Publishing and Printing with SVG

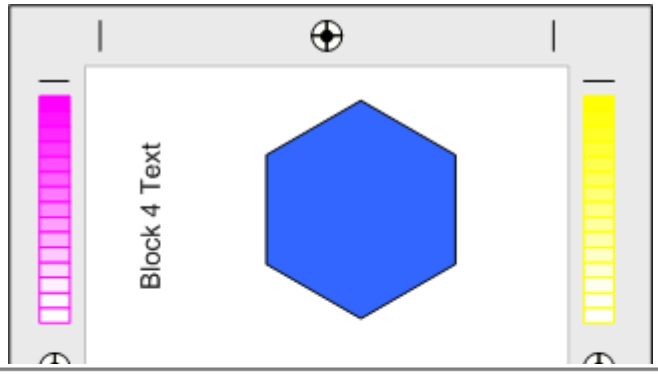
By Anthony Grasso

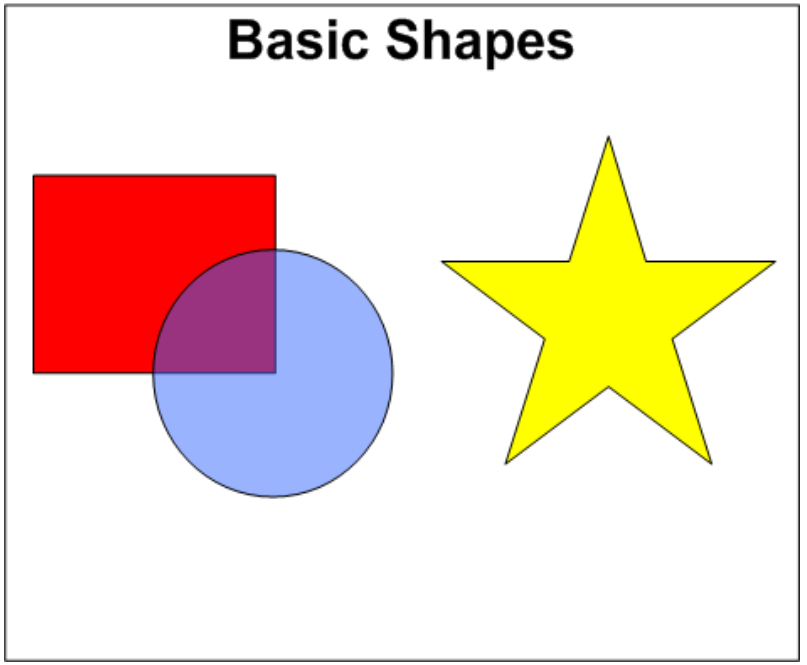
Software Engineer, CiSRA

6 September 2007

Problems Faced in Printing

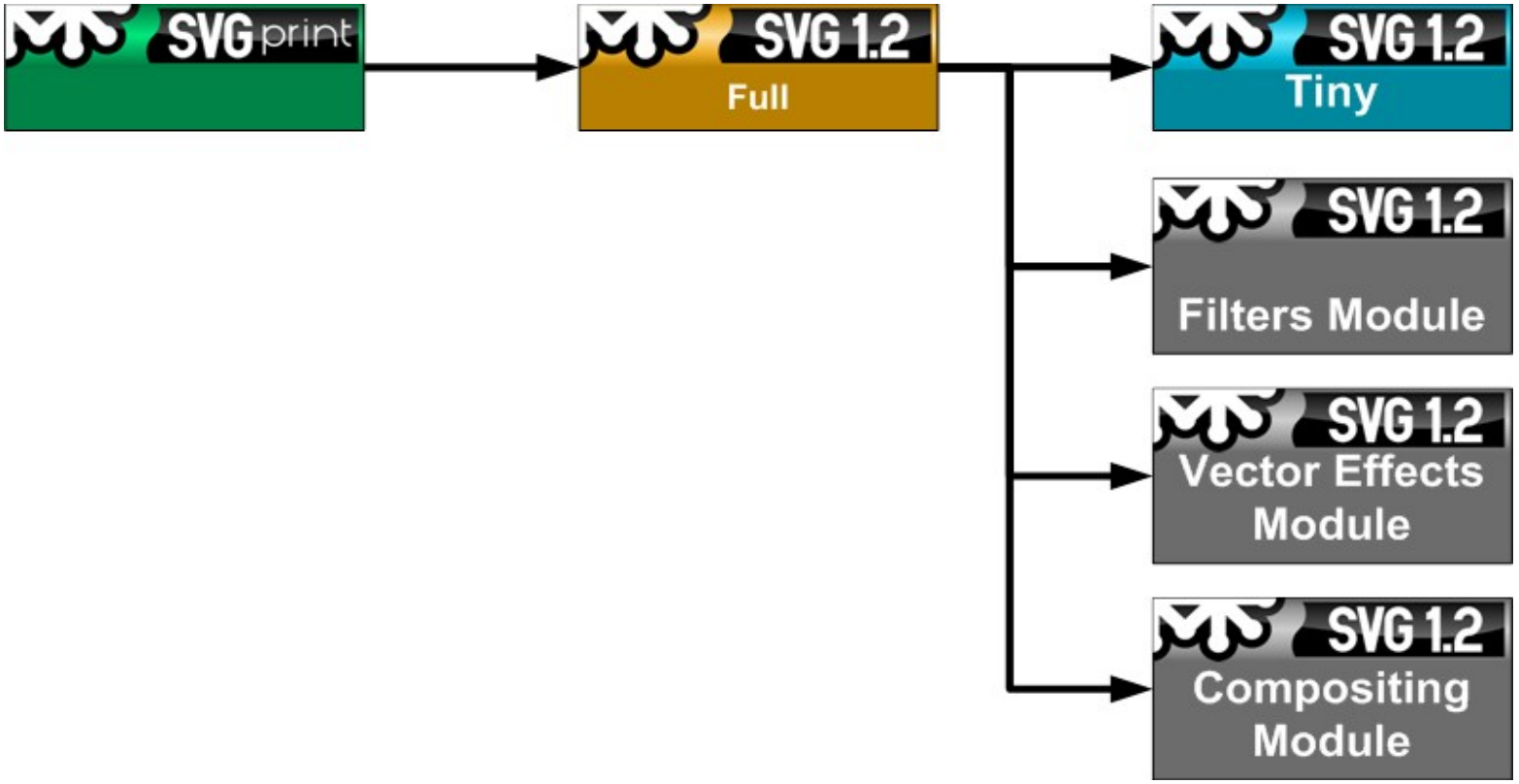
- Scenario 1:
  - Large printing company
  - Printing with specific kind of inks
  - Require advanced color management not present in SVG
- Scenario 2:
  - User wants to print SVG Tiny 1.2 content
  - Content on a web browser





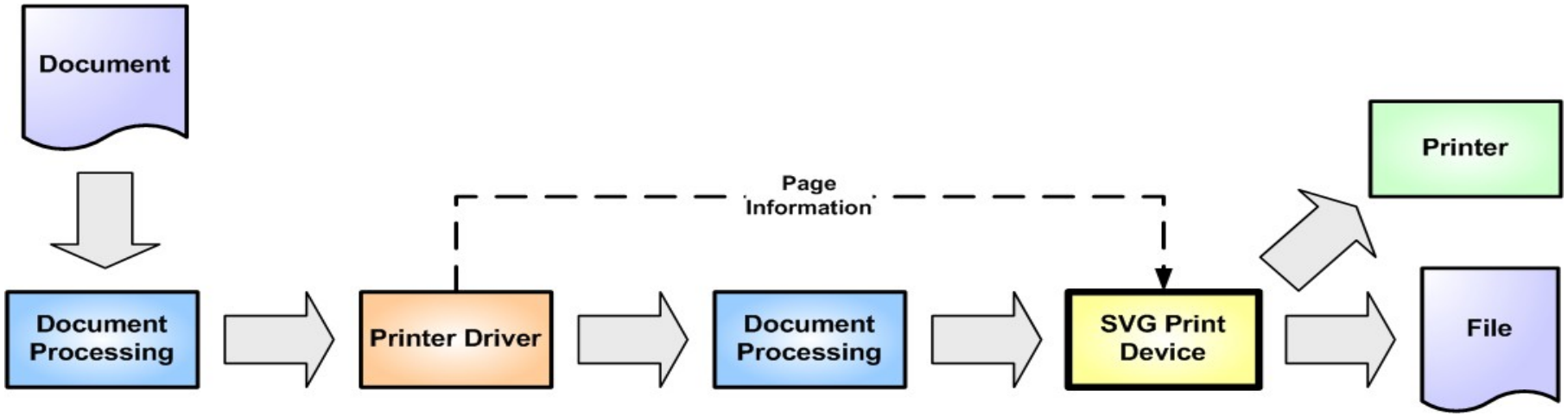
SVG Print

- Features: Basic pagination and Advanced color
- Used in conjunction with other specs



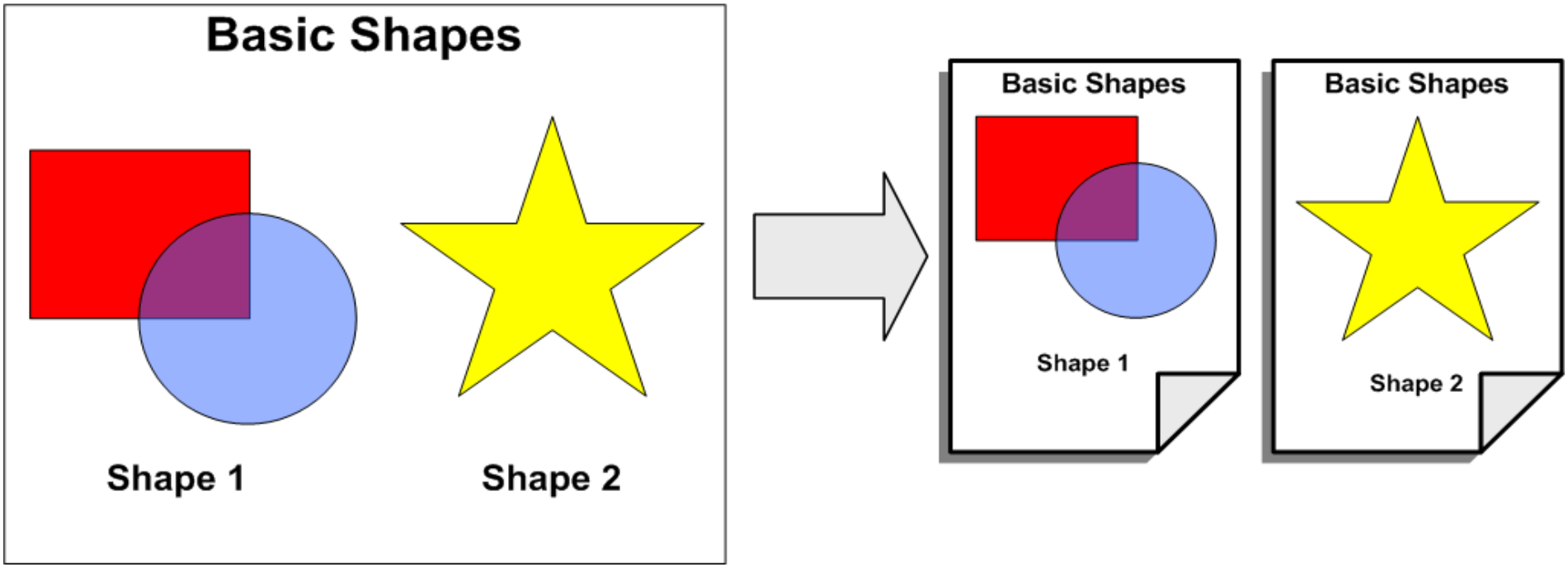
Print Workflow

- Typical print workflow:



Breaking SVG into Pages

- Use **`pageSet`** element for pagination
- The **`page`** element defines a page
  - Nested within a **`pageSet`** element

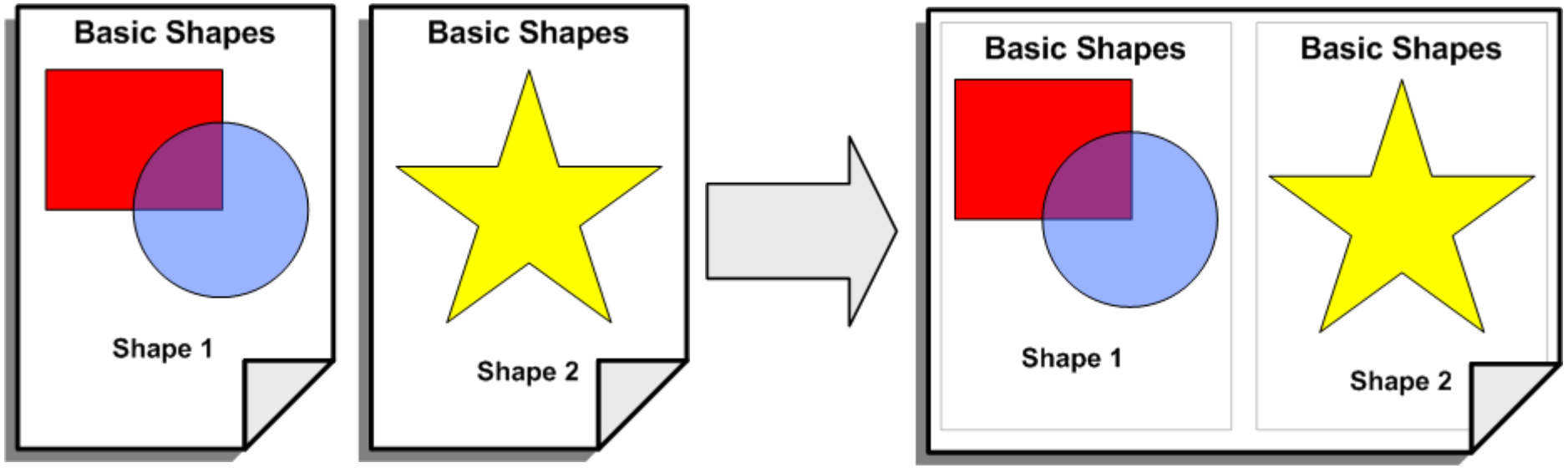


Breaking SVG into Pages

```
<svg width="100%" height="100%" viewBox="0 0 800 600"
  xlmns="http://www.w3.org/2000/SVG">

  <!-- No page adjustment settings applied to document -->
  <pageSet>
    <!-- First sheet of paper -->
    <page>
      <text x="10" y="10" font-size="14">Basic Shapes<text>
      <rect x="20" y="30" width="160" height="100" fill="red"
        stroke="black"/>
      <circle cx="180" cy="130" r="50" fill="blue" fill-opcity="0.5"
        stroke="black"/>
      <text x="50" y="270" font-size="10">Shape 1<text>
    </page>

    <!-- Second sheet of paper -->
    <page>
      <text x="10" y="10" font-size="14">Basic Shapes<text>
      <polygon fill="yellow" stroke="black" points="105,30 120,100
       200,100 135,150 150,200 105,135 50,200 75,150 10,100 90,100"/>
      <text x="50" y="280" font-size="10">Character 2<text>
    </page>
  </pageSet>

</svg>
```

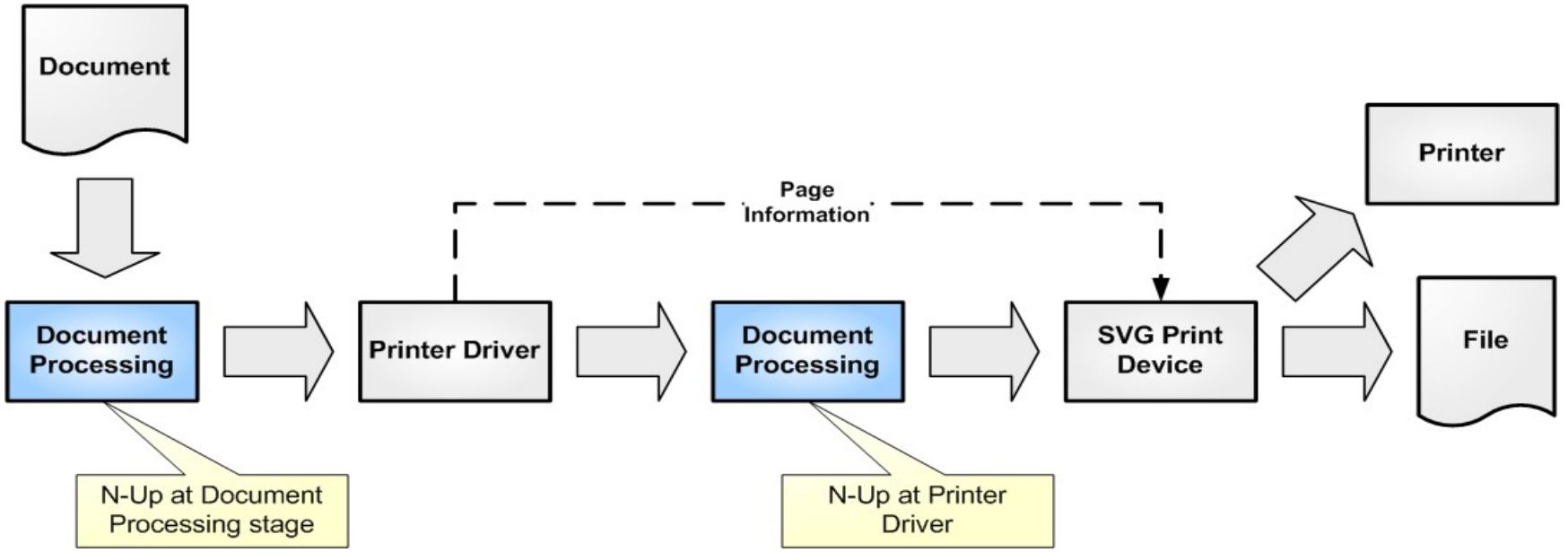
N-Up printing

- What if we want to do this?



N-Up printing

- How does N-Up printing work with SVG Print?
- Can be performed in two places:
  - Before sending document to Printer Driver (Processing stage)
  - After sending document to Printer Driver (In Printer Driver)



N-Up printing: Within Printer Driver

- N-Up settings from printer driver override SVG
- N number of page elements on 1 sheet of paper

```
<svg width="100%" height="100%" viewBox="0 0 1000 1000"
  xlmns="http://www.w3.org/2000/SVG">

  <pageSet>
    <page>
      <!-- Content for left side of first sheet of paper -->
    </page>

    <page>
      <!-- Content for right side of first sheet of paper -->
    </page>

    <page>
      <!-- Content for left side of second sheet of paper -->
    </page>

    <page>
      <!-- Content for right side of second sheet of paper -->
    </page>
  </pageSet>
</svg>
```

1st Sheet of paper

2nd Sheet of paper

N-Up printing: Before Printer Driver

- SVG is modified to apply N-Up settings
- 1 `page` element = 1 sheet of paper when printing

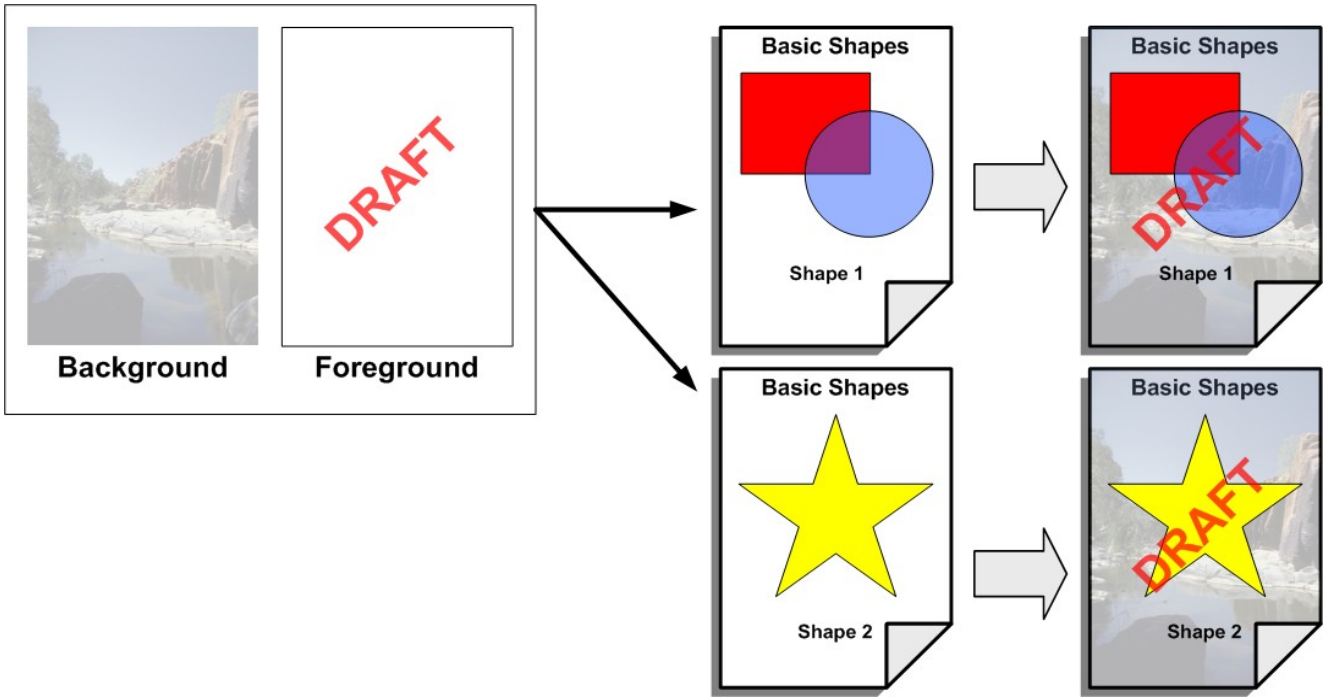
```
<svg width="100%" height="100%" viewBox="0 0 1000 1000"
  xlmns="http://www.w3.org/2000/SVG">

  <pageSet>
    <page page-orientation="90">
      <g transform="translate(0 0) scale(0.707)" >
        <!-- Content for left side of first sheet of paper -->
      </g>

      <g transform="translate(700 0) scale(0.707)" >
        <!-- Content for right side of first sheet of paper -->
      </g>
    </page>

    <page page-orientation="90">
      <!-- Content for second sheet of paper -->
    </page>
  </pageSet>
</svg>
```

1st Sheet of paper

2nd Sheet of paper

Master Pages

- What if we want content to appear in the background and/or in the foreground of our pages?
  - Could do it the repeat "manual" way
  - Could do it using Master Pages



Master Pages

- Set resuable content using **`masterPage`** element

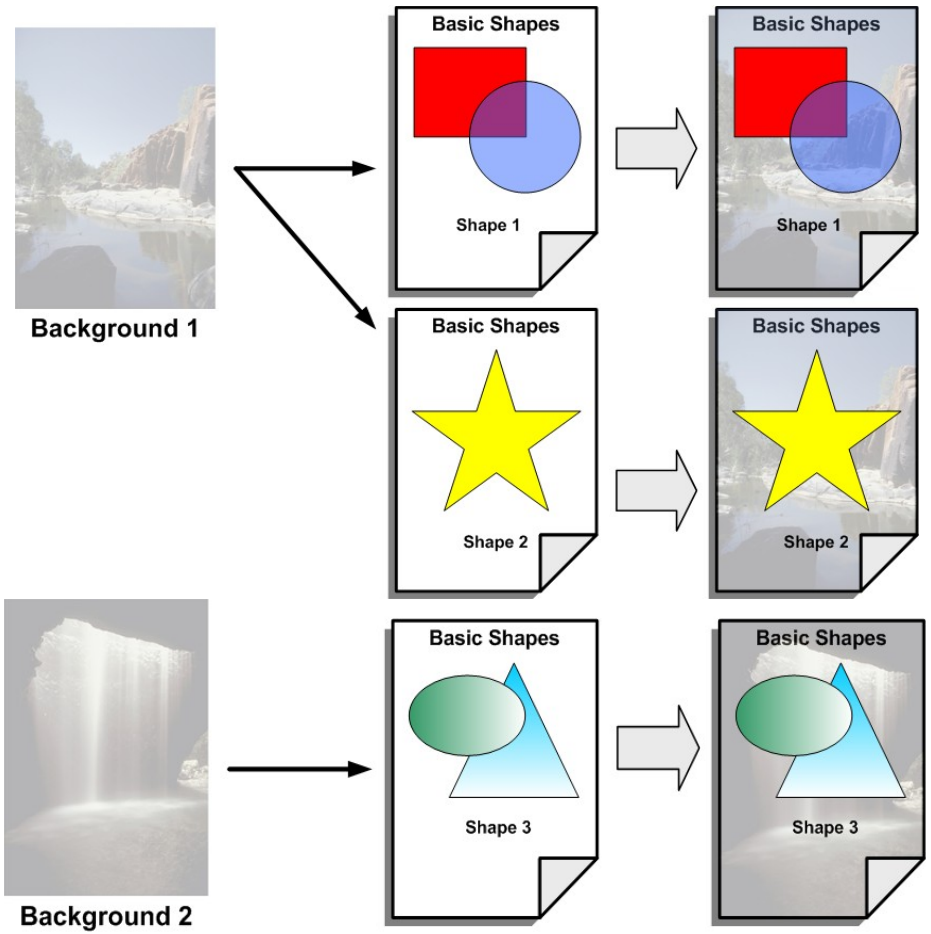
- The **`rendering-order`** attribute sets type
  - Use "**`over`**" for Foreground Master Page
  - Use "**`under`**" for Background Master Page

Master Pages

```
<svg width="100%" height="100%" viewBox="0 0 800 600"
  xlmns="http://www.w3.org/2000/SVG" xmlns:xlink="http://www.w3.org/1999/xlink">

  <pageSet>
    <masterPage rendering-order="under">
      <image x="0" y="0" width="100%" height="100%" xlink:href="background.png"/>
      <text x="10" y="10" font-size="14">Basic Shapes<text>
    </masterPage>

    <masterPage rendering-order="over">
      <g transform="rotate(-60)">
        <text x="10" y="150" fill="red" fill-opacity="0.3" font-size="30">
          DRAFT
        <text>
      </g>
    </masterPage>

    <page><!-- Content for first page --></page>

    <page><!-- Content for second page --></page>
  </pageSet>
</svg>
```

Master Pages

- What if we want to change the background half way through?



Master Pages

```
<?xml version="1.0" encoding="utf-8"?>
<svg width="100%" height="100%" viewBox="0 0 800 600"
  xlmns="http://www.w3.org/2000/SVG" xmlns:xlink="http://www.w3.org/1999/xlink">

  <pageSet>
    <masterPage>
      <image x="0" y="0" width="100%" height="100%"
        xlink:href="background.png"/>
      <text x="10" y="10" font-size="14">Basic Shapes<text>
    </masterPage>

    <page><!-- Content for Page 1 --></page>

    <page><!-- Content for Page 2 --></page>

    <masterPage>
      <image x="0" y="0" width="100%" height="100%"
        xlink:href="background2.png"/>
      <text x="10" y="10" font-size="14">Basic Shapes<text>
    </masterPage>

    <page><!-- Content for Page 3 --></page>
  </pageSet>
</svg>
```

1st Background Master Page

Page 1

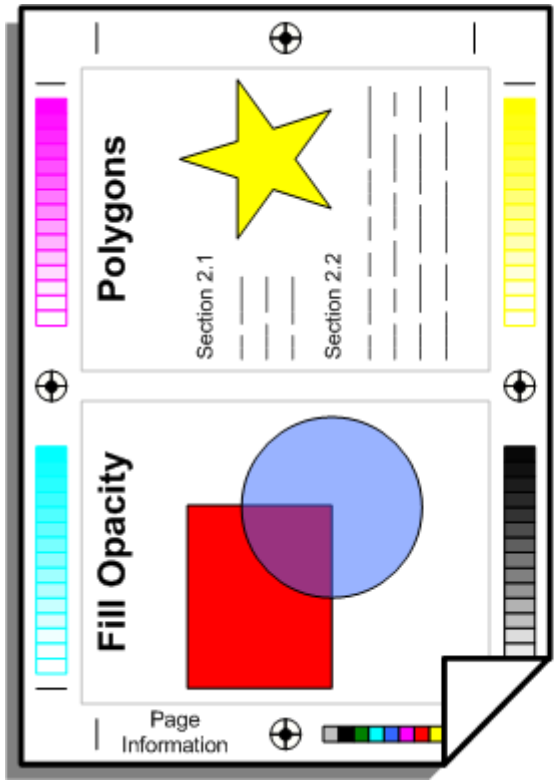
Page 2

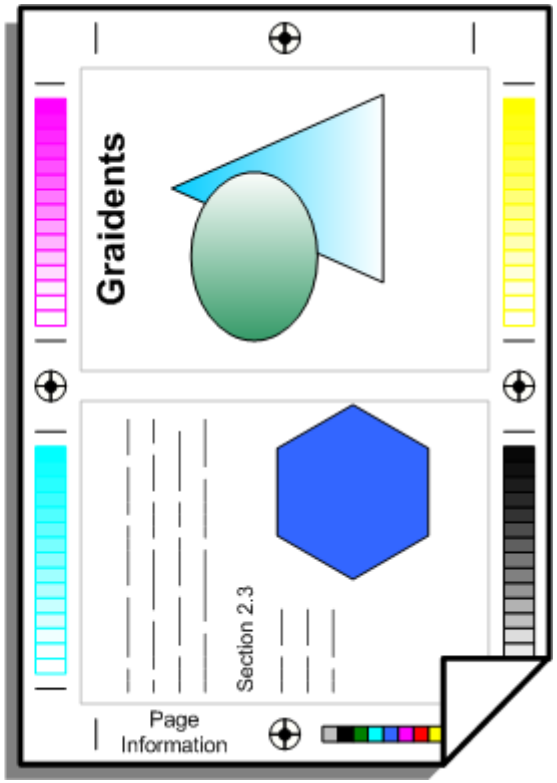
2nd Background Master Page

Page 3

Advanced Printing

- How we do this?





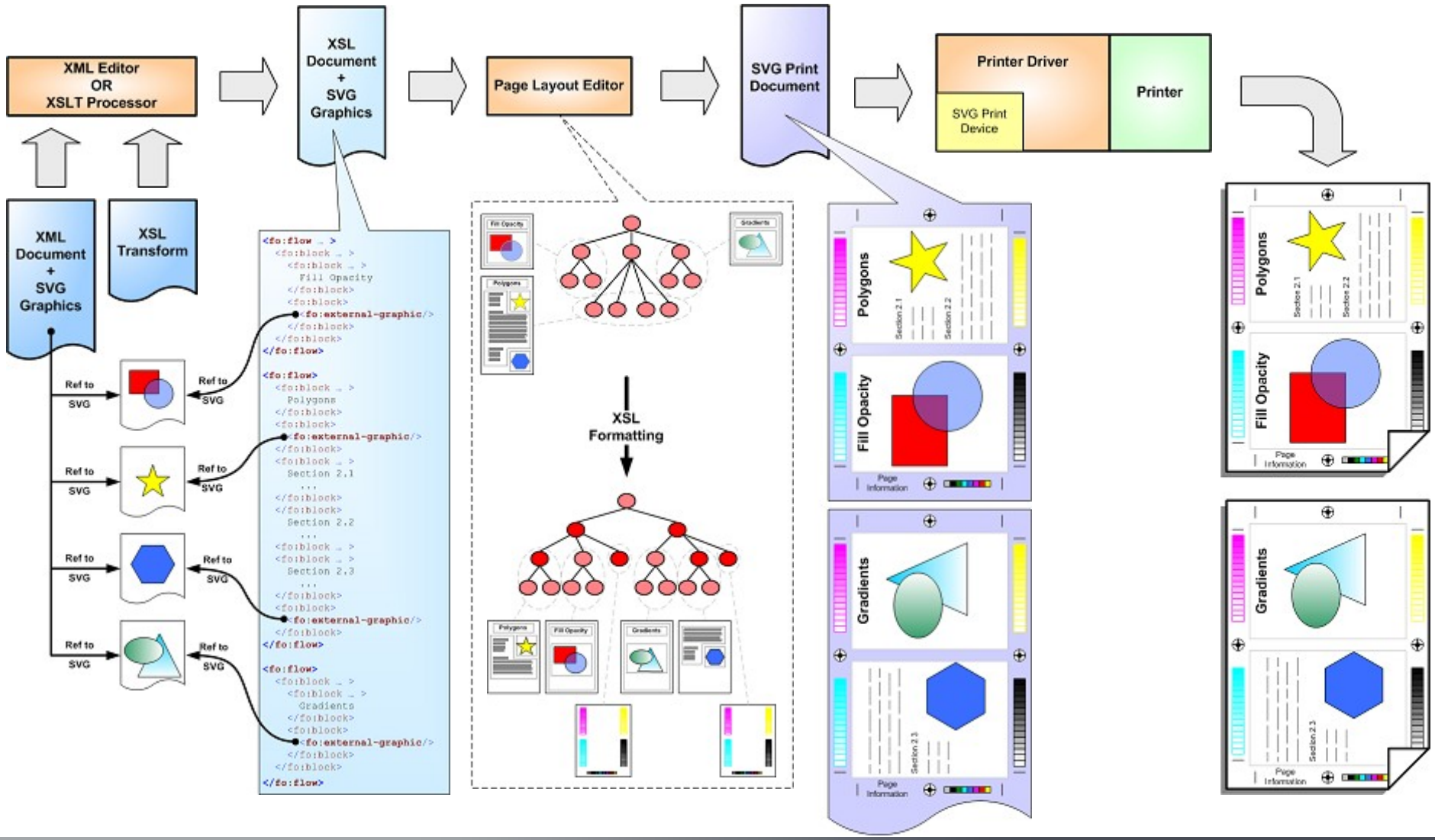
Page Description Languages

- SVG Print provides basic pagination
- Page Description Languages (PDLs) specify the layout of a page
  - CSS, PDF
- SVG Print is **deliberately not** a complete PDL in itself
  - Use a separate PDL to define page layout

SVG and XSL

- XSL and XSLT can provide page layout for XML documents
- XSL – Extensible Stylesheet Language W3C
  - Formatting of XML data for physical medium output
  - Can reference external graphics e.g. SVG
- XSLT – XSL Transform W3C
  - Convert XML data to another format
  - Can rearrange XML data
- **Combining SVG with XSL can be powerful**

SVG and XSL workflow

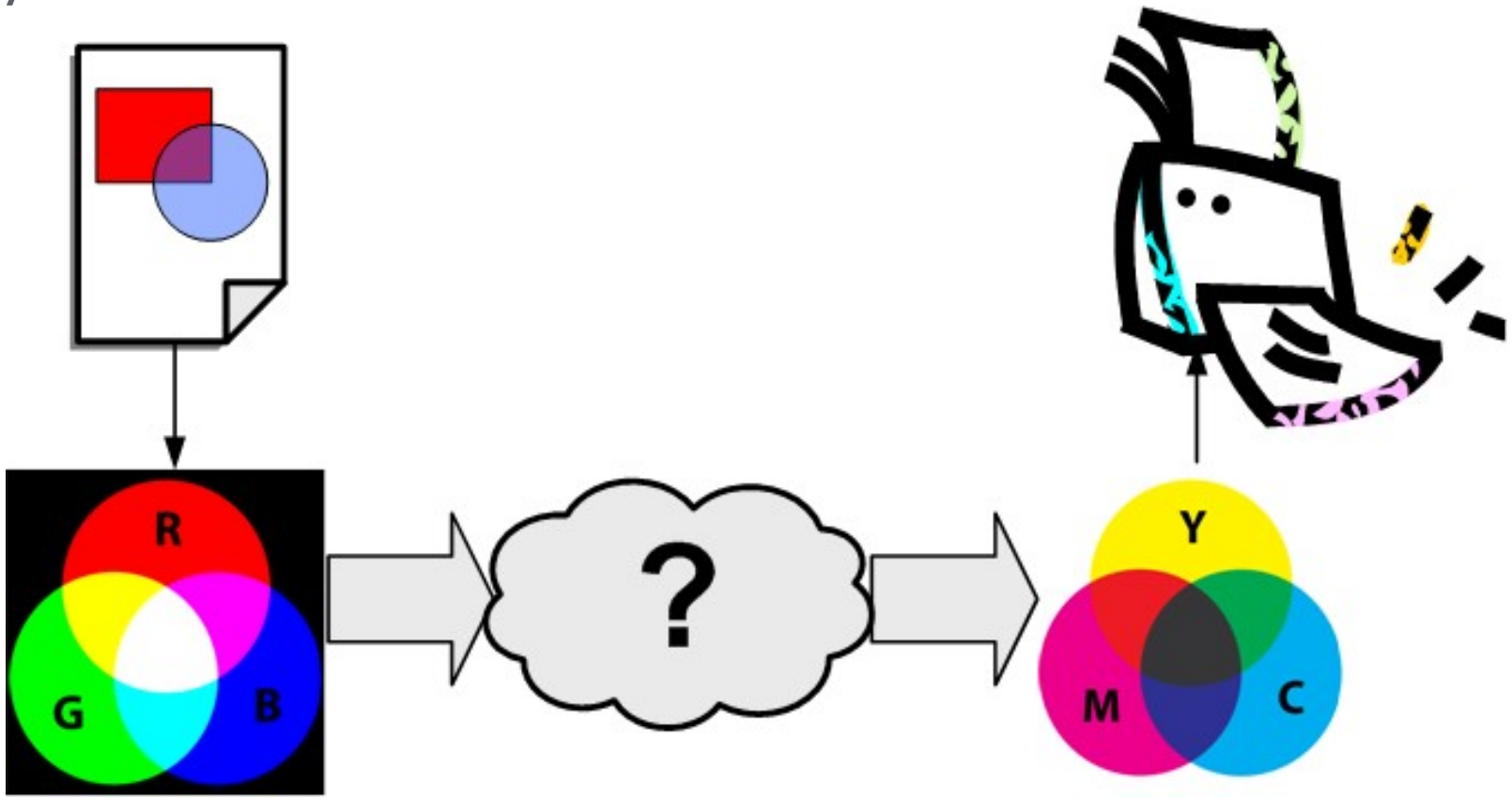


SVG and XSL workflow

- SVG Print documents are created as part of the work flow
- SVG + XSL High end workflow useful for XML documents
  - E.g. DocBook files
- Page Layout Editor requires a plug-in or native SVG Print support

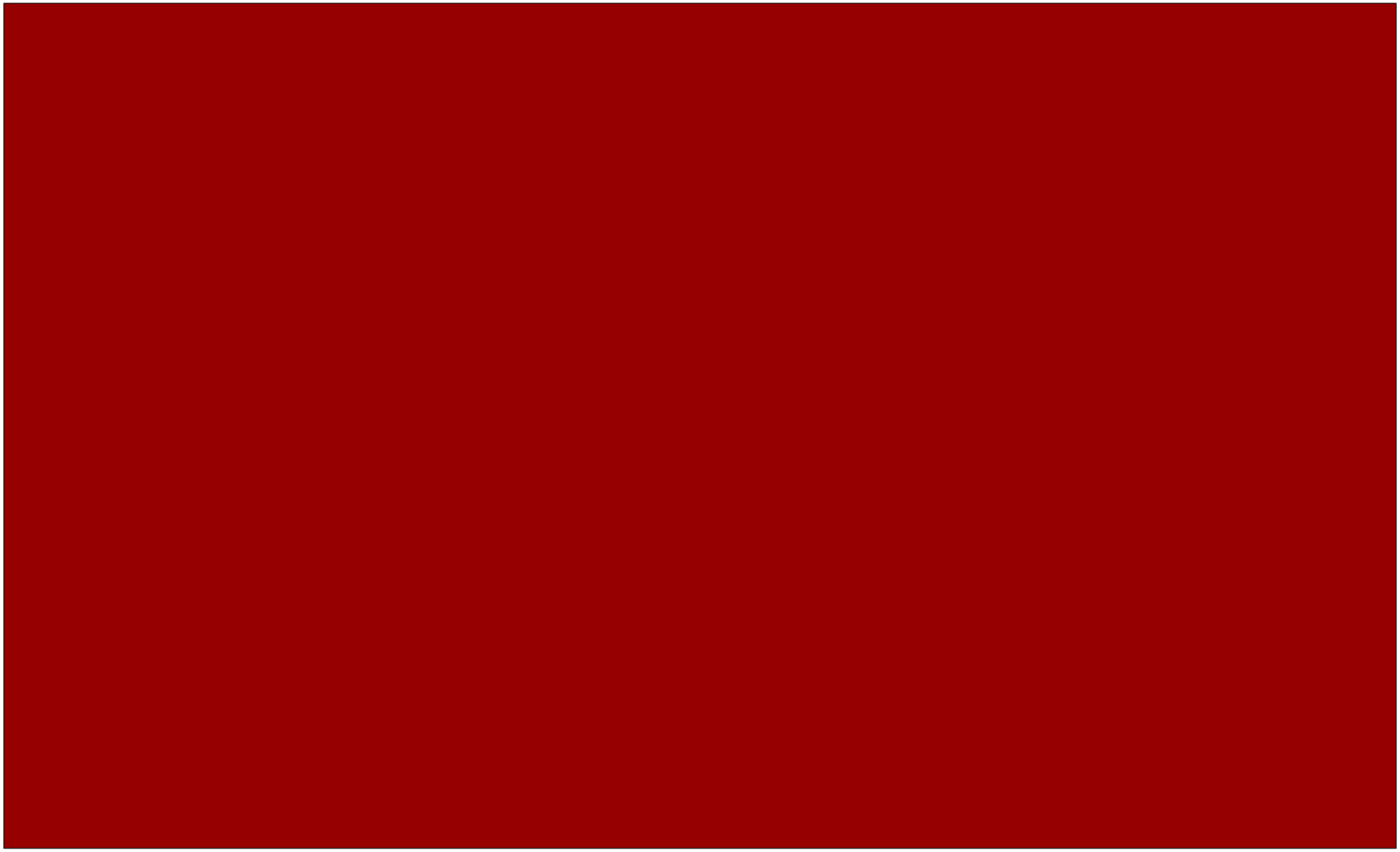
Preserving Color in Print

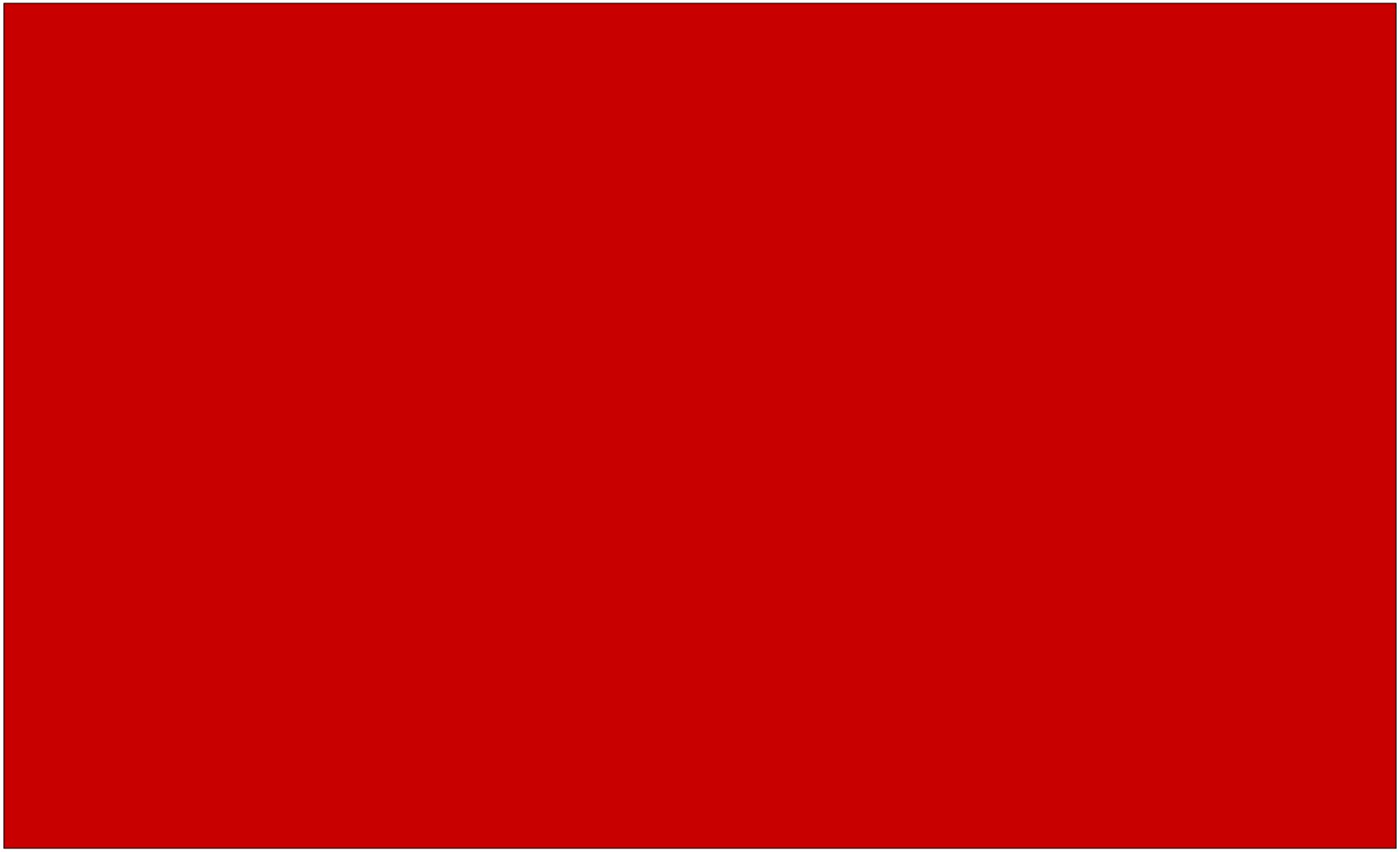
- How do guarantee the printer produces the same colors you see?

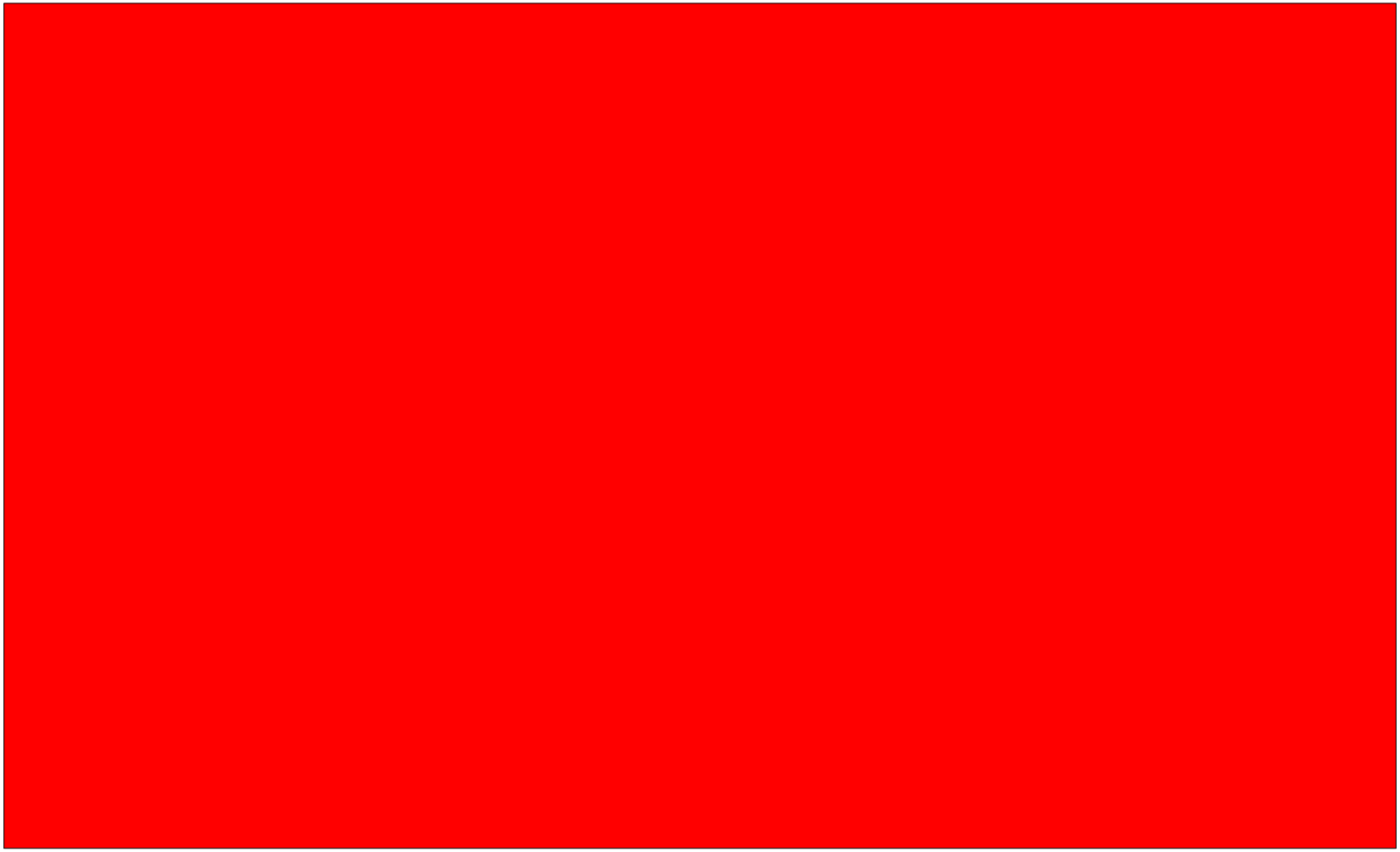


Color Definitions

- Color Space defined by
  - Color Model
  - Gamut
- Color Model
  - Abstract mathematical model (e.g. function, look up table)
  - Allows colors to be defined as a vector
- Gamut
  - The boundary of colors definable in the Color Space

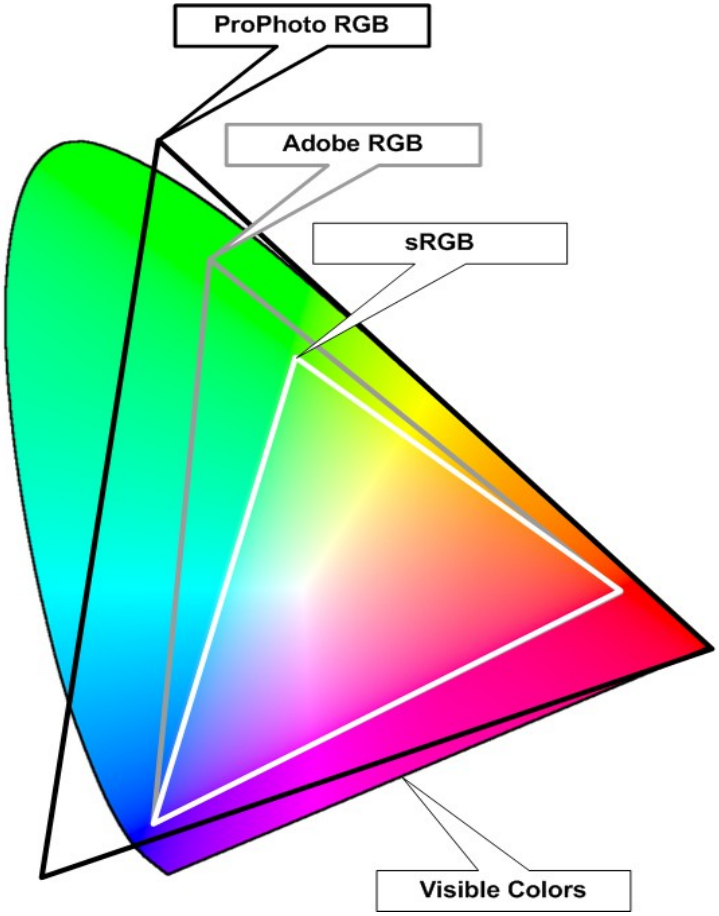






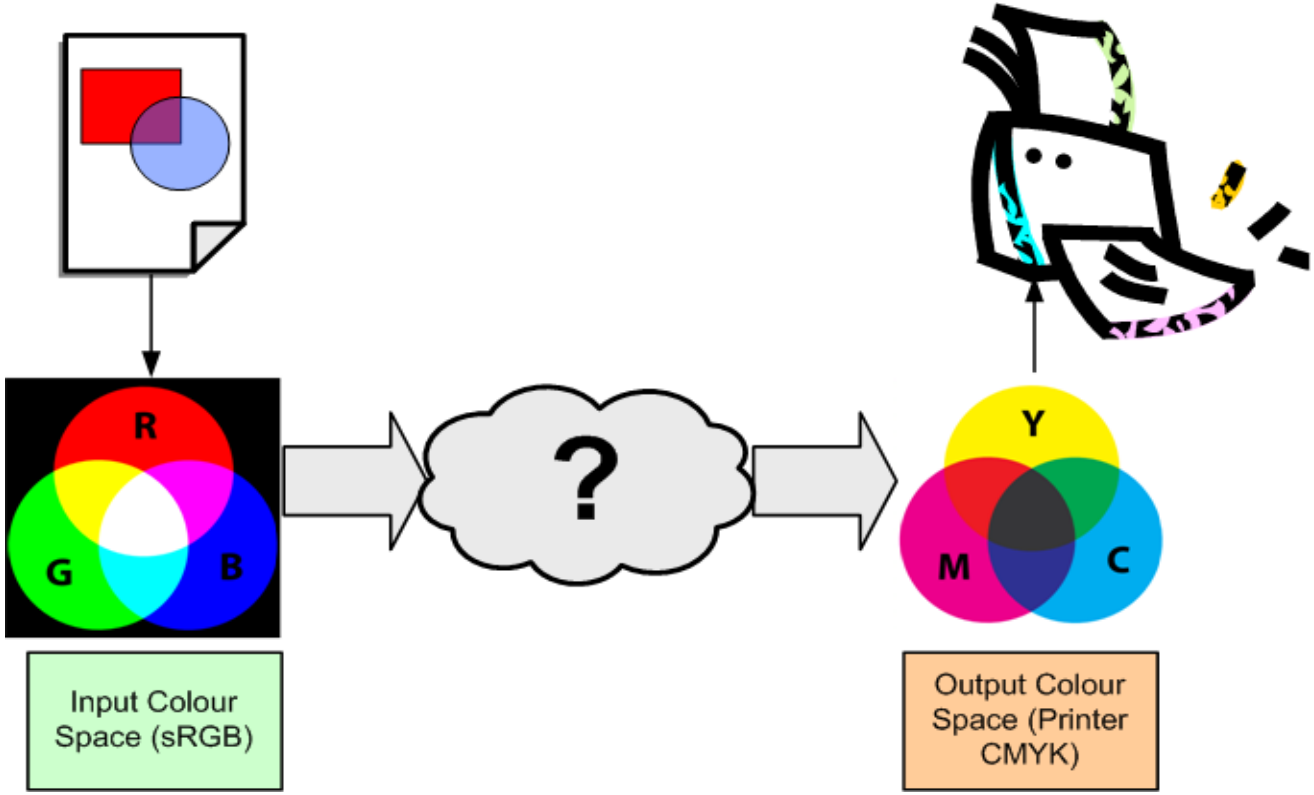
Color Spaces

- What does it mean to say a color is in RGB?



Color Translation

- Input Color Space is Color Space of Document
- Output Color Space is Color Space of Printer



Color Translation

- How do you translate color from Input to Output color space?
  - Mapping directly between devices is not practical
  - Number Input Devices x Number Output Devices
  - Need a common color space to connect Input and Output color spaces
- If...
  - Every input color space has a mapping to a commonly defined color space, and
  - Every output color space has a mapping from a commonly defined color space
- Then all devices could work together!

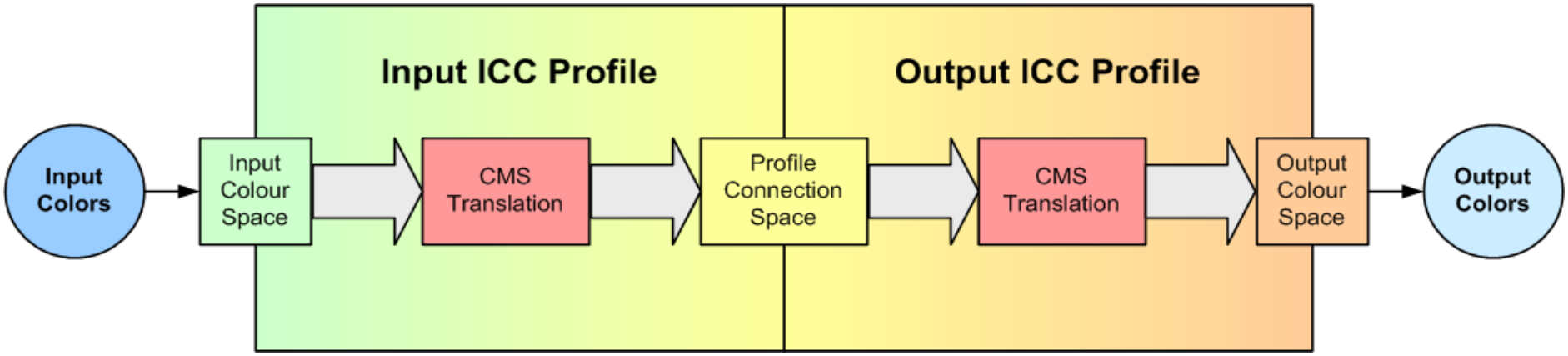
Color Translation

- The common profile space is called a **Profile Connection Space**

- An **ICC Profile** defines mappings
  - From Input Color Space to Profile Connection Space
  - From Profile Connectin Space to Output Color Space

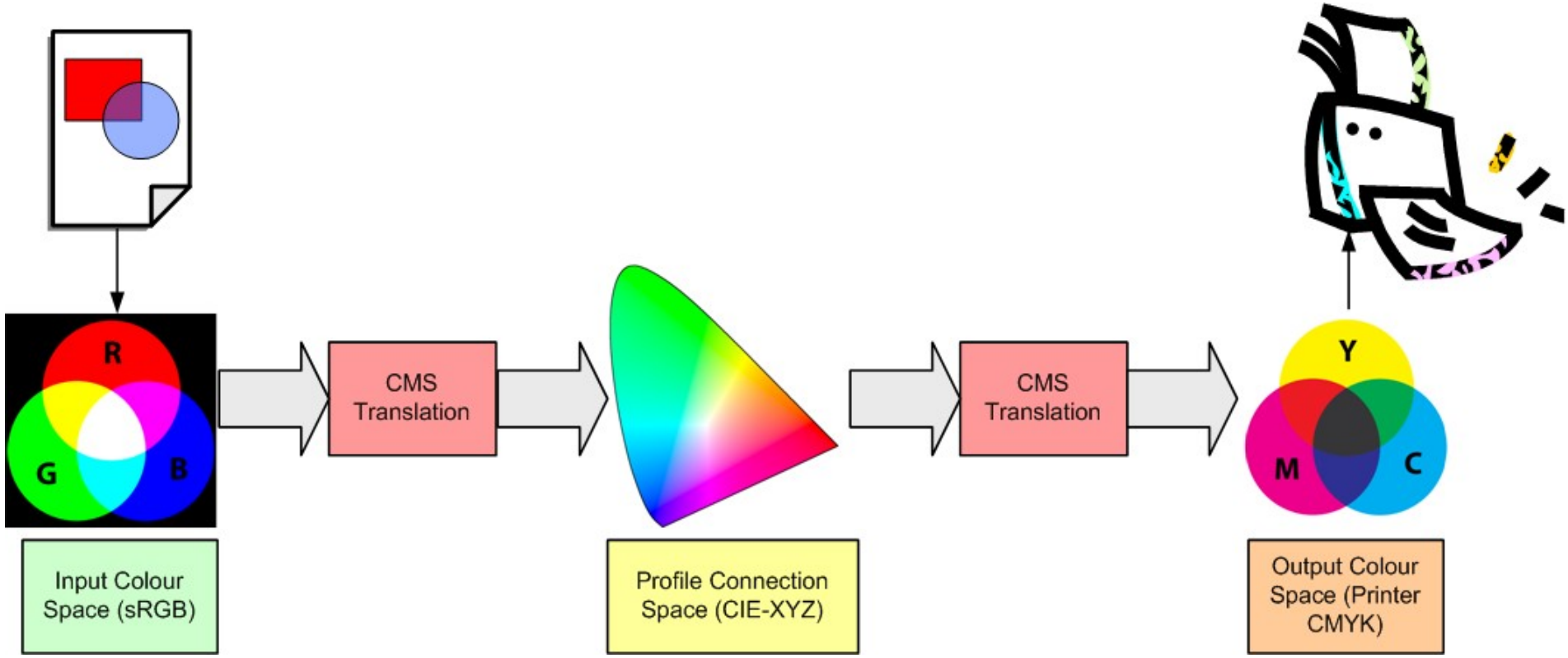
Color Translation

- A Color Management System performs the translation
- General steps when translating a color:



Color Translation

- Back to original problem
  - Steps to translate a color between a document and printer



ICC Color

- In SVG by default colors are assumed to be sRGB

- SVG Print allows input colors to be defined in other color spaces
  - Use **`color-profile`** element to define an input color profile
  - Use "**`icc-color`**" and "**`icc-named-color`**" values to specify fill color

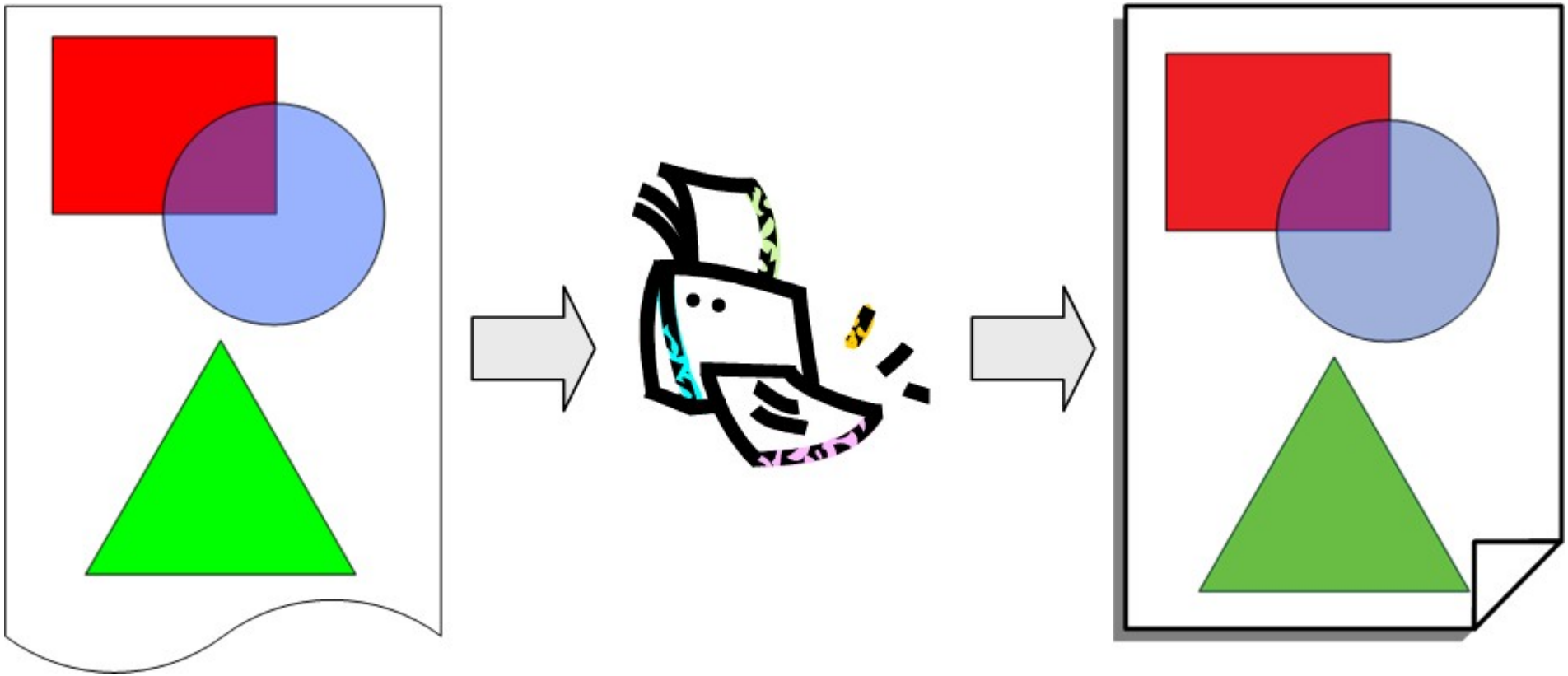
ICC Color

```
<?xml version="1.0" encoding="utf-8"?>
<svg width="100%" height="100%" viewBox="0 0 800 600"
  xlmns="http://www.w3.org/2000/SVG"
  xmlns:xlink="http://www.w3.org/1999/xlink">

  <defs>
    <color-profile name="labProfile" xlink:href="lab.icc" />

    <color-profile name="namedColorProfile"
      xlink:href="namedColor.icc" />
  </defs>

  <rect width="100" height="100" x="40" y="40"
    fill="red, icc-color(labProfile, 0.85, 0.1, 0.1)" />

  <rect width="100" height="100" x="70" y="70"
    fill="yellow,
    icc-named-color(namedColorProfile, company_color)" />
</svg>
```

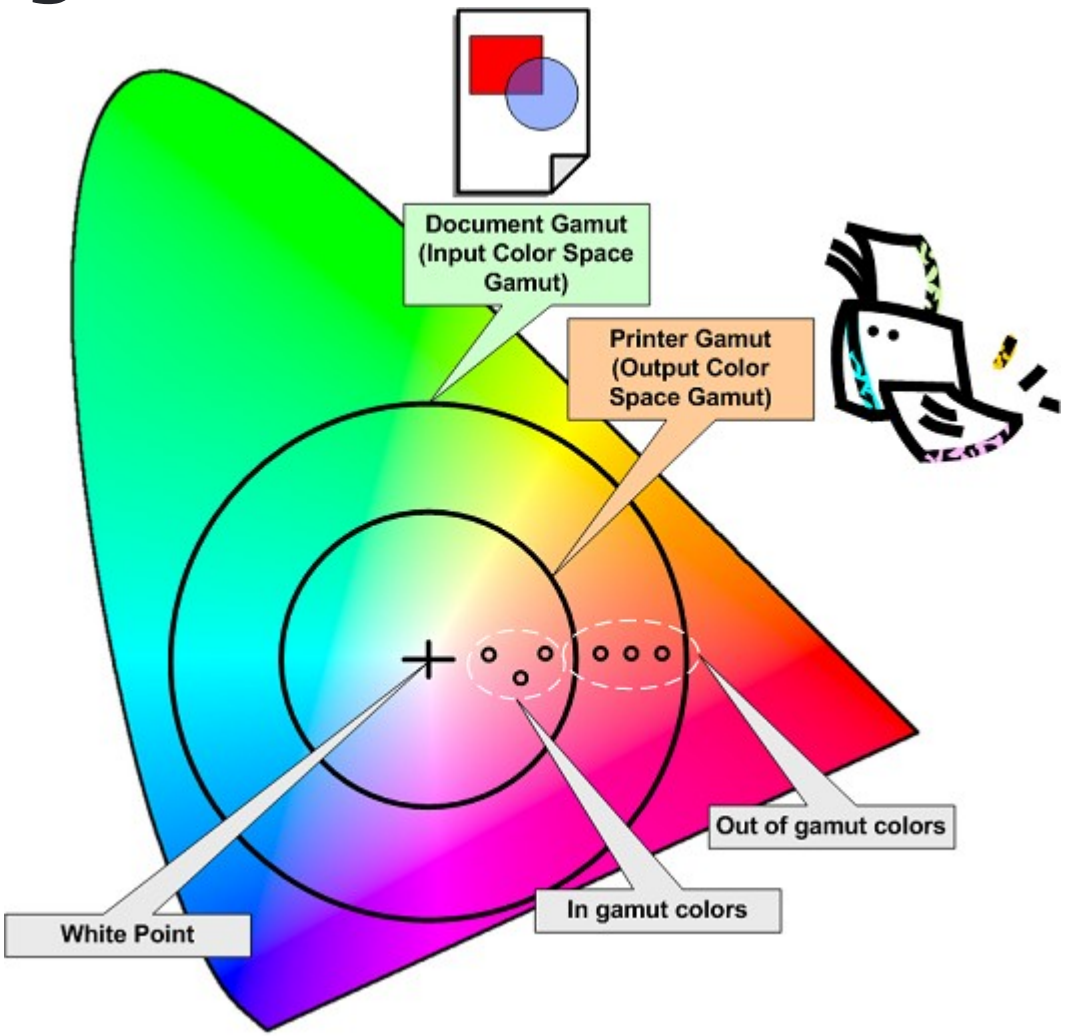
ICC Color

- The "**`icc-color`**" value allows colors to be used from a standard ICC Profile
- The "**`icc-named-color`**" value allows an input name
- Named colors are more versatile
  - Can reproduce color directly (no translation)
  - Colour can be simulated if unavailable

Printing out of gamut colors



Rendering Intents



Rendering out of Gamut Colors

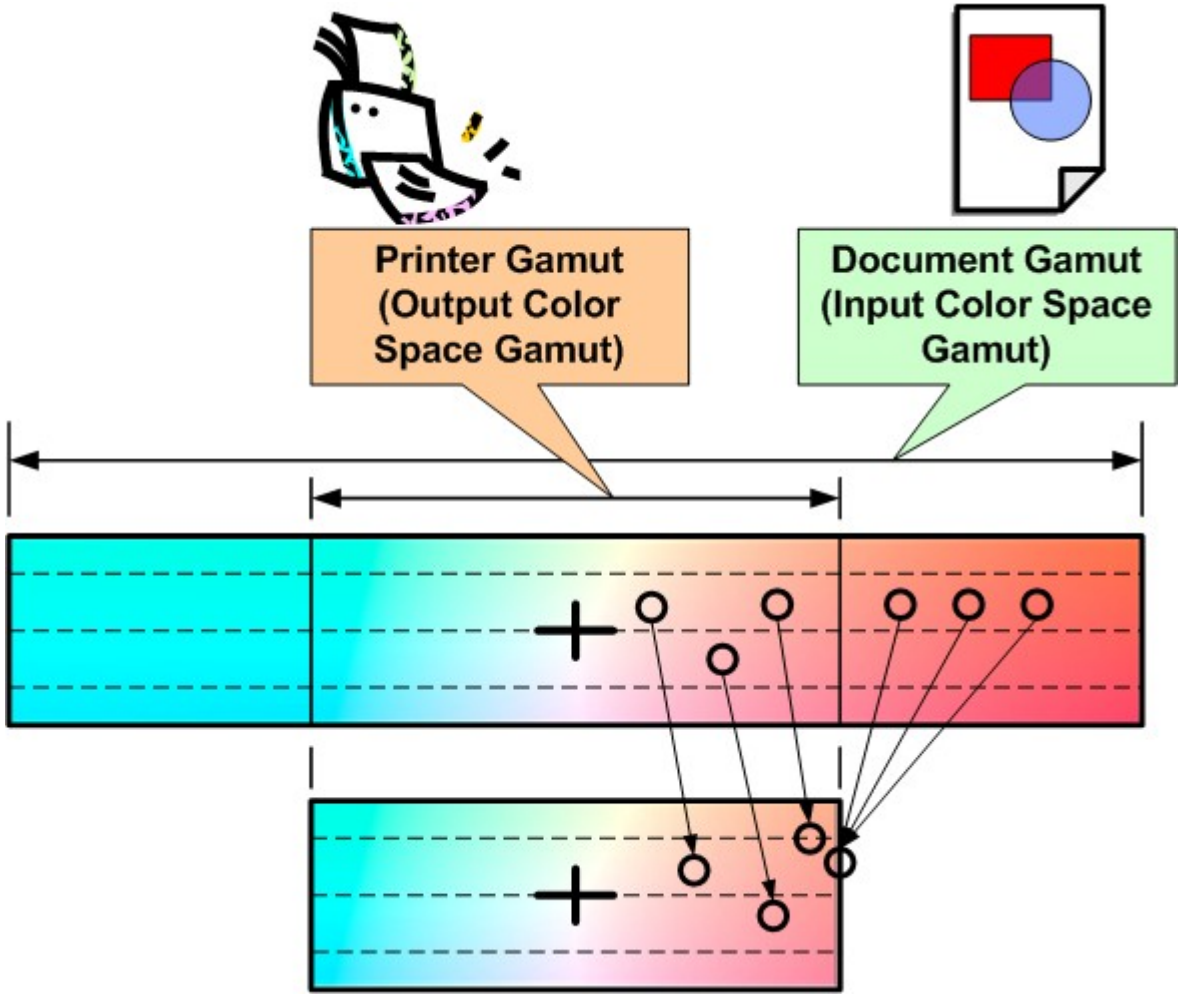
- "**icc-color**" and "**icc-named-color**" values need to be translated to the output color space

- Use rendering intent to control how color is translated
  - Specify **rendering-intent** on **color-profile** element

- Values for **rendering-intent** are
  - "**auto**" (Default value)
  - "**saturation**"
  - "**perceptual**"
  - "**relative-colorimetric**"
  - "**absolute-colorimetric**"

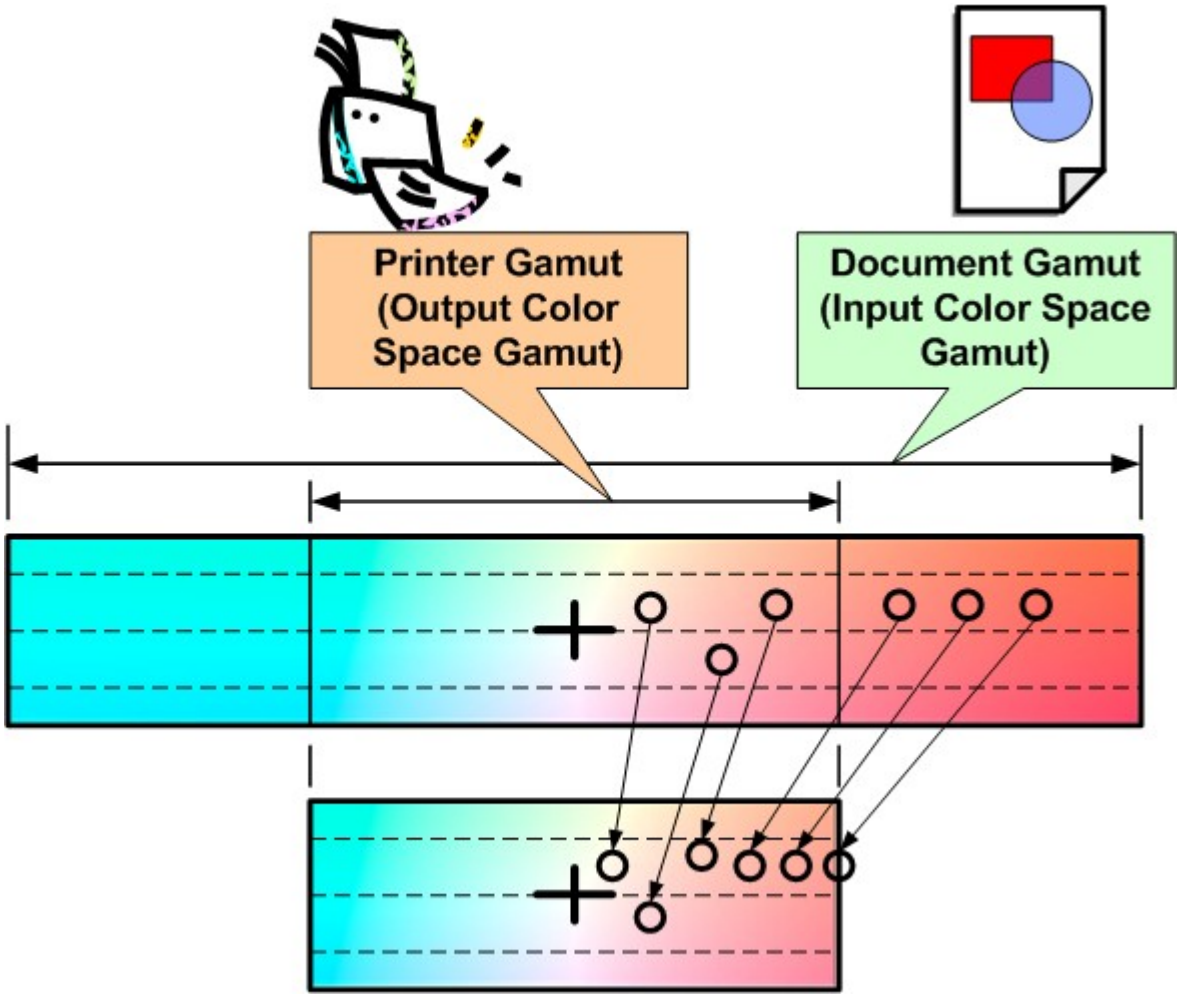
Rendering Intents - Saturation

- Colors move to edge of gamut
- Saturates color
- Uses
  - Graphics
  - Artwork
  - Improving weak images



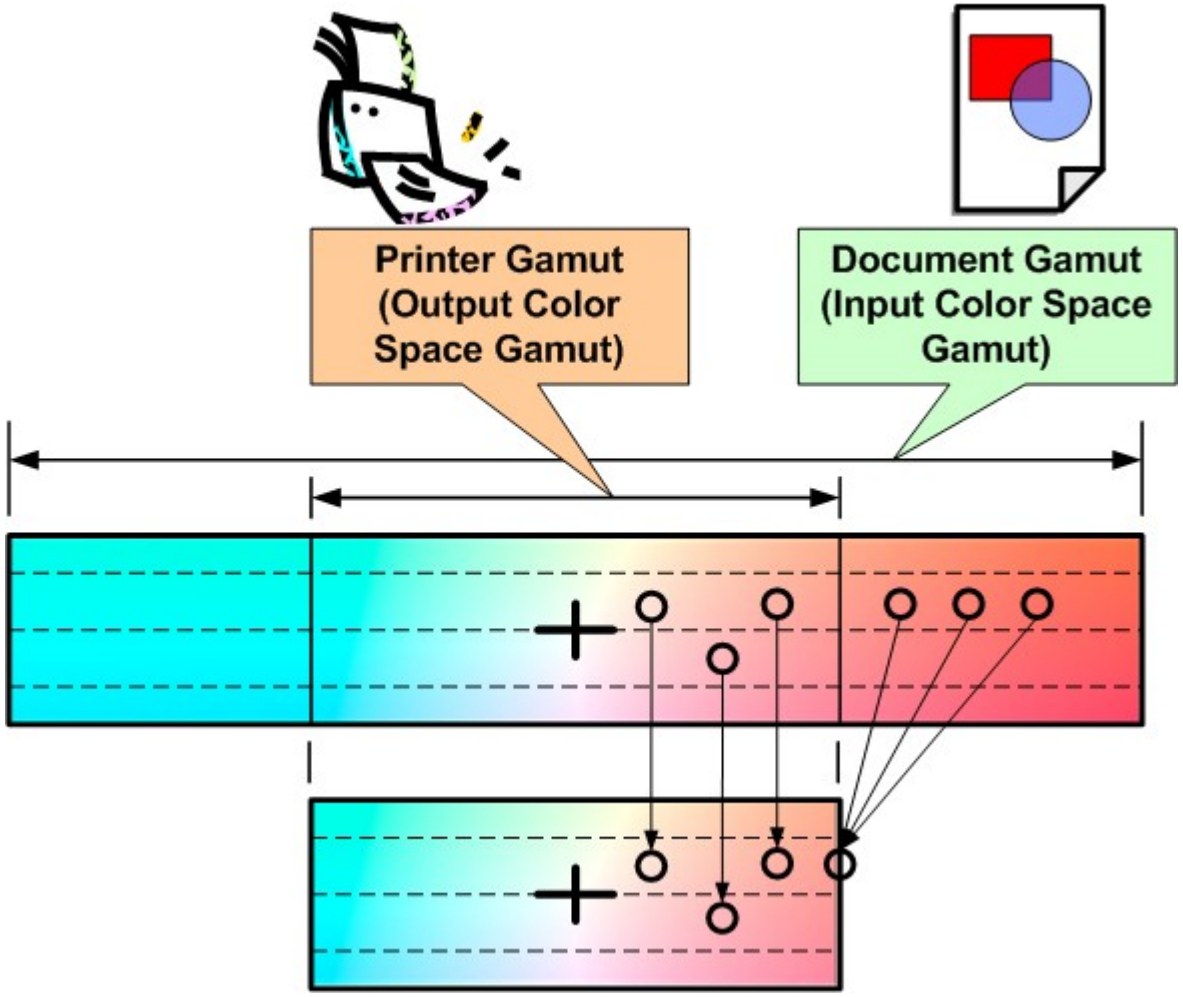
Rendering Intents - Perceptual

- Colors maintain relative distance when translated
- Uses
  - Natural images
  - Every day printing



Rendering Intents - Colorimetric

- Colors inside the gamut are preserved
- Colors outside gamut are translated to nearest match
- Uses
  - Translation between like color spaces
  - Preserving color of a logo



Rendering Intents - Colorimetric

- Relative Colorimetric
  - White point is mapped from input color space to output color space
  - Good for final print reproduction
- Absolute Colorimetric
  - Uses white point of input color space
  - Good for print preview

Device Color

- SVG Print allows Device Color input
- Data specified is only meaningful to the output device
- No ICC Profile describing input color
  - Unable to transform color if not recognized
  - Fallback color is used

Device Color

- Use **`deviceColor`** element to specify data about color
  - Element is namespace specific
- Use "**`device-color`**" values to specify fill color

Device Color

```
<svg width="100%" height="100%" viewBox="0 0 800 600"
  xlmns="http://www.w3.org/2000/SVG"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:spot="http://www.example.com/ink/spot" >

  <defs>
    <deviceColor name="processInks"
      xmlns:hexachrome="http://www.example.com/ink/hexachrome"
      hexachrome:valueCount="6"
      hexachrome:value="Cyan, Magenta, Yellow, Black, Orange,
      Green" />

    <color-profile name="cmykProfile" xlink:href="cmyk.icc" />
  </defs>

  <rect width="100" height="100" x="100" y="0 fill="steelblue,
   icc-color(cmykProfile, 0.8, 0.17, 0.1, 0.2),
              device-color(processInks, 70, 20, 10, 10, 0, 10)" />
</svg>
```

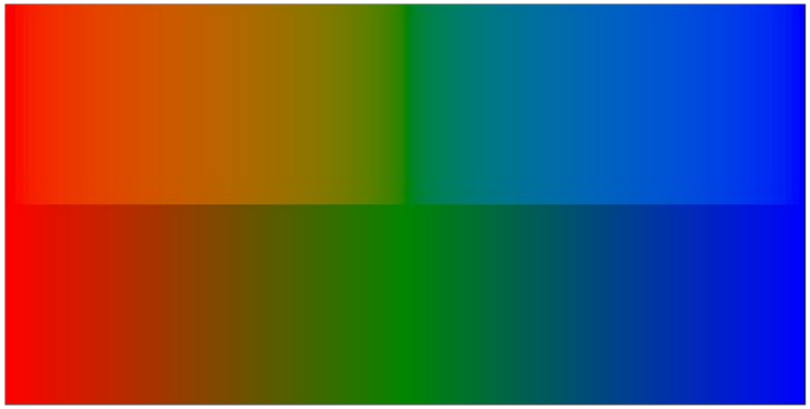
Fallback Color

- Should specify sRGB fallback color if "**`icc-color`**" or "**`icc-named-color`**" values used

- May not be able to load ICC Profile or translate color
  - If no fallback specified in this case, painting will not occur

- Should specify ICC and sRGB fallback color if "**`device-color`**" value used
  - If unavailable could be better simulated with an ICC color

Color Interpolation



interpolation-color set to linearRGB

interpolation-color set to sRGB color space

Color Interpolation

- Specifies the Color Space to composite or interpolate in
  - Can give a more visually appealing result

- Values for **`color-interpolation`** are
  - "**`auto`**"
  - "**`sRGB`**" (Default value)
  - "**`linearRGB`**"
  - "**`CIE-Lab`**"
  - "**`CIE-LCHab`**"

Color Interpolation

- Use **`color-interpolation`** property on
  - Group element
  - Gradient element

- Colors will be composited/interpolated in **`color-interpolation`** space
  - Colors in alternative color spaces will be translated to color interpolation space
  - Resultant is converted back to parent color space

Color Interpolation

```
<svg xlmns="http://www.w3.org/2000/SVG"
    width="100%" height="100%" viewBox="0 0 800 600">

  <defs>
    <!-- Gradient will be inerpolated in linearRGB -->
    <linearGradient id="linearRGBGradient" color-interpolation="linearRGB"
      gradientUnits="objectBoundingBox">

      <stop offset="0.1" stop-color="red"/>
      <stop offset="0.9" stop-color="green"/>
    </linearGradient>
  </defs>

  <rect width="100" height="100" x="10" y="10" fill="blue" />

  <g color-interpolation="linearRGB">
    <!-- Objects in group composited in linearRGB -->
  </g>

  <!-- Fill is linearRGB Gradient-->
  <rect fill="url(#linearRGBGradient)"
    width="200" height="50" x="10" y="200"/>
</svg>
```

Conclusion

- What does SVG Print offer?

- The two most requested features:
  - Support for high quality color printing
  - Pagination