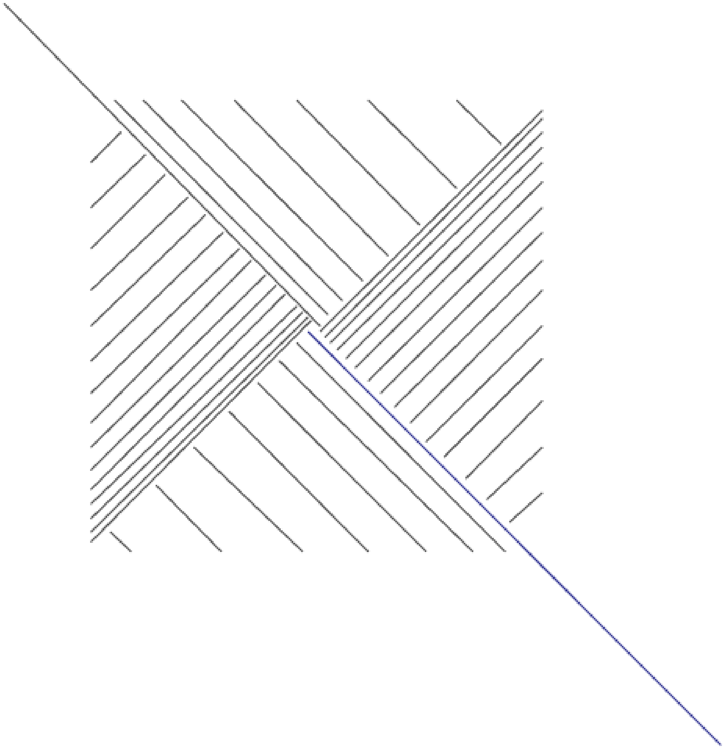
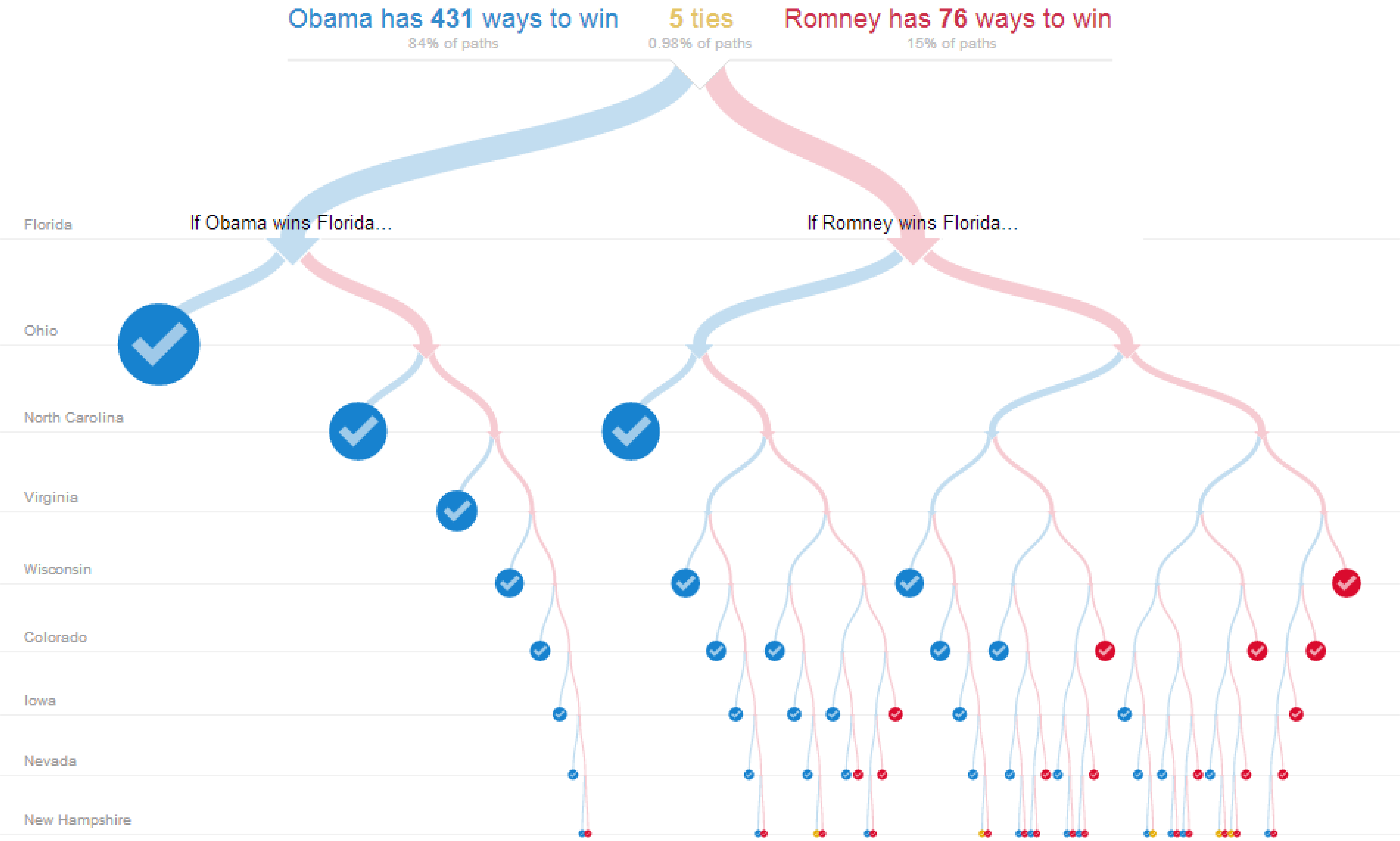


8

Getting To Know Vector Graphics

The building blocks of your soon-to-be-visualizations

Kevin McVey
CS1501 – Fall 2013



http://www.nytimes.com/interactive/2012/11/02/us/politics/paths-to-the-white-house.html

Need more JavaScript resources?

Need more JavaScript resources?

Learn JavaScript the fun way!
http://www.codecademy.com/tracks/javascript

Codecademy

Need more JavaScript resources?

Learn JavaScript the fun way!
http://www.codecademy.com/tracks/javascript

Slightly more advanced, HTML and CSS knowledge necessary
http://www.w3schools.com/js/

Finally, let's look at some homework solutions

Iterative solution by Shannon O’Donnell

```
var factorial = function(number) {
    var product = 1;
    for( number; number >= 1; --number ) {
        product = product * number;
    }
    return product;
};
```

Iterative solution by Shannon O’Donnell

```
var factorial = function(number) {
   var product = 1;
   for( number; number >= 1; --number ) {
      product = product * number;
   }
   return product;
};
```

Recursive solution by Kevin Chen

```
function factorial(n) {
   return (n==1)? 1:(n*factorial(n-1));
}
```

Iterative solution by Shannon O’Donnell

```
var factorial = function(number) {
   var product = 1;
   for( number; number >= 1; --number ) {
      product = product * number;
   }
   return product;
};
```

Recursive solution by Kevin Chen

```
function factorial(n) {
   return (n==1)? 1:(n*factorial(n-1));
}
```

```
(CONDITION) ? true : false;
```

The in-line if statement

```
if(CONDITION) {
   true
} else {
   false
}
```

Iterative solution by Shannon O'Donnell

```
var factorial = function(number) {
   var product = 1;
   for( number; number >= 1; --number ) {
      product = product * number;
   }
   return product;
};
```

Recursive solution by Kevin Chen

```
function factorial(n) {
   return (n==1)? 1:(n*factorial(n-1));
}
```

For loop solution by Tyler Robbins

```
function is_sum_even(numArray) {
   var sum = 0;
   for (var i = 0; i < numArray.length; i++) {
      sum += numArray[i];
   }
   if (sum%2 === 0) {
      return true;
   } else {
      return false;
   }
}
```

Iterative solution by Shannon O’Donnell

```
var factorial = function(number) {
    var product = 1;
    for( number; number >= 1; --number ) {
        product = product * number;
    }
    return product;
};
```

Recursive solution by Kevin Chen

```
function factorial(n) {
    return (n==1)? 1:(n*factorial(n-1));
}
```

For loop solution by Tyler Robbins

```
function is_sum_even(numArray) {
    var sum = 0;
    for (var i = 0; i < numArray.length; i++) {
        sum += numArray[i];
    }
    if (sum%2 === 0) {
        return true;
    } else {
        return false;
    }
}
```

=== and !==

Strict Equality

1 == true

1 === true

3 != “3”

3 !== “3”

Iterative solution by Shannon O’Donnell

```
var factorial = function(number) {
   var product = 1;
   for( number; number >= 1; --number ) {
      product = product * number;
   }
   return product;
};
```

Recursive solution by Kevin Chen

```
function factorial(n) {
   return (n==1)? 1:(n*factorial(n-1));
}
```

For loop solution by Tyler Robbins

```
function is_sum_even(numArray) {
   var sum = 0;
   for (var i = 0; i < numArray.length; i++) {
      sum += numArray[i];
   }
   if (sum%2 === 0) {
      return true;
   } else {
      return false;
   }
}
```

=== and !==

Strict Equality

1 == true	true
1 === true	false
3 != “3”	false
3 !== “3”	true

Iterative solution by Shannon O'Donnell

```
var factorial = function(number) {
    var product = 1;
    for( number; number >= 1; --number ) {
        product = product * number;
    }
    return product;
};
```

Recursive solution by Kevin Chen

```
function factorial(n) {
    return (n==1)? 1:(n*factorial(n-1));
}
```

For loop solution by Tyler Robbins

```
function is_sum_even(numArray) {
    var sum = 0;
    for (var i = 0; i < numArray.length; i++) {
        sum += numArray[i];
    }
    if (sum%2 === 0) {
        return true;
    } else {
        return false;
    }
}
```

For-each loop solution by Gage Gaskins

```
function is_sum_even(sumArray){
    var x = 0;
    for(var num in sumArray){
        x+=sumArray[num];
    }
    if(x%2==0){
        return true;
    } else {
        return false;
    }
}
```

Iterative solution by Shannon O'Donnell

```
var factorial = function(number) {
   var product = 1;
   for( number; number >= 1; --number ) {
      product = product * number;
   }
   return product;
};
```

Recursive solution by Kevin Chen

```
function factorial(n) {
   return (n==1)? 1:(n*factorial(n-1));
}
```

Hooray, success all around!

```
   var sum = 0;
   for (var i = 0; i < numArray.length; i++) {
      sum += numArray[i];
   }
   if (sum%2 === 0) {
      return true;
   } else {
      return false;
   }
}
```

```
   var x = 0;
   for(var num in sumArray){
      x+=sumArray[num];
   }
   if(x%2==0){
      return true;
   } else {
      return false;
   }
}
```

What’s RASTER and what’s VECTOR?

What’s RASTER and what’s VECTOR?

RASTER

Image is made of thousands of pixels with independent color information.

What’s RASTER and what’s VECTOR?

RASTER

Image is made of thousands of pixels with independent color information.

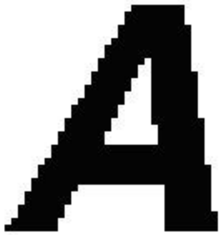
VECTOR

Formulaic. Math functions define points, lines, curves, and shapes.

What’s RASTER and what’s VECTOR?

RASTER

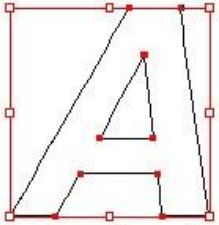
Image is made of thousands of pixels with independent color information.



Raster Graphic

VECTOR

Formulaic. Math functions define points, lines, curves, and shapes.



Vector Graphic

Of course, they each have pros and cons.

What's RASTER and what's VECTOR?

RASTER

VECTOR

What's RASTER and what's VECTOR?

RASTER ***VECTOR***

+ Detail-driven

+ Information is stored very precisely

- Do not scale up or down well

Great for photorealism

What’s RASTER and what’s VECTOR?

RASTER

+ Detail-driven

+ Information is stored very precisely

- Do not scale up or down well

Great for photorealism

VECTOR

Scale up infinitely +

Generally smaller file size +

Limited details and effects -

Great for logos and procedural graphics

What's RASTER and what's VECTOR?

RASTER

+ Detail-driven

+ Information is stored very precisely

- Do not scale up or down well

Great for photorealism

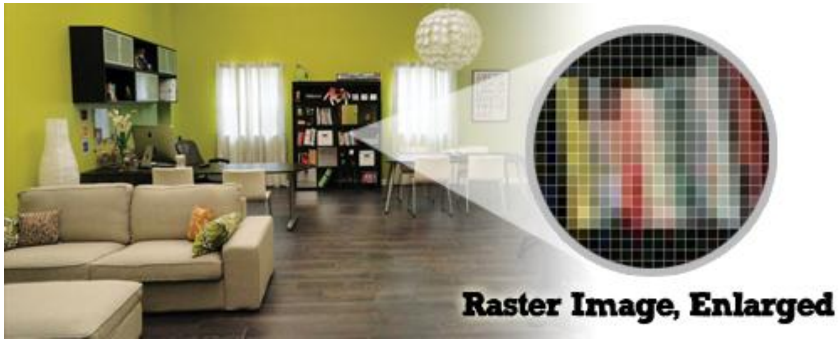
VECTOR

Scale up infinitely +

Generally smaller file size +

Limited details and effects -

Great for logos and procedural graphics



What’s RASTER and what’s VECTOR?

RASTER

+ Detail-driven

+ Information is stored very precisely

- Do not scale up or down well

Great for photorealism

But… but how?

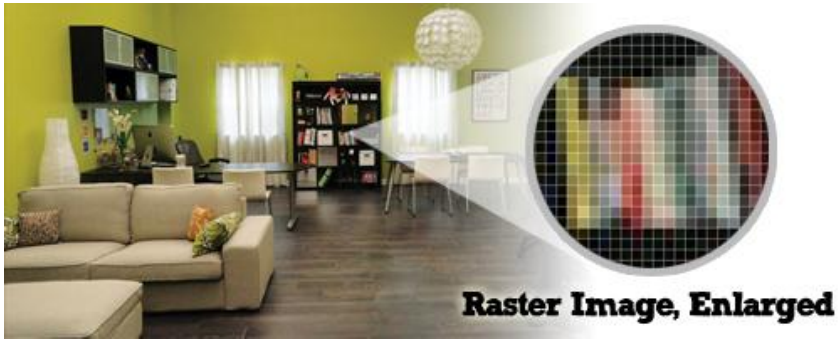
VECTOR

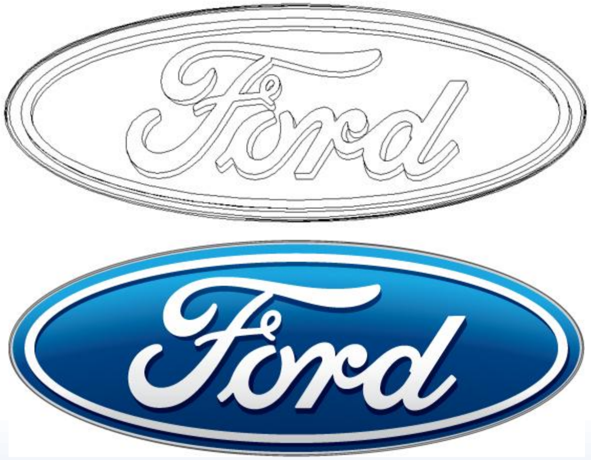
Scale up infinitely +

Generally smaller file size +

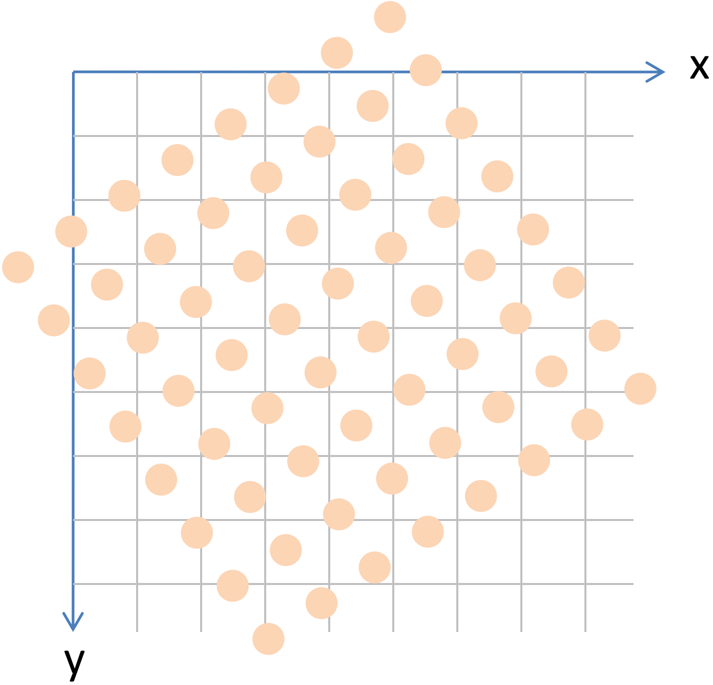
Limited details and effects -

Great for logos and procedural graphics





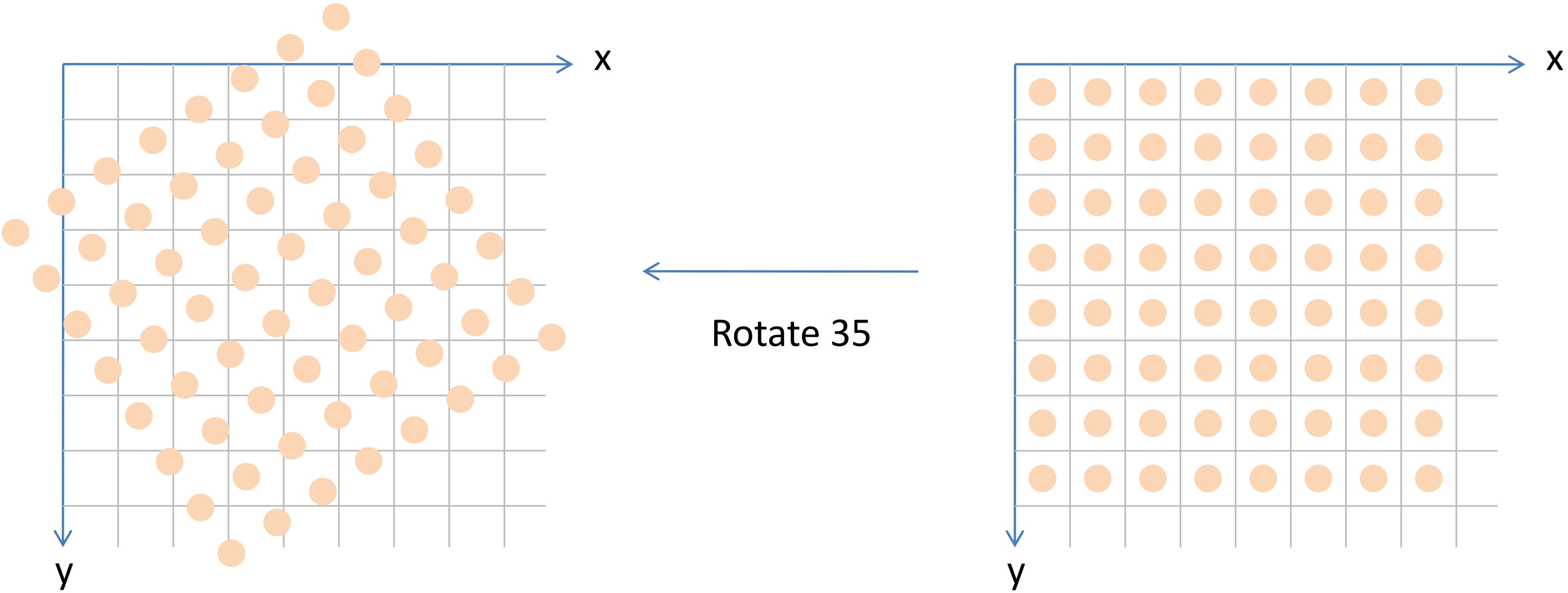
What's RASTER and what's VECTOR?



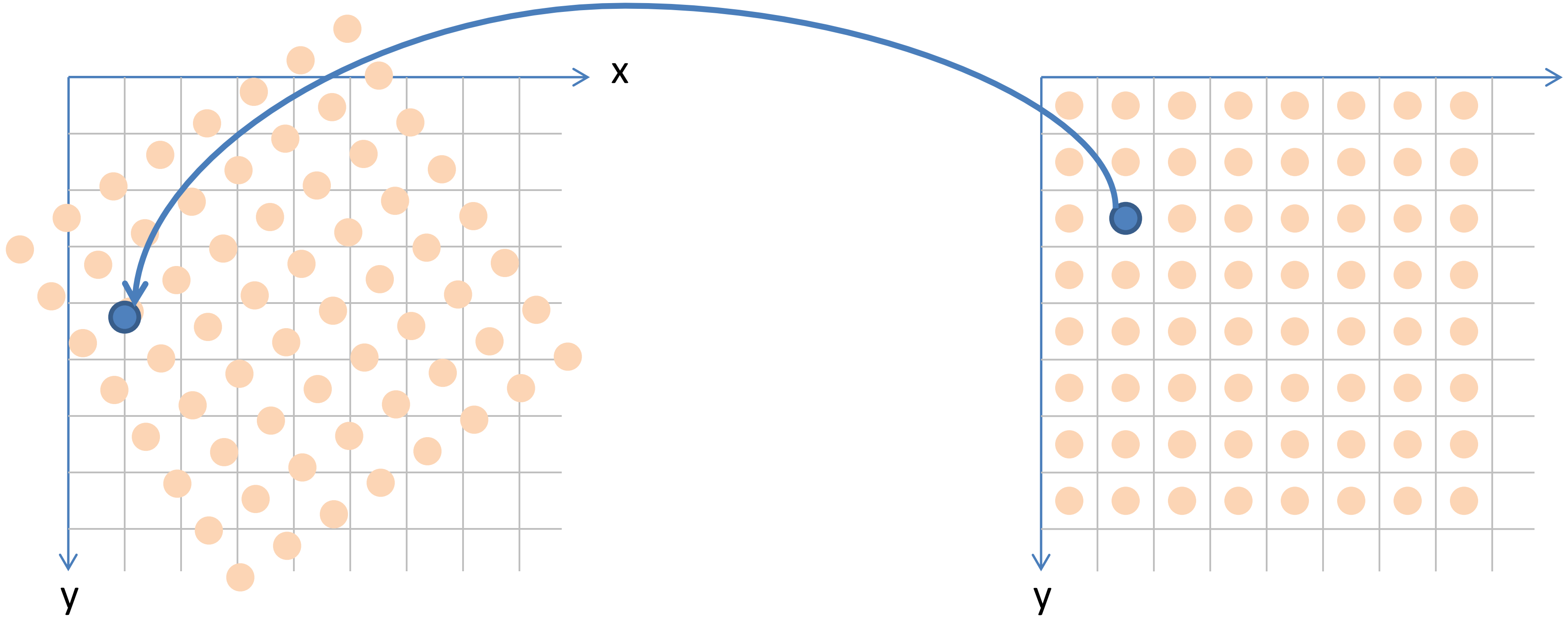
Suppose you have an image where each dot is the location of a sample on the raster image represented by an x-y plane.

Someone has rotated it 35 degrees.

What's RASTER and what's VECTOR?

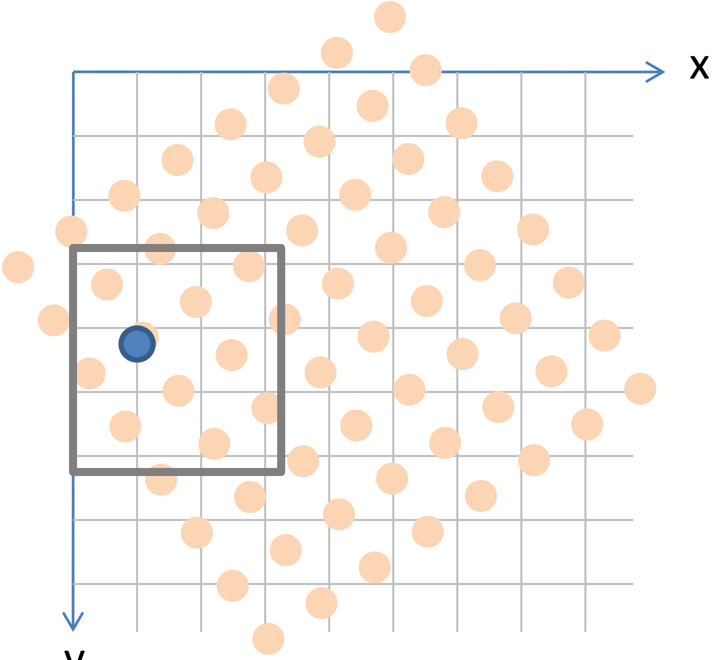


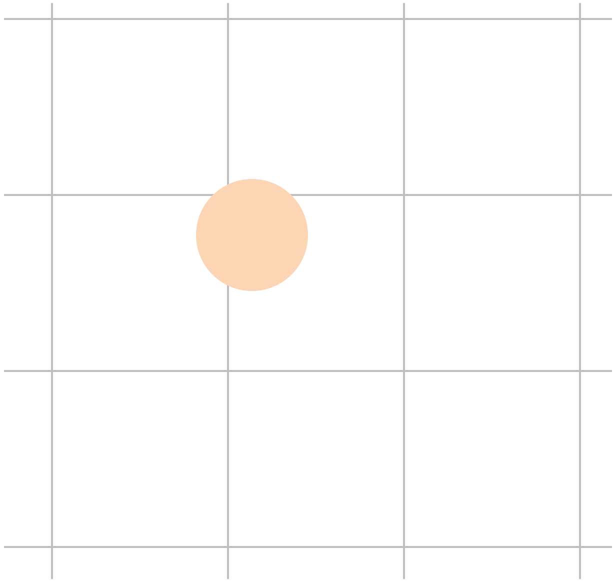
What’s RASTER and what’s VECTOR?



A 1-to-1 relationship remains for
color samples from source to destination

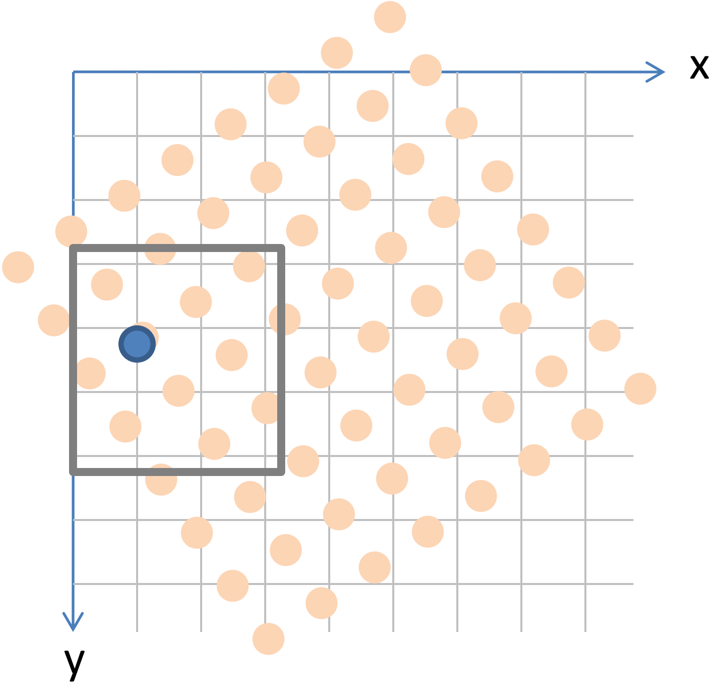
What's RASTER and what's VECTOR?

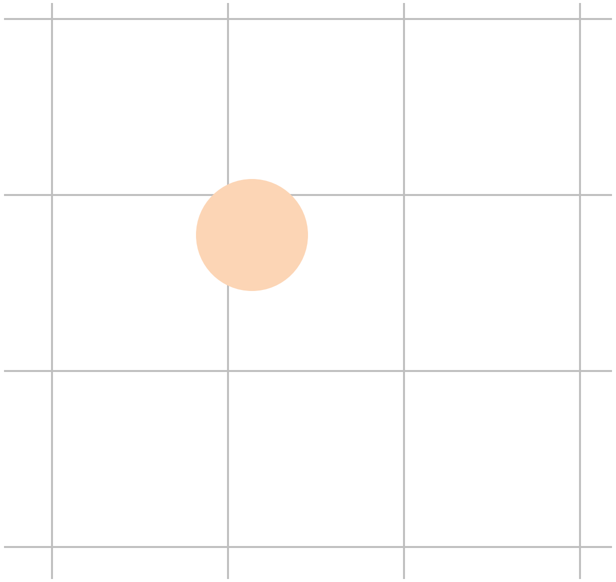




Samples now fall in floating point, not integer space.
Raster images, however, are discrete, not continuous.

What’s RASTER and what’s VECTOR?

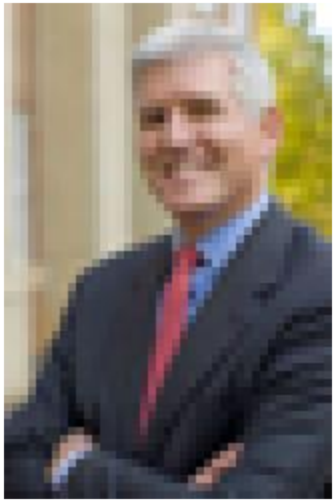




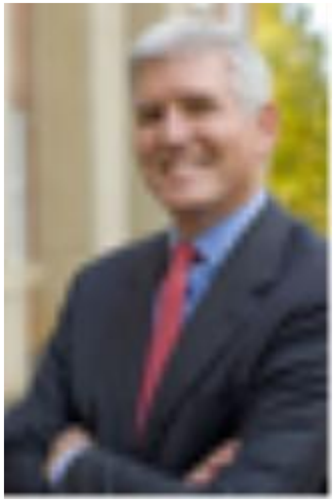
Samples now fall in floating point, not integer space.
Raster images, however, are discrete, not continuous.
This then leads to **sampling**.

What's RASTER and what's VECTOR?

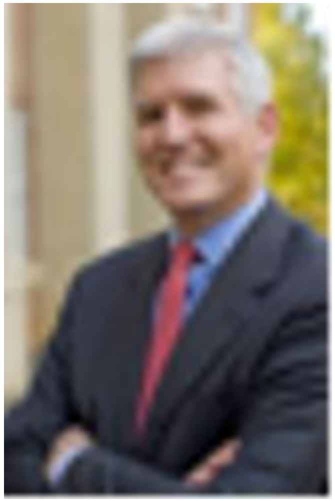
Dean Groves



Nearest Neighbor



Bilinear Interpolation

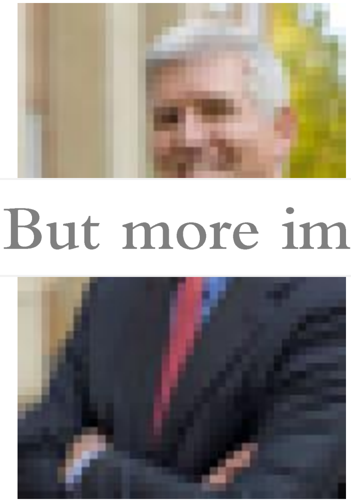


Gaussian Convolution

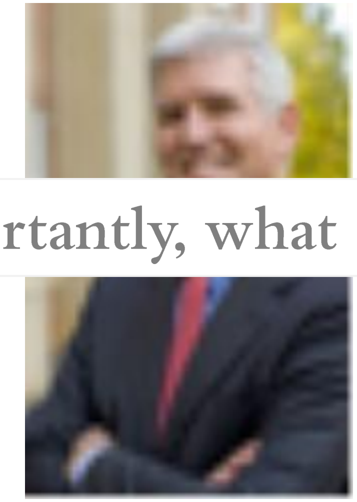
Sampling occurs in scaling and rotation in Raster images.

What's RASTER and what's VECTOR?

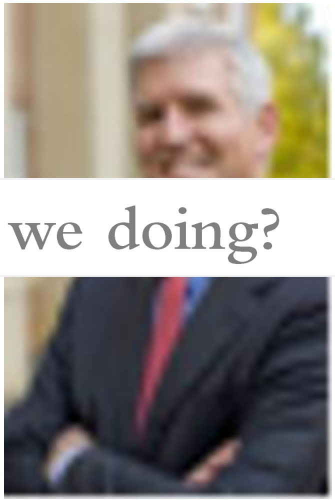
Dean Groves



Nearest Neighbor



Bilinear Interpolation



Gaussian Convolution

But more importantly, what are we doing?

Sampling occurs in scaling and rotation in Raster images.

Scalable Vector Graphics

A markup-language based vector graphics format that works in your browser.

Scalable Vector Graphics

A markup-language based vector graphics format that works in your browser.

Start with any ol’ basic HTML page

Scalable Vector Graphics

A markup-language based vector graphics format that works in your browser.

```
<!DOCTYPE html>
<html>
<body>

</body>
</html>
```

svgTest.html

Start with any ol’ basic HTML page

Scalable Vector Graphics

A markup-language based vector graphics format that works in your browser.

```
<!DOCTYPE html>
<html>
<body>
  <svg xmlns="http://www.w3.org/2000/svg" version="1.1">

  </svg>
</body>
</html>
```

svgTest.html

Add a block for SVG interpretation

Scalable Vector Graphics

A markup-language based vector graphics format that works in your browser.

```
<!DOCTYPE html>
<html>
<body>
  <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
    <rect width="300" height="100" style="fill:rgb(0,0,255);" />
  </svg>
</body>
</html>
```

svgTest.html

Add a shape!

```
<!DOCTYPE html>
<html>
<body>
  <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
    <rect width="300" height="100" style="fill:rgb(0,0,255);" />
  </svg>
</body>
</html>
```

svgTest.html

WOW.

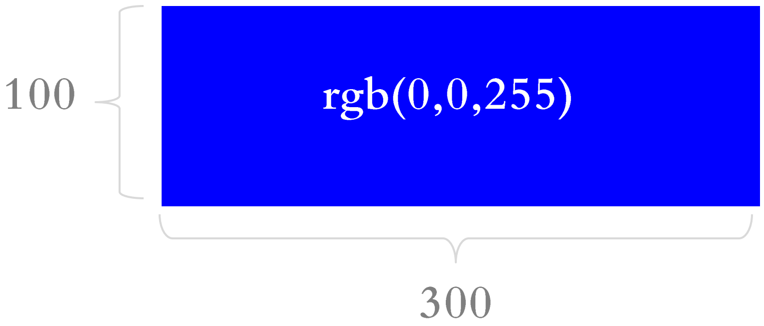
Look at this thing you made!

Wasn't that so **EASY?**

WOW.

Look at this thing you made!

Wasn't that so **EASY?**



rgb(0,0,255)

100

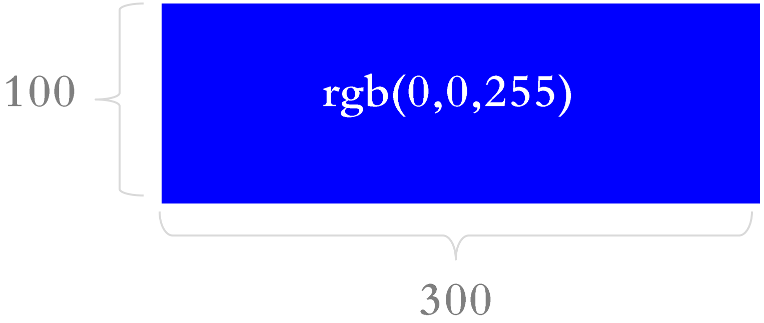
300

```
<rect width="300" height="100" style="fill:rgb(0,0,255);" />
```

WOW.

Look at this thing you made!

Wasn't that so **EASY?**



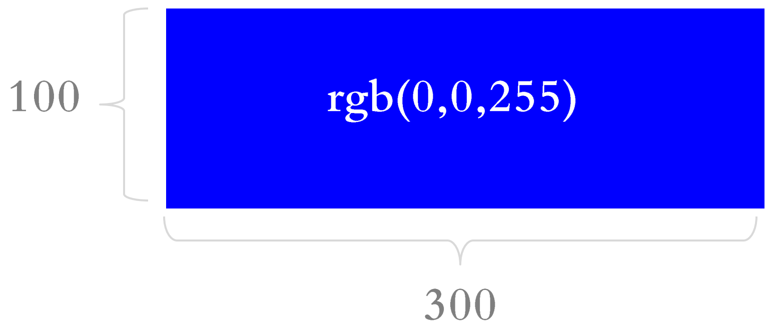
```
<rect width="300" height="100" style="fill:rgb(0,0,255);" />
```

shape attributes style

WOW.

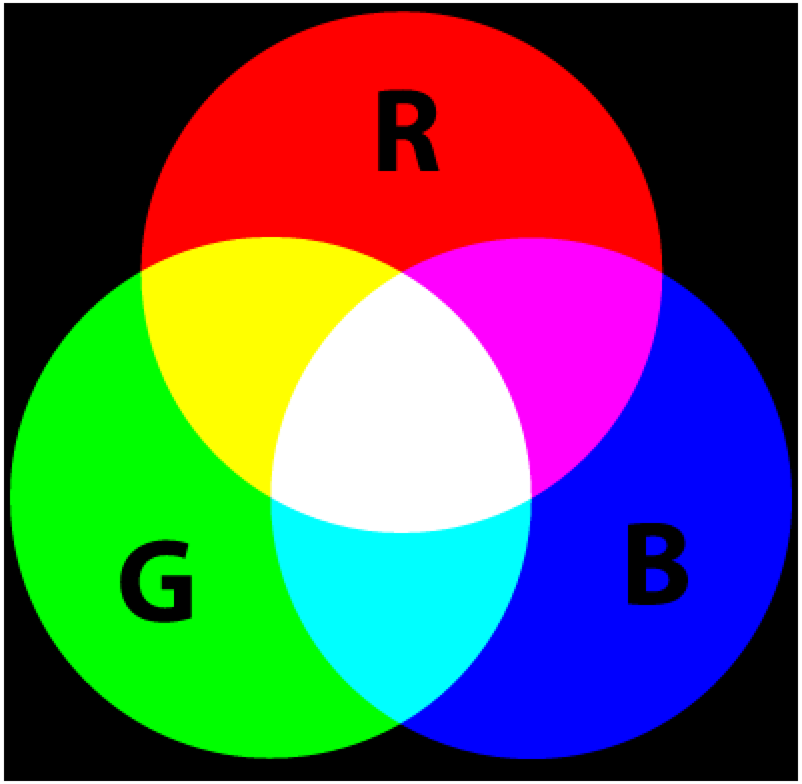
Look at this thing you made!

Wasn't that so **EASY?**



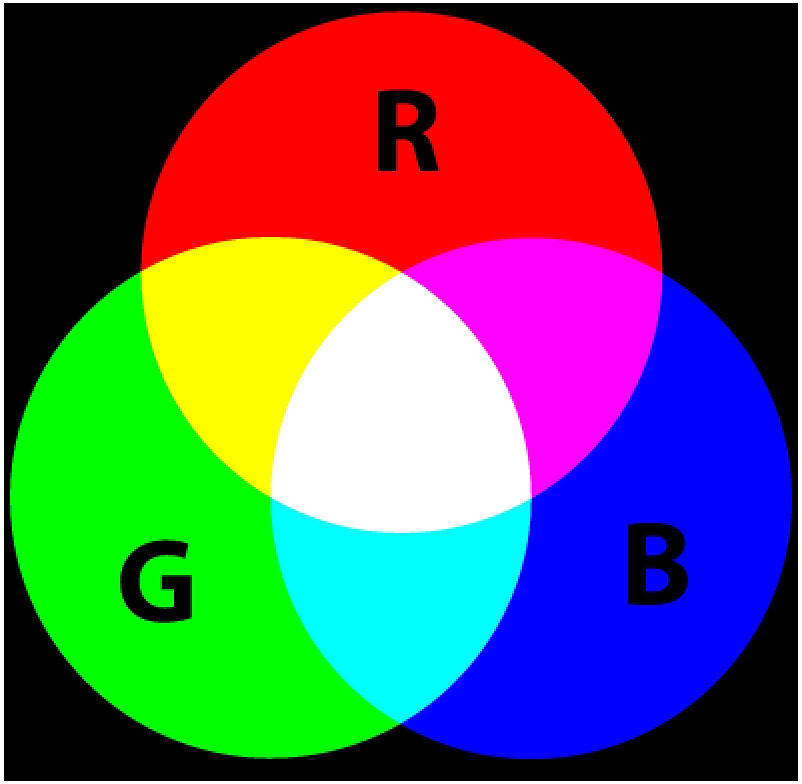
```
<rect width="300" height="100" style="fill:rgb(0,0,255);" />
```

What is this RGB thing?



RGB

An additive color model in which red, green, and blue as three separate channels are used to make up all colors.



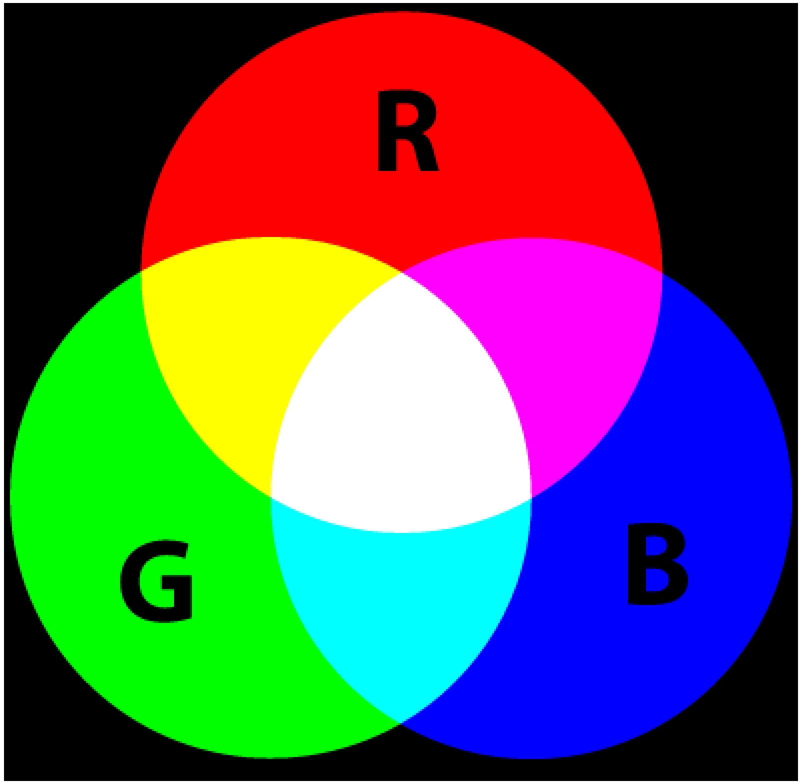
RGB

An additive color model in which red, green, and blue as three separate channels are used to make up all colors.

Each channel has 256 possible values

0
(0x00)

255
(0xFF)



RGB

An additive color model in which red, green, and blue as three separate channels are used to make up all colors.

Repeating for Red, Green, and Blue…

0 255

0 255

0 255

… gives us 16,777,216 possible colors

What color is (0, 0, 0)?

RGB

An additive color model in which red, green, and blue as three separate channels are used to make up all colors.

Repeating for Red, Green, and Blue…

0
255

0
255

0
255

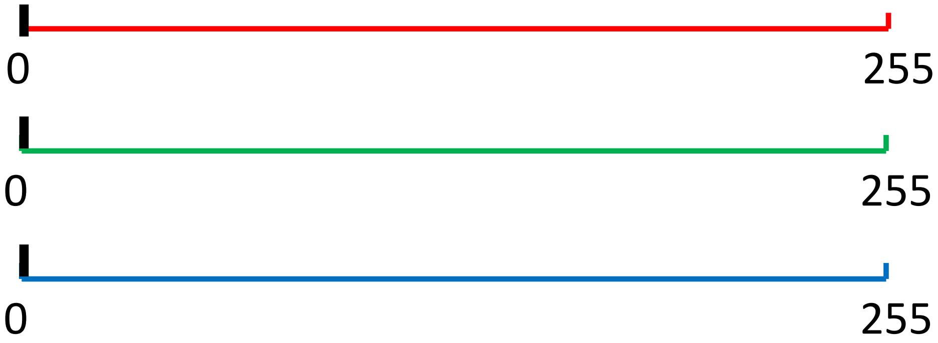
… gives us 16,777,216 possible colors

What color is (0, 0, 0)? ■

RGB

An additive color model in which red, green, and blue as three separate channels are used to make up all colors.

Repeating for Red, Green, and Blue…



… gives us 16,777,216 possible colors

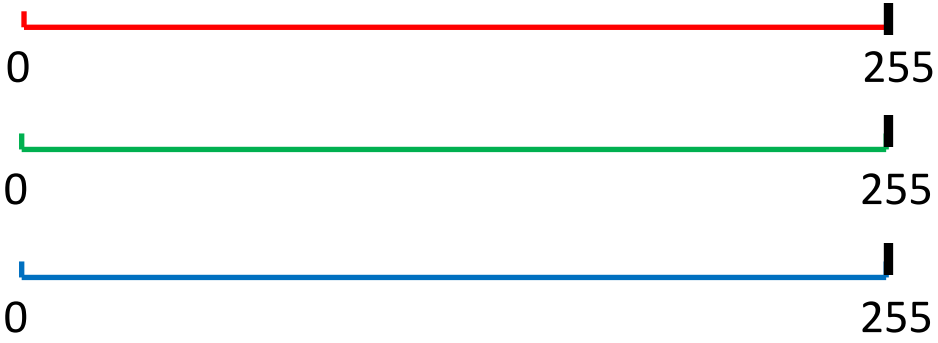
What color is (0, 0, 0)? ■

What color is (255, 255, 255)?

RGB

An additive color model in which red, green, and blue as three separate channels are used to make up all colors.

Repeating for Red, Green, and Blue…



… gives us 16,777,216 possible colors

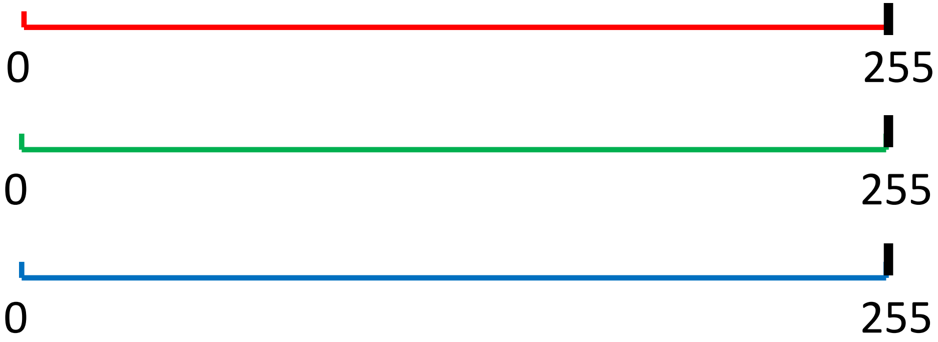
What color is (0, 0, 0)? ■

What color is (255, 255, 255)? □

RGB

An additive color model in which red, green, and blue as three separate channels are used to make up all colors.

Repeating for Red, Green, and Blue…



… gives us 16,777,216 possible colors

What color is (0, 0, 0)? ■

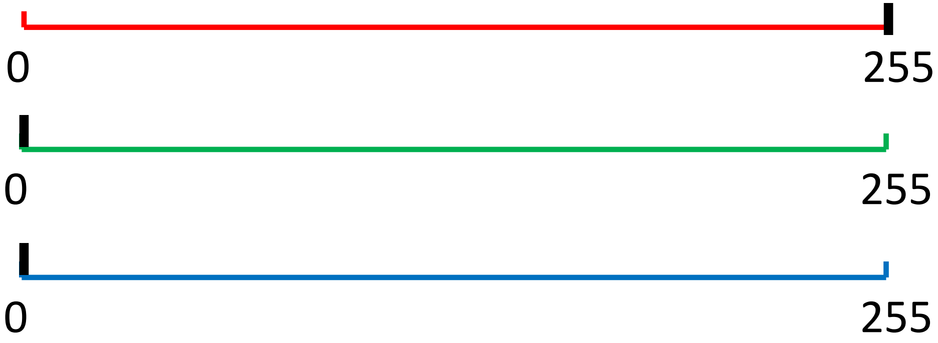
What color is (255, 255, 255)? □

What color is (255, 0, 0)?

RGB

An additive color model in which red, green, and blue as three separate channels are used to make up all colors.

Repeating for Red, Green, and Blue…



… gives us 16,777,216 possible colors

What color is (0, 0, 0)? ⬛

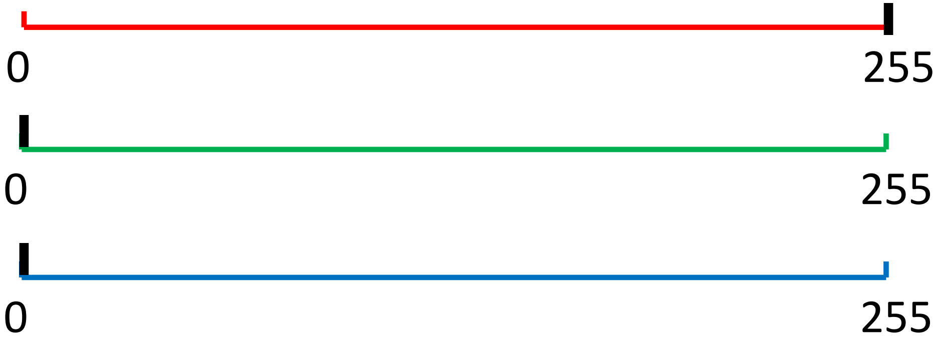
What color is (255, 255, 255)? ⬜

What color is (255, 0, 0)? 🟥

RGB

An additive color model in which red, green, and blue as three separate channels are used to make up all colors.

Repeating for Red, Green, and Blue…



… gives us 16,777,216 possible colors

What color is (0, 0, 0)? ■

What color is (255, 255, 255)? □

What color is (255, 0, 0)? 🟥

What color is (0, 255, 255)?

What color is (127, 0, 255)?

What color is (127, 127, 127?)

RGB

An additive color model in which red, green, and blue as three separate channels are used to make up all colors.

Repeating for Red, Green, and Blue…

0 255

0 255

0 255

… gives us 16,777,216 possible colors

What color is (0, 0, 0)? ⬛

What color is (255, 255, 255)? ⬜

What color is (255, 0, 0)? 🟥

What color is (0, 255, 255)? 🟦

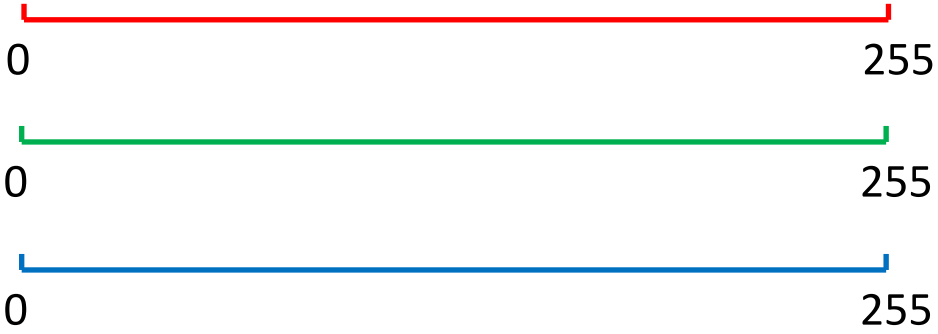
What color is (127, 0, 255)? 🟪

What color is (127, 127, 127?) ▩

RGB

An additive color model in which red, green, and blue as three separate channels are used to make up all colors.

Repeating for Red, Green, and Blue…



… gives us 16,777,216 possible colors

SVG stuff you’ll probably end up wanting at some point

SVG stuff you’ll probably end up wanting at some point

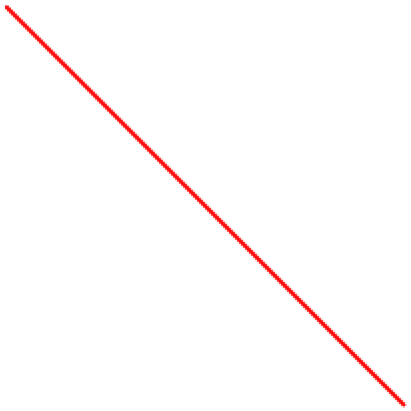
line

```
<line x1="0" y1="0" x2="200" y2="200"
style="stroke:rgb(255,0,0);stroke-width:2;" />
```

SVG stuff you'll probably end up wanting at some point

line

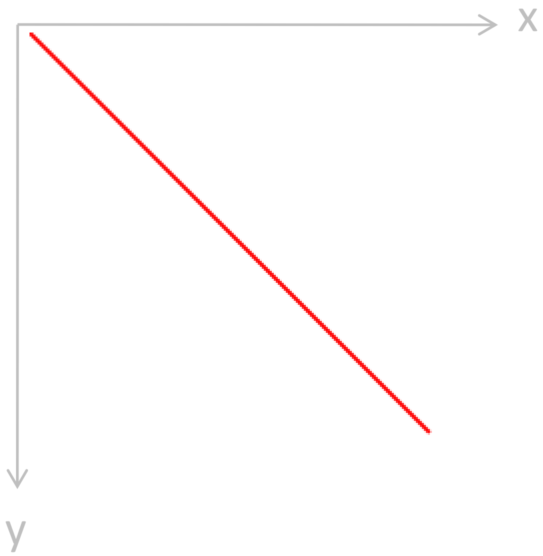
```
<line x1="0" y1="0" x2="200" y2="200"
style="stroke:rgb(255,0,0);stroke-width:2;" />
```



SVG stuff you'll probably end up wanting at some point

line

```
<line x1="0" y1="0" x2="200" y2="200"
style="stroke:rgb(255,0,0);stroke-width:2;" />
```



(pay attention to the coordinate system!)

SVG stuff you’ll probably end up wanting at some point

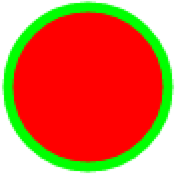
circle

```
<circle cx="200" cy="50" r="40" style="fill:rgb(255,0,0);
stroke:rgb(0,255, 0); stroke-width:5;"/>
```

SVG stuff you’ll probably end up wanting at some point

circle

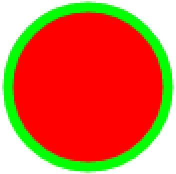
```
<circle cx="200" cy="50" r="40" style="fill:rgb(255,0,0);
stroke:rgb(0,255, 0); stroke-width:5;"/>
```



SVG stuff you'll probably end up wanting at some point

circle

```
<circle cx="200" cy="50" r="40" style="fill:rgb(255,0,0);
stroke:rgb(0,255, 0); stroke-width:5;"/>
```

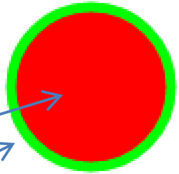


Notice the difference between "fill" and "stroke"

SVG stuff you'll probably end up wanting at some point

circle

```
<circle cx="200" cy="50" r="40" style="fill:rgb(255,0,0);
stroke:rgb(0,255, 0); stroke-width:5;"/>
```



Notice the difference between "fill" and "stroke"

SVG stuff you’ll probably end up wanting at some point

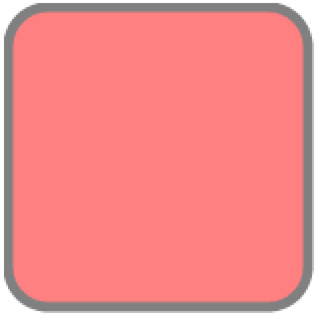
rect

```
<rect x="50" y="20" rx="20" ry="20" width="150" height="150"
style="fill:red; stroke:black; stroke-width:5; opacity:0.5“; />
```

SVG stuff you’ll probably end up wanting at some point

rect

```
<rect x="50" y="20" rx="20" ry="20" width="150" height="150"
style="fill:red; stroke:black; stroke-width:5; opacity:0.5“; />
```

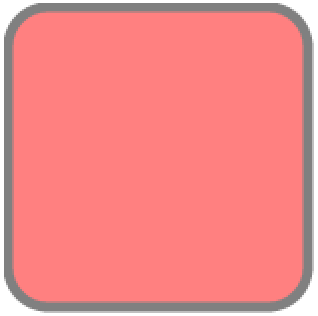


SVG stuff you'll probably end up wanting at some point

Using rx and ry rounded the rect's corners

rect

```
<rect x="50" y="20" rx="20" ry="20" width="150" height="150"
style="fill:red; stroke:black; stroke-width:5; opacity:0.5“; />
```



Opacity sets the “transparency” of the object

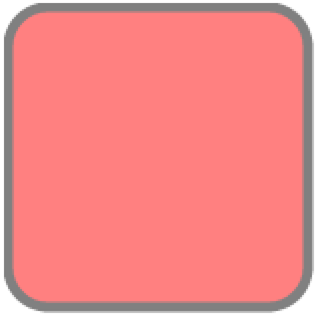
SVG stuff you'll probably end up wanting at some point

Using rx and ry rounded the rect's corners

rect

```
<rect x="50" y="20" rx="20" ry="20" width="150" height="150"
style="fill:red; stroke:black; stroke-width:5; opacity:0.5“; />
```

Opacity sets the “transparency” of the object



SVG stuff you'll probably end up wanting at some point

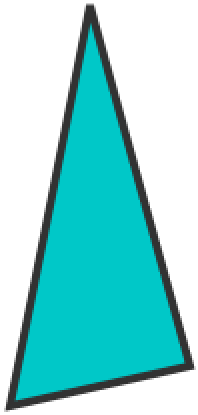
polygon

```
 <polygon points="200,10 250,190 160,210"
style="fill:rgb(0,200,200); stroke:rgb(50,50,50);
stroke-width:4;" />
```

SVG stuff you'll probably end up wanting at some point

polygon

```
<polygon points="200,10 250,190 160,210"
style="fill:rgb(0,200,200); stroke:rgb(50,50,50);
stroke-width:4;" />
```

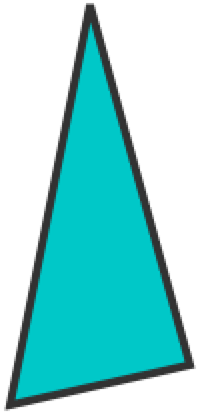


SVG stuff you'll probably end up wanting at some point

Important: There are spaces between point tuples

polygon

```
<polygon points="200,10 250,190 160,210"
style="fill:rgb(0,200,200); stroke:rgb(50,50,50);
stroke-width:4;" />
```

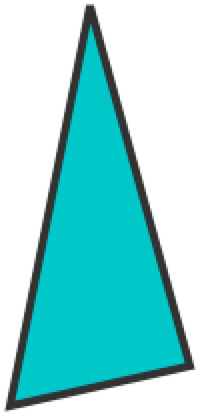


SVG stuff you'll probably end up wanting at some point

Important: There are spaces between point tuples

polygon

```
 <polygon points="200,10 250,190 160,210"
style="fill:rgb(0,200,200); stroke:rgb(50,50,50);
stroke-width:4;" />
```



SVG stuff you’ll probably end up wanting at some point

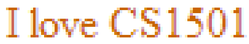
text

```
<text x="0" y="15" style="fill:rgb(200,100,0);">I love CS1501</text>
```

SVG stuff you'll probably end up wanting at some point

text

```
<text x="0" y="15" style="fill:rgb(200,100,0);">I love CS1501</text>
```

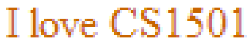


I love CS1501

SVG stuff you’ll probably end up wanting at some point

text

```
<text x="0" y="15" style="fill:rgb(200,100,0);">I love CS1501</text>
```



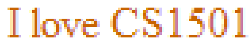
I love CS1501

Note: Your text goes between “open” and “close” tags

SVG stuff you'll probably end up wanting at some point

text

```
<text x="0" y="15" style="fill:rgb(200,100,0);">I love CS1501</text>
```



I love CS1501

Note: Your text goes between "open" and "close" tags

Surely this is all defined somewhere, right?

Surely this is all defined somewhere, right?

So much detail. Expect difficulty
http://www.w3.org/TR/2011/REC-SVG11-20110816/

Surely this is all defined somewhere, right?

So much detail. Expect difficulty
http://www.w3.org/TR/2011/REC-SVG11-20110816/

w3schools.com

Good examples, less good documentation.
http://www.w3schools.com/svg/svg_examples.asp

Surely this is all defined somewhere, right?

So much detail. Expect difficulty
http://www.w3.org/TR/2011/REC-SVG11-20110816/

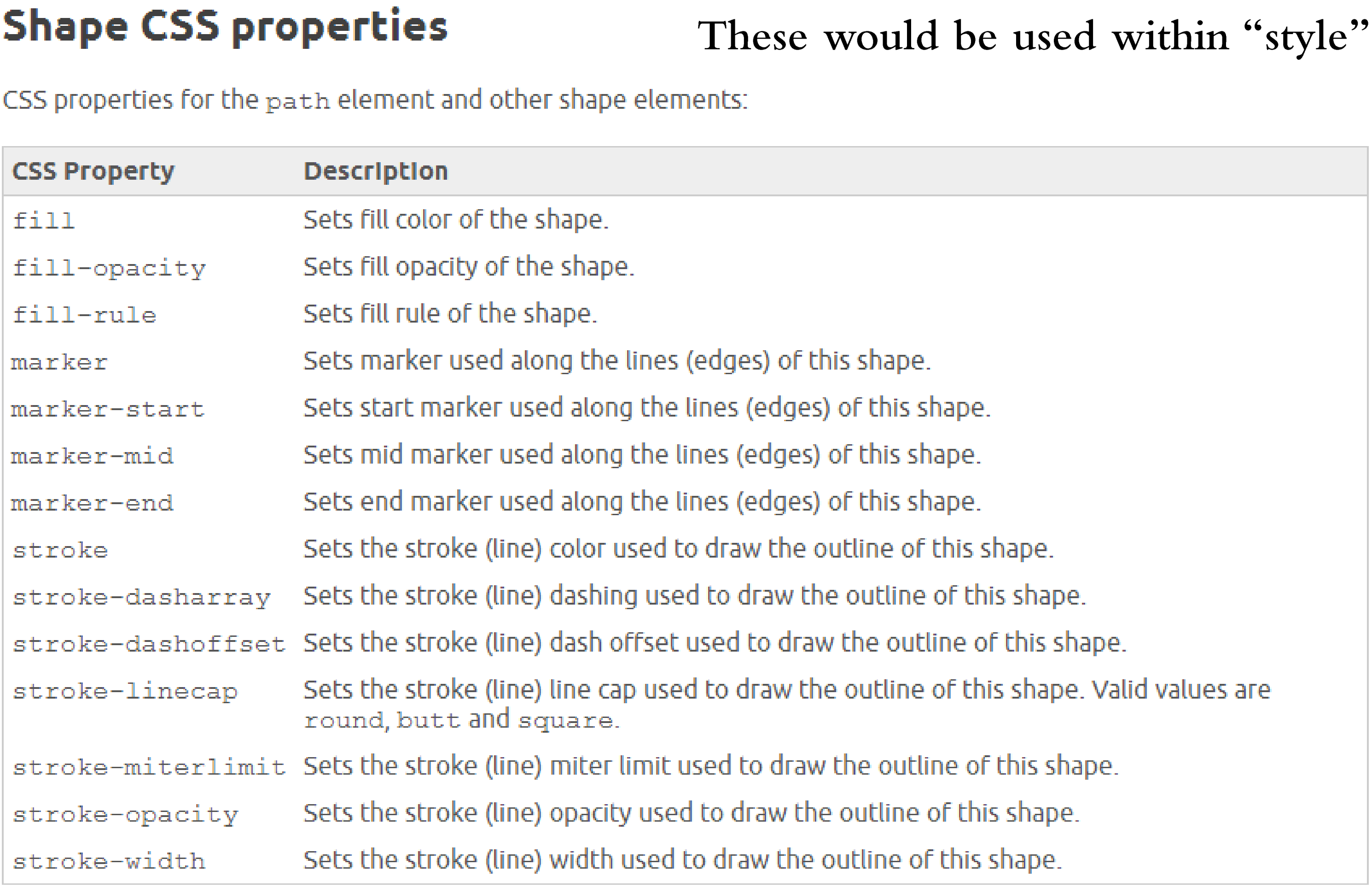
w3schools.com

Good examples, less good documentation.
http://www.w3schools.com/svg/svg_examples.asp

The next slide includes a table of properties
you might find useful for reference…

Surely this is all defined somewhere, right?

These would be used within “style”



Shape CSS properties

CSS properties for the `path` element and other shape elements:

CSS Property	Description
`fill`	Sets fill color of the shape.
`fill-opacity`	Sets fill opacity of the shape.
`fill-rule`	Sets fill rule of the shape.
`marker`	Sets marker used along the lines (edges) of this shape.
`marker-start`	Sets start marker used along the lines (edges) of this shape.
`marker-mid`	Sets mid marker used along the lines (edges) of this shape.
`marker-end`	Sets end marker used along the lines (edges) of this shape.
`stroke`	Sets the stroke (line) color used to draw the outline of this shape.
`stroke-dasharray`	Sets the stroke (line) dashing used to draw the outline of this shape.
`stroke-dashoffset`	Sets the stroke (line) dash offset used to draw the outline of this shape.
`stroke-linecap`	Sets the stroke (line) line cap used to draw the outline of this shape. Valid values are `round`, `butt` and `square`.
`stroke-miterlimit`	Sets the stroke (line) miter limit used to draw the outline of this shape.
`stroke-opacity`	Sets the stroke (line) opacity used to draw the outline of this shape.
`stroke-width`	Sets the stroke (line) width used to draw the outline of this shape.

http://tutorials.jenkov.com/svg/svg-and-css.html

Cool, but complicated.

Cool, but complicated.

Raphael

A lightweight SVG library for JavaScript

http://raphaeljs.com/

We’ll be going over this next week!

HOMEWORK

(This is on Collab)

“FHVHETGLWIZDZKRXGFIVLUZSLFHV”

Check collab. =)