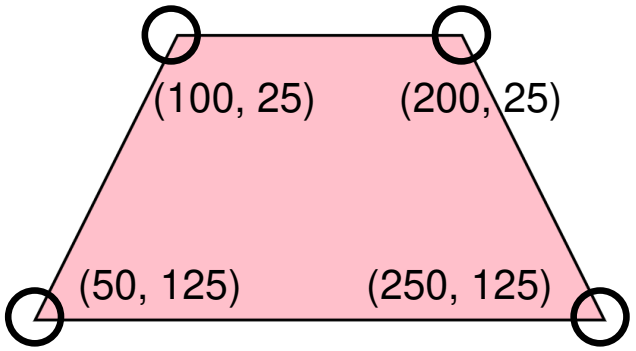
# Creating Paths

- A path is a kind of drawing language in itself
- You can describe any shape/path using these:

| | |
|---|---|
| `M` | Move to |
| `L` | Draw a straight line to |
| `H` | Draw a horizontal line to |
| `V` | Draw a vertical line to |
| `C` | Draw a cubic curve to |
| `S` | Draw a smooth cubic curve to |
| `Q` | Draw a quadratic curve to |
| `T` | Draw a smooth quadratic curve to |
| `A` | Draw an arc to |
| `Z` | Finish/ go back to the beginning |

*You don't need to understand these*

# Path Examples

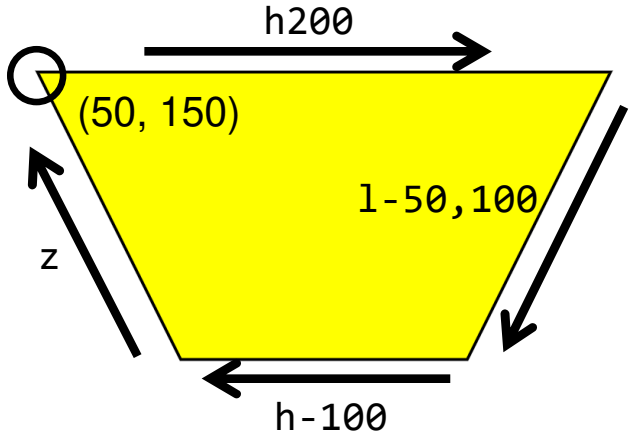



- For example, here is a path:

```
<path d="M100,25 L200,25 L250,125 L50,125 Z"
      fill="pink" stroke="black" />
```

- You can change the command letters to small letters; in that case, the commands will use relative movement, like this:

```
<path d="M50,150 h200 l-50,100 h-100 z"
      fill="yellow"
      stroke="black" />
```



# Using Text

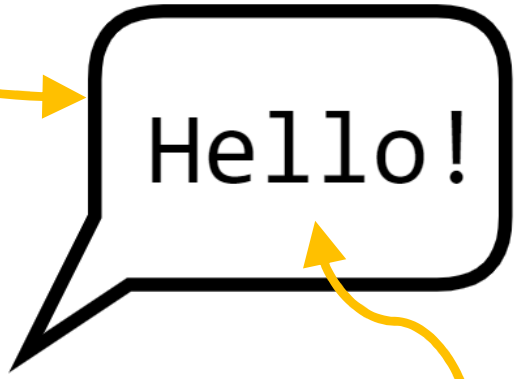
- Text can be added by the `<text>` elements
- For example, you can put a piece of text "I am inside SVG!" at (50, 100)

```
<text x="50" y="100"
      font-size="30" fill="navy">
  I am inside SVG!
</text>
```

# Robot's Speech Bubble

- You can then draw the speech bubble of the robot using a path and text

```
<path d="M230,150
         l20,-40
         l0,-40
         q0,-20 20,-20
         l80,0
         q20,0 20,20
         l0,40
         q0,20 -20,20
         l-90,0
         z"
      stroke="black" stroke-width="4"
      fill="white" />

<text x="265" y="100"
      font-family="monospace" font-size="30">
    Hello!
</text>
```




*This means using a 'fixed width' font*

# Creating SVG in Graphical Editors

- If you find it difficult to ‘draw’ pictures by typing in SVG inside a text editor, you can use some graphical editors
- For example, here are two editors that can create SVG:
  - Adobe Illustrator
    - It can output graphics as SVG and it is available in the Virtual Barn
  - SVG Edit (https://github.com/SVG-Edit/svgedit)
    - This is a free online editor for SVG