

# Encapsulating and Manipulating Component Object Graphics (COGs) using SVG

**Alexander Macdonald**

**David Brailsford**

**Steven Bagley**



The University of  
**Nottingham**

2  
DocEng  
0  
5

# Why we use PDF

- Word / Latex don't guarantee portability
- Different versions have different features
- Fonts not embedded
- PDF provides a high quality final form document format

# Why we wish we didn't use PDF

- People want editability
- Editability is hard without document structure
- Finest level of granularity in PDF is the page

# The COG solution

- Graphical objects are encapsulated as COGs
- Each COG is self contained and can't affect other COGs
- A COG-PDF document contains a sequence of COG definitions and spacers to position them

# The benefits of COGs

- Increase the granularity of PDF documents
- COGs can be moved, added or removed from documents
- COGs can be programatically manipulated

# Scalable Vector Graphics (SVG)

- W3C's XML based vector graphics language

```
<svg width="151" height="21">  
  <rect fill="lightblue" width="150" height="20" stroke-width="2"  
  stroke="red"/>  
  <text y="18" font-family="Arial" font-size="20">Hello World!</text>  
</svg>
```



Hello World!

# Scalable Vector Graphics (SVG)

- Rendering model similar to Postscript / PDF
- Lots of web-centric features
- Version 1.2 will add support for pagesets

# SVG vs PDF

- A page in PDF is a single stream of commands
- Being XML based, SVG has a tree structure
- Groups allow for more structure



# Extracting graphical content

```
<svg width="151" height="21">  
  <g fill="lightblue">  
    <g font-family="Arial">  
      <rect width="150" height="20" stroke-width="2" stroke="red"/>  
      <text y="18" fill="black" font-size="20">Hello World!</text>  
    </g>  
  </g>  
</svg>
```



Hello World!

# Extracting graphical content

- Inherited attributes must be flattened onto group
- Unspecified attributes must be specified on group

# Extracting graphical content

- All rendering occurs on the canvas
- Bounds of the viewport define the viewable subset of the canvas
- SVG allows nesting of documents

# Extracting graphical content

- “ref” is the inverse of current transformation matrix
- `<g>` doesn't create a new viewport so can't be used for encapsulation
- `<svg>` does create a new viewport

# SVG COGs

- `<defs>` is used to store the COG definition
- `<use>` element can reference another part of the document

# SVG COGs

```
<svg width="200" height="100">  
  <defs>  
    <svg width="151" height="21" id="cog1">  
      <rect fill="lightblue" width="150" height="20" stroke-width="2"  
stroke="red"/>  
      <text y="18" font-family="Arial" fill="black" font-size="20">Hello World!</  
text>  
    </svg>  
  </defs>  
  
  <use xlink:href="#cog1" x="10" y="10"/>  
  
  <use xlink:href="#cog1" x="20" y="30" transform="rotate(15)"/>  
</svg>
```

Hello World!

Hello World!

# Conclusion

- SVG makes it easier to retain structure, but does not enforce it
- COGs provide a standardised way of encapsulating graphical objects