

Lecture 3 : More features in Canvas 2D drawing (input elements, events/triggers, intro to transforms)

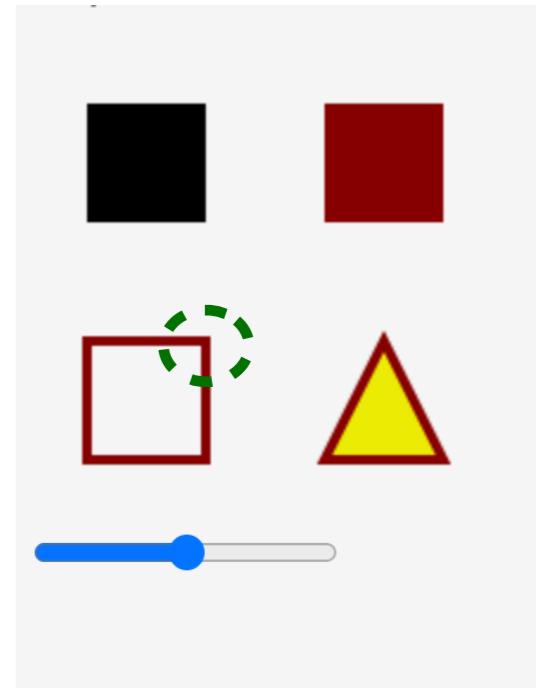
Thursday September 14th 2023

Logistics/Announcements

- We will use Canvas for posting Assignment Descriptions, rather than the Forum-like webpage that we used in past years' offering of 559 (alternatives were explored, not looking all that sustainable)
- We will still leverage some pointers to tutorials that have been put together in the past; those will be linked from your Assignments, and they do work.
- We'll continue to use Piazza as the main communication hub.

Your first programming assignment

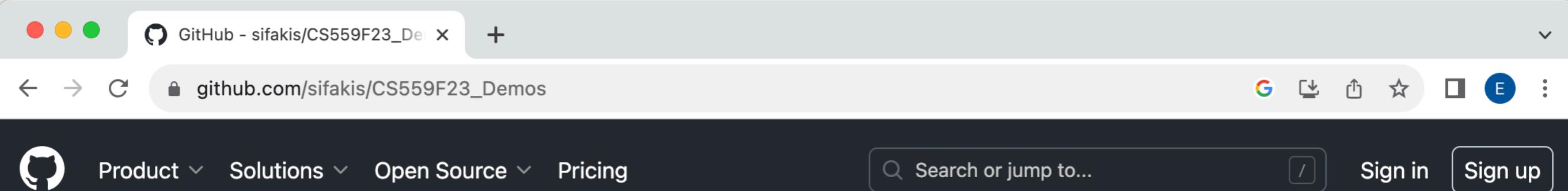
- Where do I find it?
 - Look at the Assignment Description page on Canvas which is also where you will turn in your work.
 - It has already been posted (and should be visible, tell us if not)
 - TL;DR: Create a simple 2D drawing, as an html page (not in JSbin) and include both non-filled strokes and filled shapes. Use slider(s) to change something in the drawing. If submitting more than a single .html, put in a .zip.



Maybe moving the slider changes the location of this vertex ...

Your first programming assignment

- Deadline is Thursday 9/21 @ midnight (late turn-ins till 9/24; check late policy on assignment description)
 - The description of the programming assignment includes links to reference materials, and a few brief tutorials.
Please review these materials! You will find them handy!
 - You are given a link to a public GitHub repository where the demos (shown in class via JSbin) will be duplicated, but in the form that you are expected to develop your own programs. Study those, and run them in browser.
 - Your hand-in cannot be a JSbin link. You need to provide an actual .html file (ideally with a separated-out .js file for the JavaScript code). We'll see examples of this.



Product Solutions Open Source Pricing

Search or jump to... /

Sign in

Sign up

sifakis / CS559F23_Demos

Public

Notifications

Fork 0

Star 0

Code Issues Pull requests Actions Projects Security Insights

main ▾

1 branch

0 tags

Go to file

Code ▾

About



sifakis Add files via upload

dfe467b 2 days ago 2 commits



Week2

Add files via upload

2 days ago



LICENSE

Initial commit

2 days ago

Software artifacts and Demos for CS559
(Fall 2023) "Computer Graphics"

BSD-2-Clause license

Activity

0 stars

1 watching

0 forks

Report repository

Releases

No releases published

Packages

GitHub - sifakis/CS559F23_De X +

github.com/sifakis/CS559F23_Demos

Product Solutions Open Source Pricing

Search or jump to... / Sign in Sign up

sifakis / CS559F23_Demos Public

This is a public repository (you can download/clone it, but not write to it; anyone can, in fact)

Code Issues Pull requests Actions Projects Security Insights

main 1 branch 0 tags Go to file Code About

sifakis Add files via upload dfe467b 2 days ago 2 commits

Week2 Add files via upload 2 days ago

LICENSE Initial commit 2 days ago

Software artifacts and Demos for CS559 (Fall 2023) "Computer Graphics"

BSD-2-Clause license

Activity

0 stars

1 watching

0 forks

Report repository

Releases

No releases published

Packages

A large orange arrow points to the "Open Source" menu item in the top navigation bar.

GitHub - sifakis/CS559F23_De X +

github.com/sifakis/CS559F23_Demos

You might be familiar with the Git version control system ...
We'll go through practical use "by example" in a limited capacity, but there are many good tutorials (e.g. [this](#))

Product Solutions Open

sifakis / CS559F23_Demos Public

Notifications Fork 0 Star 0

Sign up

<> Code Issues Pull requests Actions Projects Security Insights

main ▾ 1 branch 0 tags

Go to file

Code ▾

About

Software artifacts and Demos for CS559
(Fall 2023) "Computer Graphics"

BSD-2-Clause license

Activity

0 stars

1 watching

0 forks

Report repository

Releases

No releases published

Packages

GitHub - sifakis/CS559F23_Demos

github.com/sifakis/CS559F23_Demos

Product Solutions Open Source Pricing

Search or jump to... / Sign in Sign up

sifakis / CS559F23_Demos Public Notifications Fork 0 Star 0

Code Issues Pull requests Actions Projects Security Insights

main 1 branch 0 tags

Go to file Code ▾

Local Codespaces

Clone

HTTPS GitHub CLI

https://github.com/sifakis/CS559F23_Demos

Use Git or checkout with SVN using the web URL.

Open with GitHub Desktop

Download ZIP

About

Software artifacts and Demos for CS559 (Fall 2023) "Computer Graphics"

BSD-2-Clause license

Activity

0 stars

1 watching

0 forks

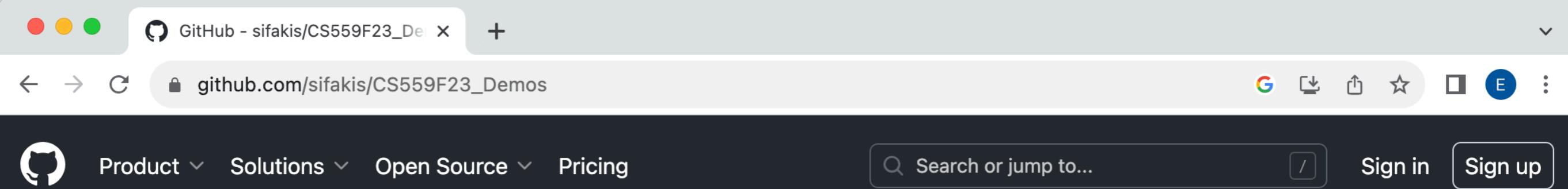
Report repository

Releases

No releases published

Packages

Quickest way to start ... download a .zip archive



sifakis / CS559F23_Demos

Public

<> Code Issues Pull requests Actions Projects Security Insights

main ▾

1 branch

0 tags

 sifakis Add files via upload

 Week2

Add files via up

 LICENSE

Initial commit

Go to file

Code ▾

Local

Codespaces

Clone

HTTPS GitHub CLI

https://github.com/sifakis/CS559F23_Demos 

Use Git or checkout with SVN using the web URL.

 Download ZIP

About

Software artifacts and Demos for CS559
(Fall 2023) "Computer Graphics"

 BSD-2-Clause license

 Activity

 0 stars

 1 watching

 0 forks

*Even better ... clone the repository to work locally
(check out the tutorial)*

Releases

No releases published

Packages



Search or jump to...

/ Pull requests Issues Marketplace Explore

sifakis / CS559F21_Demos

Public

Un

<> Code ⚡ Issues ⚡ Pull requests ⚡ Actions ⚡ Projects ⚡ Wiki ⚡ Security ⚡ Insights ⚡ Settings

main ▾

1 branch

0 tags

Go to file

Add file ▾

Code ▾

```
eftychioss-mini:Development sifakis$ git clone https://github.com/sifakis/CS559F21_Demos.git
Cloning into 'CS559F21_Demos'...
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 18 (delta 3), reused 18 (delta 3), pack-reused 0
Unpacking objects: 100% (18/18), done.
eftychioss-mini:Development sifakis$ cd CS559F21_Demos/
eftychioss-mini:CS559F21_Demos sifakis$ ls
LICENSE      README.md      Week2
eftychioss-mini:CS559F21_Demos sifakis$ cd Week2/
eftychioss-mini:Week2 sifakis$ ls
Demo0  Demo1  Demo2  Demo3
eftychioss-mini:Week2 sifakis$ cd Demo0/
eftychioss-mini:Demo0 sifakis$ ls
demo.html      demo.js
eftychioss-mini:Demo0 sifakis$
```

(Caveat: past years' links ...)

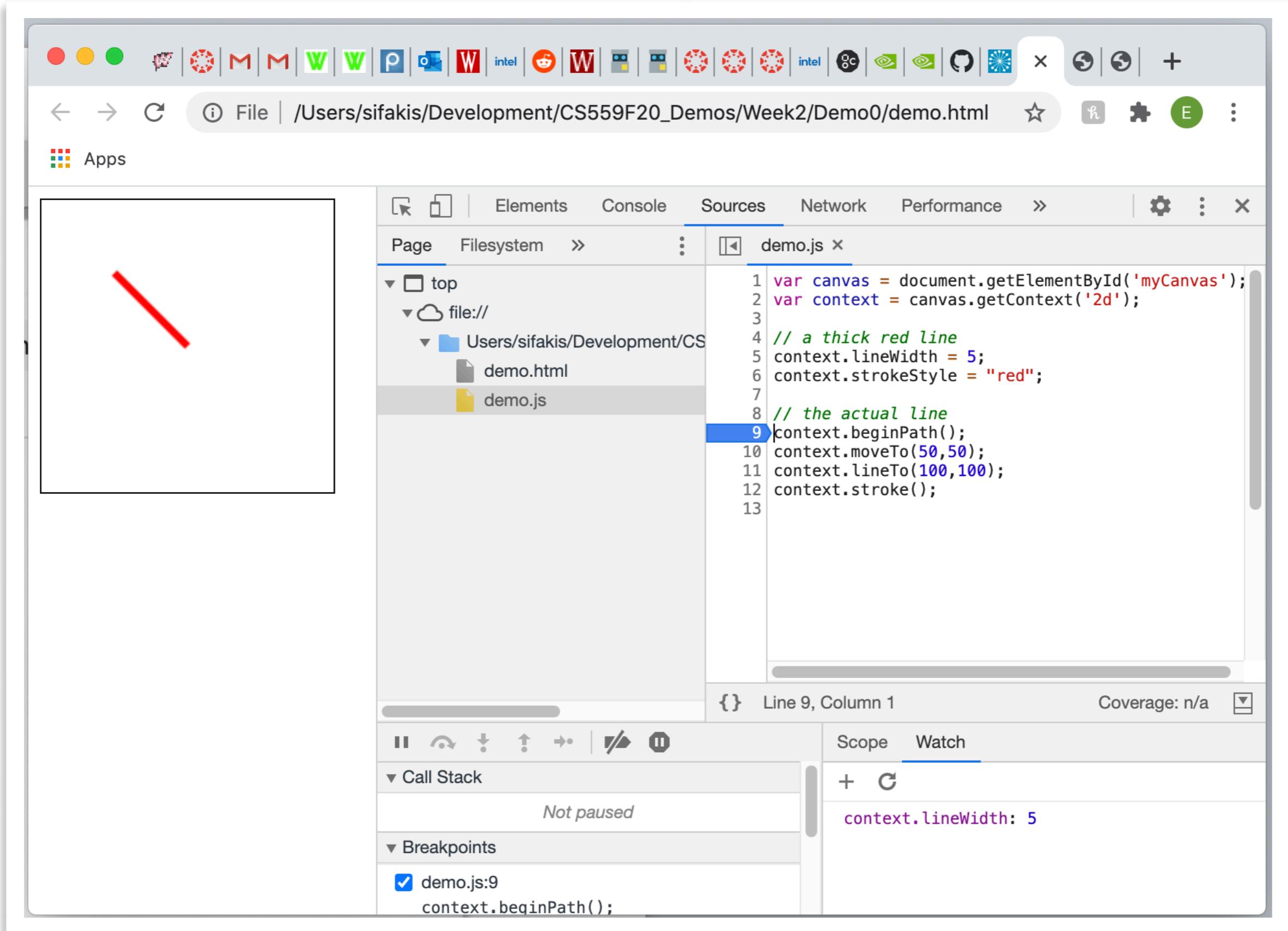
Then, simply “open” the .html file with your browser ...



Try right-click and then ...

- ... “View Page Source” to get to the HTML source (browser dependent)*
- ... click on “demo.js” to get the JavaScript source (here it’s separate)*
- ... or “Inspect” to launch the debugger (in Chrome)*

The debugger within Chrome (similar in other browsers)



The debugger within Chrome (similar in other browsers)

A screenshot of the Chrome DevTools interface, specifically the Sources tab. The tab bar at the top includes Elements, Console, Sources (which is selected and highlighted with an orange border), Network, and Performance. Below the tab bar, the left sidebar shows the file structure under 'demo.html': 'top' (empty), 'file://', and 'Users/sifakis/Development/CS' (containing 'demo.html' and 'demo.js'). The right panel displays the code for 'demo.js'. A red arrow points from the left sidebar towards the code area. The code is as follows:

```
1 var canvas = document.getElementById('myCanvas');
2 var context = canvas.getContext('2d');
3
4 // a thick red line
5 context.lineWidth = 5;
6 context.strokeStyle = "red";
7
8 // the actual line
9 context.beginPath();
10 context.moveTo(50,50);
11 context.lineTo(100,100);
12 context.stroke();
13
```

The line '9 context.beginPath();' is highlighted with a blue selection bar and a blue arrow pointing to it from the left sidebar. The status bar at the bottom indicates 'Line 9, Column 1'. In the bottom right corner of the main panel, there is a 'Coverage: n/a' section with a dropdown menu. The bottom navigation bar contains icons for play/pause, step, and breakpoints, along with sections for 'Call Stack' (showing 'Not paused') and 'Breakpoints' (listing 'demo.js:9 context.beginPath();' with a checked checkbox). On the far right, there are 'Scope' and 'Watch' tabs, with the 'Watch' tab currently active. The 'Watch' tab shows the variable 'context.lineWidth: 5'.

Many interface elements ...

What do the individual files (.html, js) look like?

demo.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>A simple drawing demo</title>
  </head>
  <body>
    <canvas id="myCanvas"
      width="200" height="200"
      style="border:1px solid #000;">
    </canvas>
    <script src="demo.js" id="module"></script>
  </body>
</html>
```

demo.js

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// a thick red line
context.lineWidth = 5;
context.strokeStyle = "red";

// the actual line
context.beginPath();
context.moveTo(50,50);
context.lineTo(100,100);
context.stroke();
```

*Difference from JSbin (or JSbin-downloaded monolithic html file) :
The JavaScript file is separate, and is specified as the source of the script*

Is this the only way?

- Running your code in your browser, from a local copy, is infinitely better than via JSbin (for reasons discussed)
- Another option: Host the page on a web-server
 - This allows much broader functionality, as browsers are peculiar with how they allow manipulation of files if not run over a web server
 - Instead of hosting the page on the web (your homepage?) you can run it via a local server
 - We *may* talk about this option in greater detail, later (if the nature of assignments strongly suggests it)

Additional features (JavaScript/Canvas)

jsbin.com/suhujar

Week2/Demo1

demo.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Coordinate axes and a filled shape</title>
  </head>
  <body>
    <canvas id="myCanvas"
      width="400" height="400">
    </canvas>
    <script src="demo.js" id="module"></script>
  </body>
</html>
```



demo.js

```
var canvas = document.getElementById('myCanvas');

function draw() {
  var context = canvas.getContext('2d');

  function DrawLshape(color) {
    context.beginPath();
    context.fillStyle = color;
    context.moveTo(50,25);
    context.lineTo(150,25);
    context.lineTo(150,75);
    context.lineTo(100,75);
    context.lineTo(100,175);
    context.lineTo(50,175);
    context.closePath();
    context.fill();
  }

  function DrawAxes(color) {
    context.strokeStyle=color;
    context.beginPath();
    // Axes
    context.moveTo(120,0);context.lineTo(0,0);context.lineTo(0,120);
    // Arrowheads
    context.moveTo(110,5);context.lineTo(120,0);context.lineTo(110,-5);
    context.moveTo(5,110);context.lineTo(0,120);context.lineTo(-5,110);
    // X-label
    context.moveTo(130,0);context.lineTo(140,10);
    context.moveTo(130,10);context.lineTo(140,0);
    // Y-label
    context.moveTo(0,130);context.lineTo(5,135);context.lineTo(10,130);
    context.moveTo(5,135);context.lineTo(5,142);

    context.stroke();
  }

  // make sure you understand these
  DrawAxes("green");
  DrawLshape("green");

}
draw();
```

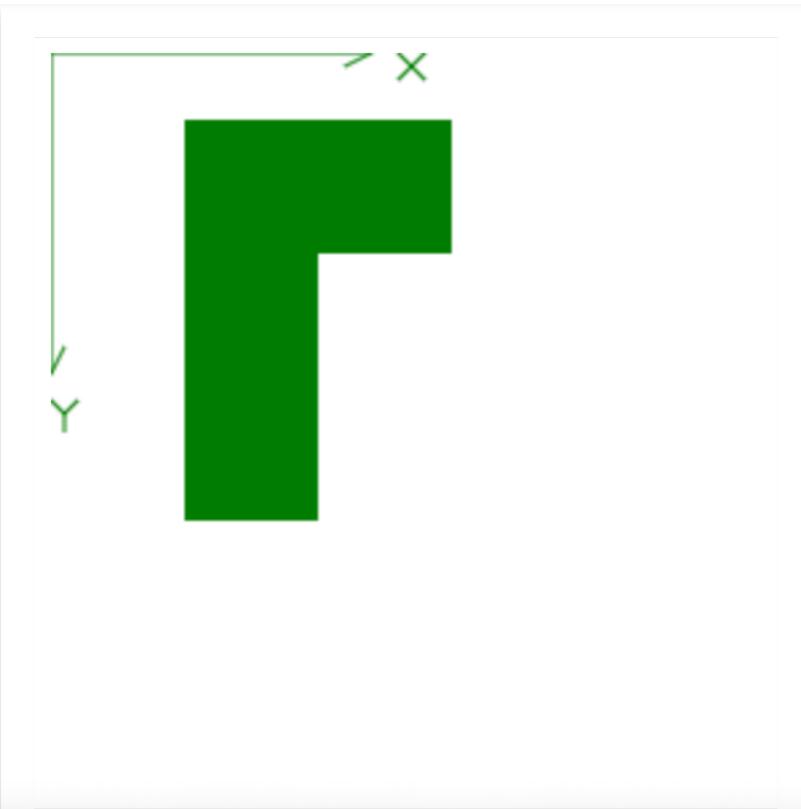
Additional features (JavaScript/Canvas)

jsbin.com/suhujar

Week2/Demo1

demo.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Coordinate axes and a filled shape</title>
  </head>
  <body>
    <canvas id="myCanvas"
      width="400" height="400">
    </canvas>
    <script src="demo.js" id="module"></script>
  </body>
</html>
```



Functions, nested functions, and calls (with parameters)

demo.js

```
var canvas = document.getElementById('myCanvas');

function draw() {
  var context = canvas.getContext('2d');

  function DrawLshape(color) {
    context.beginPath();
    context.fillStyle = color;
    context.moveTo(50,25);
    context.lineTo(150,25);
    context.lineTo(150,75);
    context.lineTo(100,75);
    context.lineTo(100,175);
    context.lineTo(50,175);
    context.closePath();
    context.fill();
  }

  function DrawAxes(color) {
    context.strokeStyle=color;
    context.beginPath();
    // Axes
    context.moveTo(120,0);context.lineTo(0,0);context.lineTo(0,120);
    // Arrowheads
    context.moveTo(110,5);context.lineTo(120,0);context.lineTo(110,-5);
    context.moveTo(5,110);context.lineTo(0,120);context.lineTo(-5,110);
    // X-label
    context.moveTo(130,0);context.lineTo(140,10);
    context.moveTo(130,10);context.lineTo(140,0);
    // Y-label
    context.moveTo(0,130);context.lineTo(5,135);context.lineTo(10,130);
    context.moveTo(5,135);context.lineTo(5,142);

    context.stroke();
  }

  // make sure you understand these
  DrawAxes("green");
  DrawLshape("green");

}

draw();
```

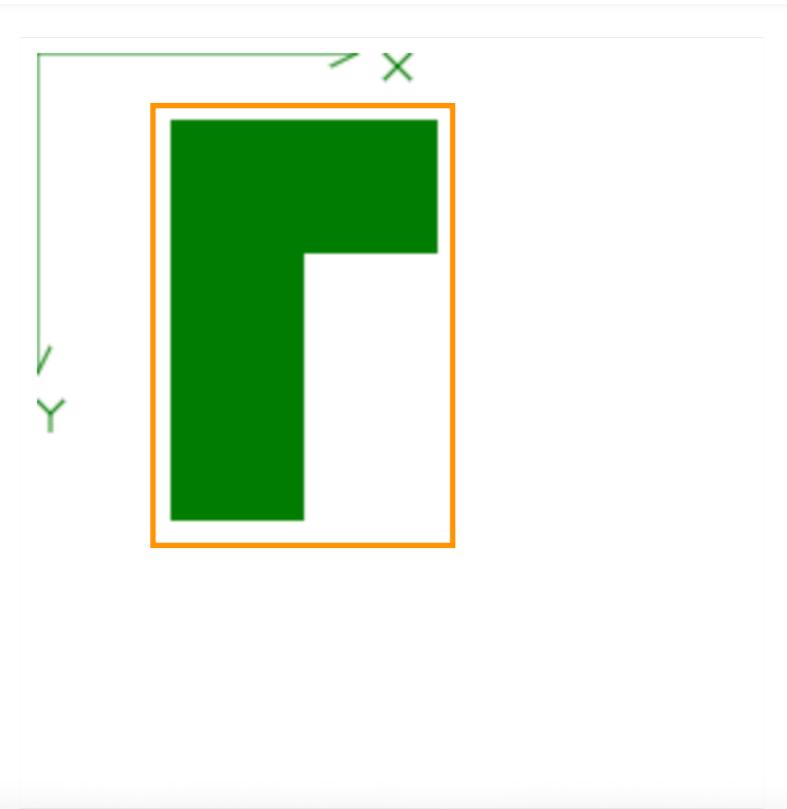
Additional features (JavaScript/Canvas)

jsbin.com/suhujar

Week2/Demo1

demo.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Coordinate axes and a filled shape</title>
  </head>
  <body>
    <canvas id="myCanvas"
      width="400" height="400">
    </canvas>
    <script src="demo.js" id="module"></script>
  </body>
</html>
```



Drawing closed, filled polygons, with specified fillStyle (vs strokeStyle)

demo.js

```
var canvas = document.getElementById('myCanvas');

function draw() {
  var context = canvas.getContext('2d');

  function DrawLshape(color) {
    context.beginPath();
    context.fillStyle = color;
    context.moveTo(50,25);
    context.lineTo(150,25);
    context.lineTo(150,75);
    context.lineTo(100,75);
    context.lineTo(100,175);
    context.lineTo(50,175);
    context.closePath();
    context.fill();
  }

  function DrawAxes(color) {
    context.strokeStyle=color;
    context.beginPath();
    // Axes
    context.moveTo(120,0);context.lineTo(0,0);context.lineTo(0,120);
    // Arrowheads
    context.moveTo(110,5);context.lineTo(120,0);context.lineTo(110,-5);
    context.moveTo(5,110);context.lineTo(0,120);context.lineTo(-5,110);
    // X-label
    context.moveTo(130,0);context.lineTo(140,10);
    context.moveTo(130,10);context.lineTo(140,0);
    // Y-label
    context.moveTo(0,130);context.lineTo(5,135);context.lineTo(10,130);
    context.moveTo(5,135);context.lineTo(5,142);

    context.stroke();
  }

  // make sure you understand these
  DrawAxes("green");
  DrawLshape("green");

} draw();
```

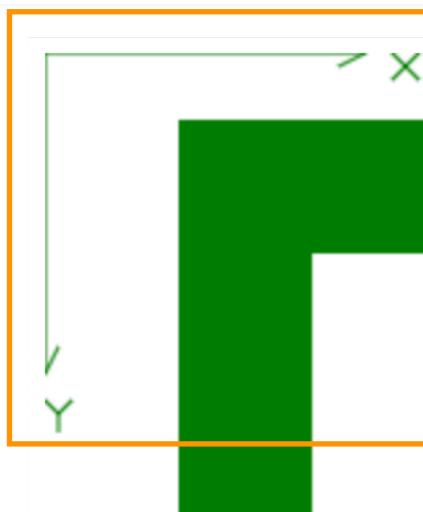
Additional features (JavaScript/Canvas)

jsbin.com/suhujar

Week2/Demo1

demo.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Coordinate axes and a filled shape</title>
  </head>
  <body>
    <canvas id="myCanvas"
      width="400" height="400">
    </canvas>
    <script src="demo.js" id="module"></script>
  </body>
</html>
```



*Drawing several (disconnected) chains of line segments
(notice moveTo/lineTo succession)*

demo.js

```
var canvas = document.getElementById('myCanvas');

function draw() {
  var context = canvas.getContext('2d');

  function DrawLshape(color) {
    context.beginPath();
    context.fillStyle = color;
    context.moveTo(50,25);
    context.lineTo(150,25);
    context.lineTo(150,75);
    context.lineTo(100,75);
    context.lineTo(100,175);
    context.lineTo(50,175);
    context.closePath();
    context.fill();
  }

  function DrawAxes(color) {
    context.strokeStyle=color;
    context.beginPath();
    // Axes
    context.moveTo(120,0);context.lineTo(0,0);context.lineTo(0,120);
    // Arrowheads
    context.moveTo(110,5);context.lineTo(120,0);context.lineTo(110,-5);
    context.moveTo(5,110);context.lineTo(0,120);context.lineTo(-5,110);
    // X-label
    context.moveTo(130,0);context.lineTo(140,10);
    context.moveTo(130,10);context.lineTo(140,0);
    // Y-label
    context.moveTo(0,130);context.lineTo(5,135);context.lineTo(10,130);
    context.moveTo(5,135);context.lineTo(5,142);

    context.stroke();
  }

  // make sure you understand these
  DrawAxes("green");
  DrawLshape("green");

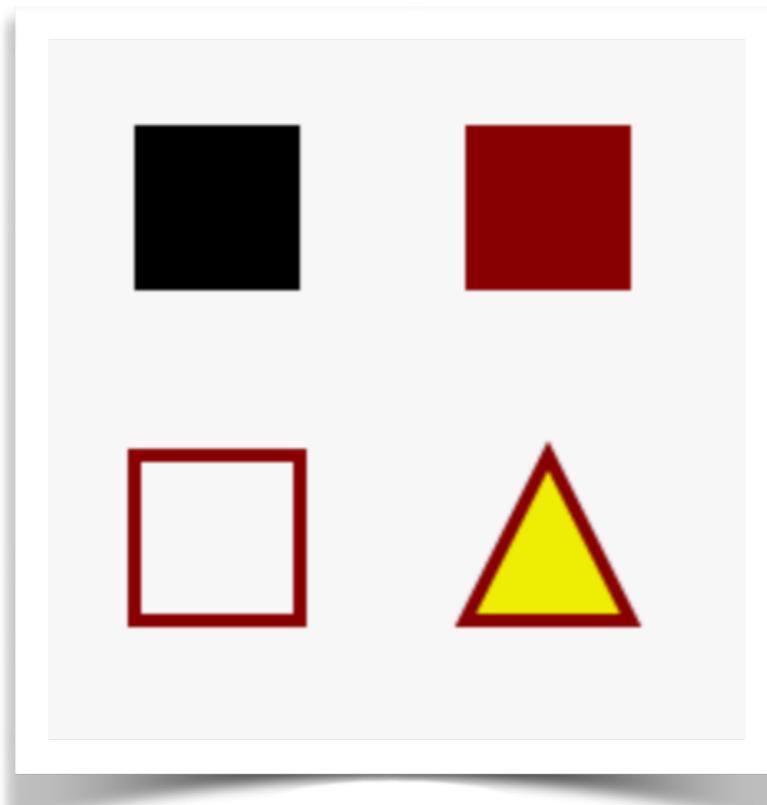
}

draw();
```

Additional features (JavaScript/Canvas)

demo.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>A few static shapes</title>
  </head>
  <body>
    <canvas id="myCanvas"
      width="200" height="200">
    </canvas>
    <script src="demo.js" id="module"></script>
  </body>
</html>
```



demo.js

```
// function draw() {
window.onload = function() {
  var canvas = document.getElementById('myCanvas');
  var context = canvas.getContext('2d');
  context.beginPath();
  context.rect(25,25,50,50);
  context.fill();

  context.beginPath();
  context.rect(125,25,50,50);
  context.fillStyle = "#800";
  context.fill();

  context.beginPath();
  context.rect(25,125,50,50);
  context.strokeStyle = "#800";
  context.lineWidth = 4;
  context.stroke();

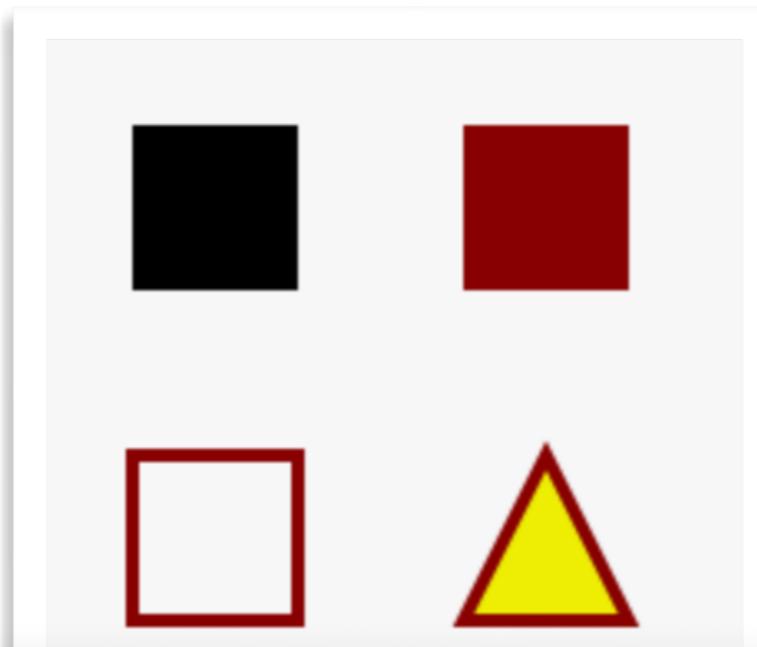
  context.beginPath();
  context.moveTo(150,125);
  context.lineTo(125,175);
  context.lineTo(175,175);
  context.closePath();
  context.fillStyle="#EE0";
  context.fill();
  context.stroke();
};

// window.onload = draw;
```

Additional features (JavaScript/Canvas)

demo.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>A few static shapes</title>
  </head>
  <body>
    <canvas id="myCanvas"
      width="200" height="200">
    </canvas>
    <script src="demo.js" id="module"></script>
  </body>
</html>
```



*Specifying which function is run
upon “loading” of the page
(check tutorial)*

Also, contrast 2 forms of syntax ...

demo.js

```
// function draw() {
window.onload = function() {
  var canvas = document.getElementById('myCanvas');
  var context = canvas.getContext('2d');
  context.beginPath();
  context.rect(25,25,50,50);
  context.fill();

  context.beginPath();
  context.rect(125,25,50,50);
  context.fillStyle = "#800";
  context.fill();

  context.beginPath();
  context.rect(25,125,50,50);
  context.strokeStyle = "#800";
  context.lineWidth = 4;
  context.stroke();

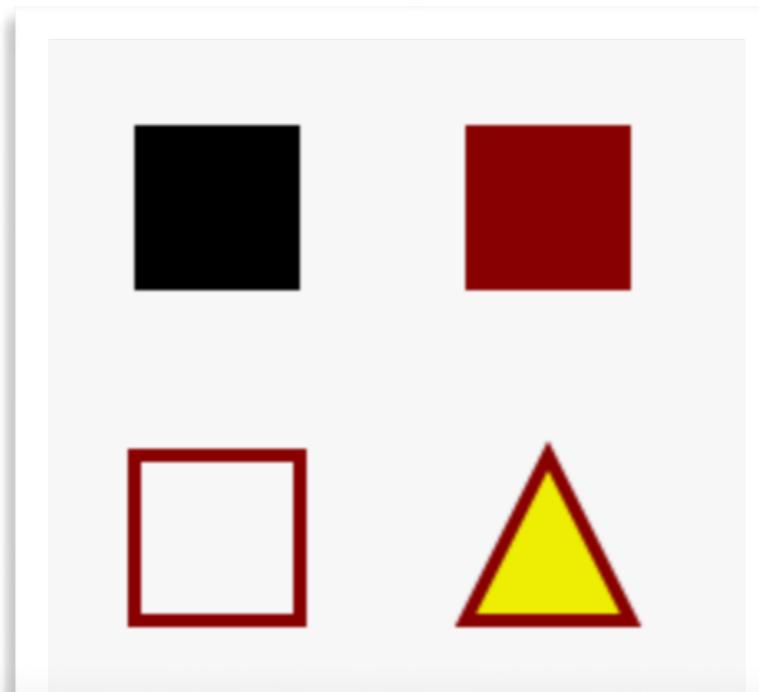
  context.beginPath();
  context.moveTo(150,125);
  context.lineTo(125,175);
  context.lineTo(175,175);
  context.closePath();
  context.fillStyle="#EE0";
  context.fill();
  context.stroke();
};

// window.onload = draw;
```

Additional features (JavaScript/Canvas)

demo.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>A few static shapes</title>
  </head>
  <body>
    <canvas id="myCanvas"
      width="200" height="200">
    </canvas>
    <script src="demo.js" id="module"></script>
  </body>
</html>
```



Drawing rectangles, and drawing polygons with different fill color and different outline color

demo.js

```
// function draw() {
window.onload = function() {
  var canvas = document.getElementById('myCanvas');
  var context = canvas.getContext('2d');
  context.beginPath();
  context.rect(25,25,50,50);
  context.fill();

  context.beginPath();
  context.rect(125,25,50,50);
  context.fillStyle = "#800";
  context.fill();

  context.beginPath();
  context.rect(25,125,50,50);
  context.strokeStyle = "#800";
  context.lineWidth = 4;
  context.stroke();

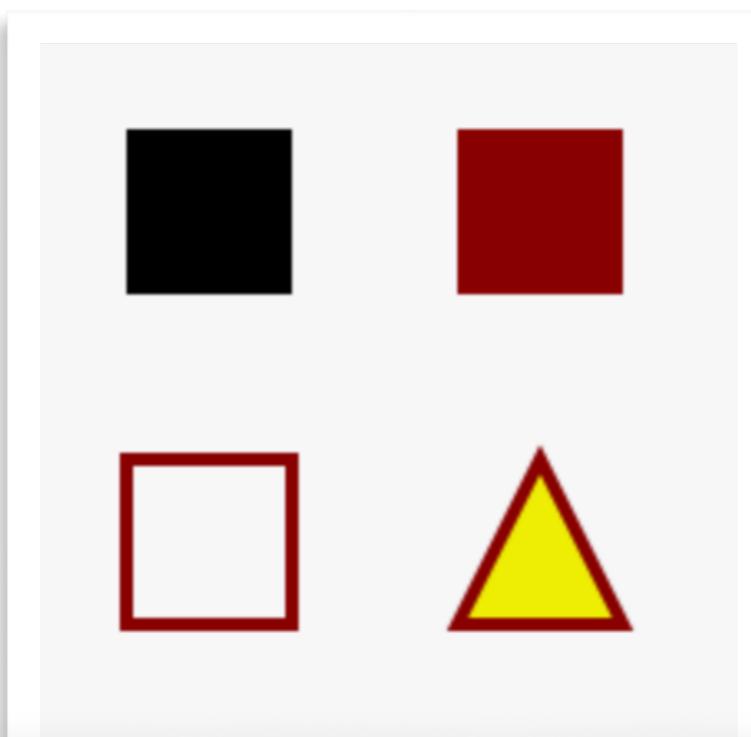
  context.beginPath();
  context.moveTo(150,125);
  context.lineTo(125,175);
  context.lineTo(175,175);
  context.closePath();
  context.fillStyle="#EE0";
  context.fill();
  context.stroke();
};

// window.onload = draw;
```

Additional features (JavaScript/Canvas)

demo.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>A few static shapes</title>
  </head>
  <body>
    <canvas id="myCanvas"
      width="200" height="200">
    </canvas>
    <script src="demo.js" id="module"></script>
  </body>
</html>
```



demo.js

```
// function draw() {
window.onload = function() {
  var canvas = document.getElementById('myCanvas');
  var context = canvas.getContext('2d');
  context.beginPath();
  context.rect(25,25,50,50);
  context.fill();

  context.beginPath();
  context.rect(125,25,50,50);
  context.fillStyle = "#800";
  context.fill();

  context.beginPath();
  context.rect(25,125,50,50);
  context.strokeStyle = "#800";
  context.lineWidth = 4;
  context.stroke();

  context.beginPath();
  context.moveTo(150,125);
  context.lineTo(125,175);
  context.lineTo(175,175);
  context.closePath();
  context.fillStyle="#EE0";
  context.fill();
  context.stroke();
};

// window.onload = draw;
```

Note alternate specification for color
(see [this tutorial](#) as well)

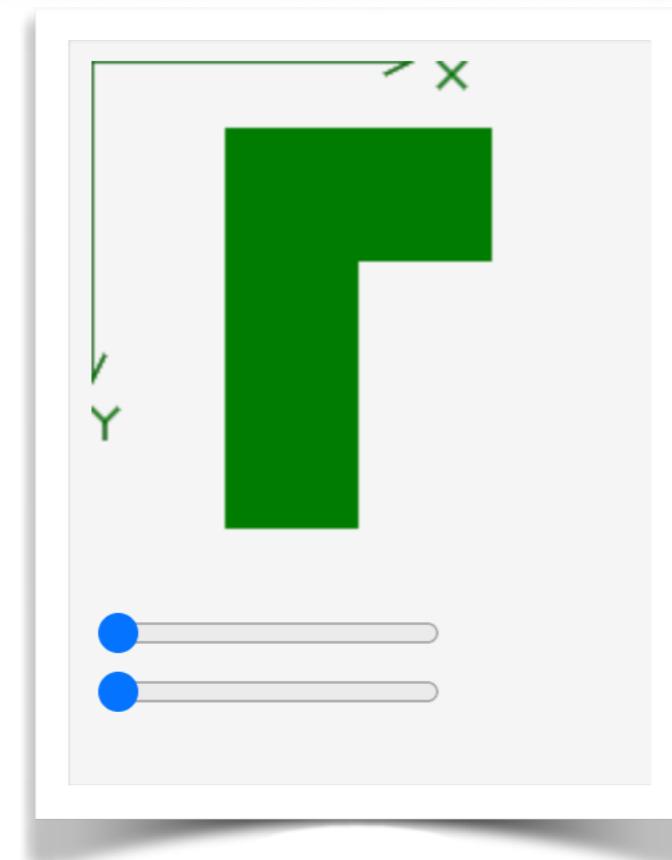
Additional features (JavaScript)

demo.js

Week2/Demo2

demo.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Simple demonstration of slider interface</title>
  </head>
  <body>
    <canvas id="myCanvas"
      width="400" height="400">
    </canvas>
    <br/>
    <input id="slider1" type="range" min="0" max="100" />
    <br/>
    <input id="slider2" type="range" min="0" max="100" />
    <script src="demo.js" id="module"></script>
  </body>
</html>
```



```
function setup() {
  var canvas = document.getElementById('myCanvas');
  var slider1 = document.getElementById('slider1');
  slider1.value = 0;
  var slider2 = document.getElementById('slider2');
  slider2.value = 0;
  function draw() {
    var context = canvas.getContext('2d');
    canvas.width = canvas.width;
    // use the sliders to get various parameters
    var dx = slider1.value;
    var dy = slider2.value;

    function DrawLshape(color) {
      context.beginPath();
      context.fillStyle = color;
      context.moveTo(50,25);context.lineTo(150,25);context.lineTo(150,75);
      context.lineTo(100,75);context.lineTo(100,175);context.lineTo(50,175);
      context.closePath();
      context.fill();
    }

    function DrawAxes(color) {
      context.strokeStyle=color;
      context.beginPath();
      // Axes
      context.moveTo(120,0);context.lineTo(0,0);context.lineTo(0,120);
      // Arrowheads
      context.moveTo(110,5);context.lineTo(120,0);context.lineTo(110,-5);
      context.moveTo(5,110);context.lineTo(0,120);context.lineTo(-5,110);
      // X-label
      context.moveTo(130,0);context.lineTo(140,10);
      context.moveTo(130,10);context.lineTo(140,0);
      // Y-label
      context.moveTo(0,130);context.lineTo(5,135);context.lineTo(10,130);
      context.moveTo(5,135);context.lineTo(5,142);

      context.stroke();
    }

    // make sure you understand these
    DrawAxes("black");
    context.save();
    context.translate(dx,dy);
    DrawAxes("green");
    DrawLshape("green");
    context.restore();
  }
  slider1.addEventListener("input",draw);
  slider2.addEventListener("input",draw);
  draw();
}
window.onload = setup;
```

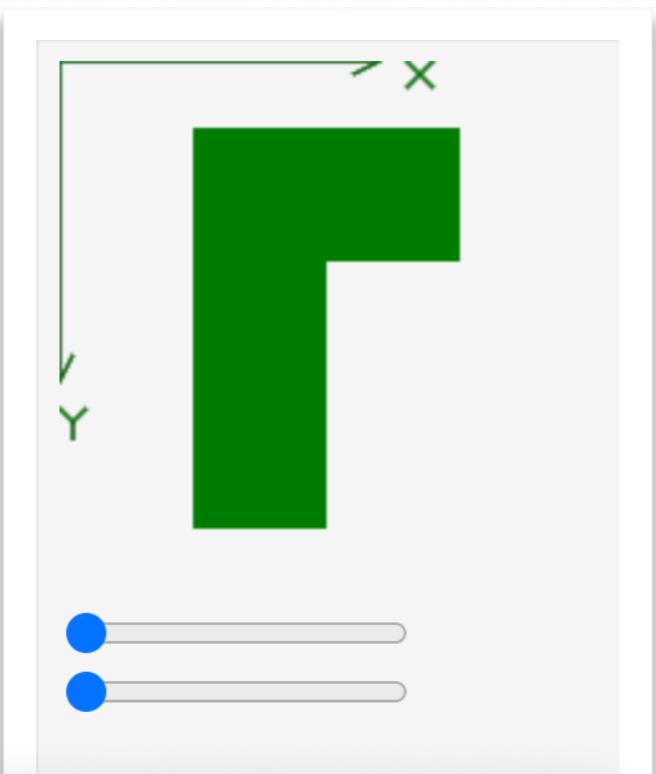
Additional features (JavaScript)

demo.js

Week2/Demo2

demo.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Simple demonstration of slider interface</title>
  </head>
  <body>
    <canvas id="myCanvas"
      width="400" height="400">
    </canvas>
    <br/>
    <input id="slider1" type="range" min="0" max="100" />
    <br/>
    <input id="slider2" type="range" min="0" max="100" />
    <script src="demo.js" id="module"></script>
  </body>
</html>
```



Sliders (declaration, initialization, retrieval of values)

```
function setup() {
  var canvas = document.getElementById('myCanvas');
  var slider1 = document.getElementById('slider1');
  slider1.value = 0;
  var slider2 = document.getElementById('slider2');
  slider2.value = 0;
  function draw() {
    var context = canvas.getContext('2d');
    canvas.width = canvas.width;
    // use the sliders to get various parameters
    var dx = slider1.value;
    var dy = slider2.value;

    function DrawLshape(color) {
      context.beginPath();
      context.fillStyle = color;
      context.moveTo(50,25);context.lineTo(150,25);context.lineTo(150,75);
      context.lineTo(100,75);context.lineTo(100,175);context.lineTo(50,175);
      context.closePath();
      context.fill();
    }

    function DrawAxes(color) {
      context.strokeStyle=color;
      context.beginPath();
      // Axes
      context.moveTo(120,0);context.lineTo(0,0);context.lineTo(0,120);
      // Arrowheads
      context.moveTo(110,5);context.lineTo(120,0);context.lineTo(110,-5);
      context.moveTo(5,110);context.lineTo(0,120);context.lineTo(-5,110);
      // X-label
      context.moveTo(130,0);context.lineTo(140,10);
      context.moveTo(130,10);context.lineTo(140,0);
      // Y-label
      context.moveTo(0,130);context.lineTo(5,135);context.lineTo(10,130);
      context.moveTo(5,135);context.lineTo(5,142);

      context.stroke();
    }

    // make sure you understand these
    DrawAxes("black");
    context.save();
    context.translate(dx,dy);
    DrawAxes("green");
    DrawLshape("green");
    context.restore();
  }
  slider1.addEventListener("input",draw);
  slider2.addEventListener("input",draw);
  draw();
}
window.onload = setup;
```

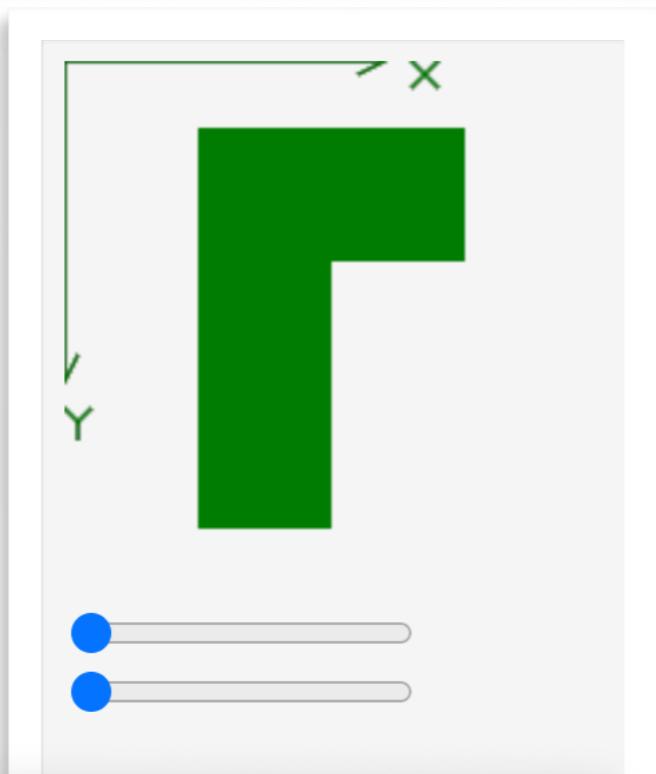
Additional features (JavaScript)

demo.js

Week2/Demo2

demo.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Simple demonstration of slider interface</title>
  </head>
  <body>
    <canvas id="myCanvas"
      width="400" height="400">
    </canvas>
    <br/>
    <input id="slider1" type="range" min="0" max="100" />
    <br/>
    <input id="slider2" type="range" min="0" max="100" />
    <script src="demo.js" id="module"></script>
  </body>
</html>
```



Using slider input as a trigger for re-drawing (again, tutorial)

```
function setup() {
  var canvas = document.getElementById('myCanvas');
  var slider1 = document.getElementById('slider1');
  slider1.value = 0;
  var slider2 = document.getElementById('slider2');
  slider2.value = 0;
  function draw() {
    var context = canvas.getContext('2d');
    canvas.width = canvas.width;
    // use the sliders to get various parameters
    var dx = slider1.value;
    var dy = slider2.value;

    function DrawLshape(color) {
      context.beginPath();
      context.fillStyle = color;
      context.moveTo(50,25);context.lineTo(150,25);context.lineTo(150,75);
      context.lineTo(100,75);context.lineTo(100,175);context.lineTo(50,175);
      context.closePath();
      context.fill();
    }

    function DrawAxes(color) {
      context.strokeStyle=color;
      context.beginPath();
      // Axes
      context.moveTo(120,0);context.lineTo(0,0);context.lineTo(0,120);
      // Arrowheads
      context.moveTo(110,5);context.lineTo(120,0);context.lineTo(110,-5);
      context.moveTo(5,110);context.lineTo(0,120);context.lineTo(-5,110);
      // X-label
      context.moveTo(130,0);context.lineTo(140,10);
      context.moveTo(130,10);context.lineTo(140,0);
      // Y-label
      context.moveTo(0,130);context.lineTo(5,135);context.lineTo(10,130);
      context.moveTo(5,135);context.lineTo(5,142);

      context.stroke();
    }

    // make sure you understand these
    DrawAxes("black");
    context.save();
    context.translate(dx,dy);
    DrawAxes("green");
    DrawLshape("green");
    context.restore();
  }
  slider1.addEventListener("input",draw);
  slider2.addEventListener("input",draw);
  draw();
}
window.onload = setup;
```

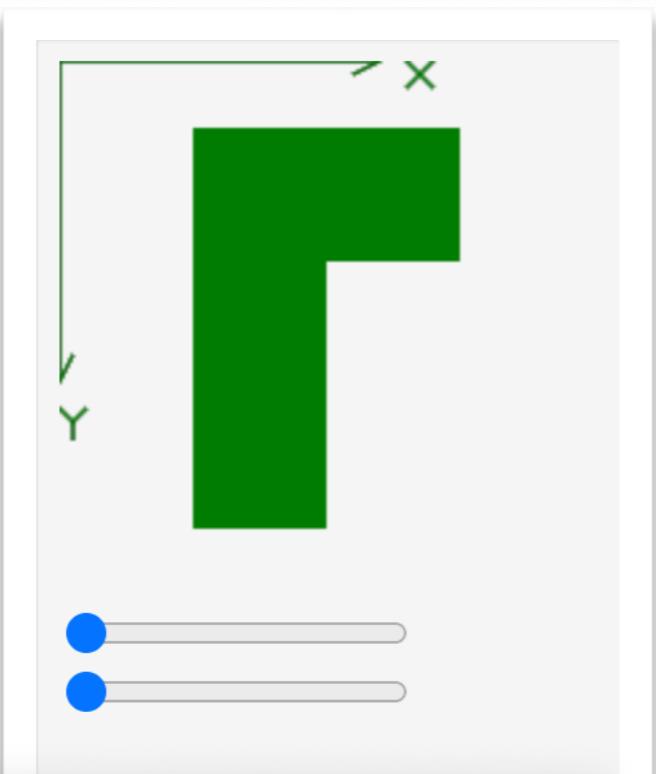
Additional features (JavaScript)

demo.js

Week2/Demo2

demo.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Simple demonstration of slider interface</title>
  </head>
  <body>
    <canvas id="myCanvas"
      width="400" height="400">
    </canvas>
    <br/>
    <input id="slider1" type="range" min="0" max="100" />
    <br/>
    <input id="slider2" type="range" min="0" max="100" />
    <script src="demo.js" id="module"></script>
  </body>
</html>
```



Transforms (we'll get into this in much more detail next week ...)

```
function setup() {
  var canvas = document.getElementById('myCanvas');
  var slider1 = document.getElementById('slider1');
  slider1.value = 0;
  var slider2 = document.getElementById('slider2');
  slider2.value = 0;
  function draw() {
    var context = canvas.getContext('2d');
    canvas.width = canvas.width;
    // use the sliders to get various parameters
    var dx = slider1.value;
    var dy = slider2.value;

    function DrawLshape(color) {
      context.beginPath();
      context.fillStyle = color;
      context.moveTo(50,25);context.lineTo(150,25);context.lineTo(150,75);
      context.lineTo(100,75);context.lineTo(100,175);context.lineTo(50,175);
      context.closePath();
      context.fill();
    }

    function DrawAxes(color) {
      context.strokeStyle=color;
      context.beginPath();
      // Axes
      context.moveTo(120,0);context.lineTo(0,0);context.lineTo(0,120);
      // Arrowheads
      context.moveTo(110,5);context.lineTo(120,0);context.lineTo(110,-5);
      context.moveTo(5,110);context.lineTo(0,120);context.lineTo(-5,110);
      // X-label
      context.moveTo(130,0);context.lineTo(140,10);
      context.moveTo(130,10);context.lineTo(140,0);
      // Y-label
      context.moveTo(0,130);context.lineTo(5,135);context.lineTo(10,130);
      context.moveTo(5,135);context.lineTo(5,142);

      context.stroke();
    }

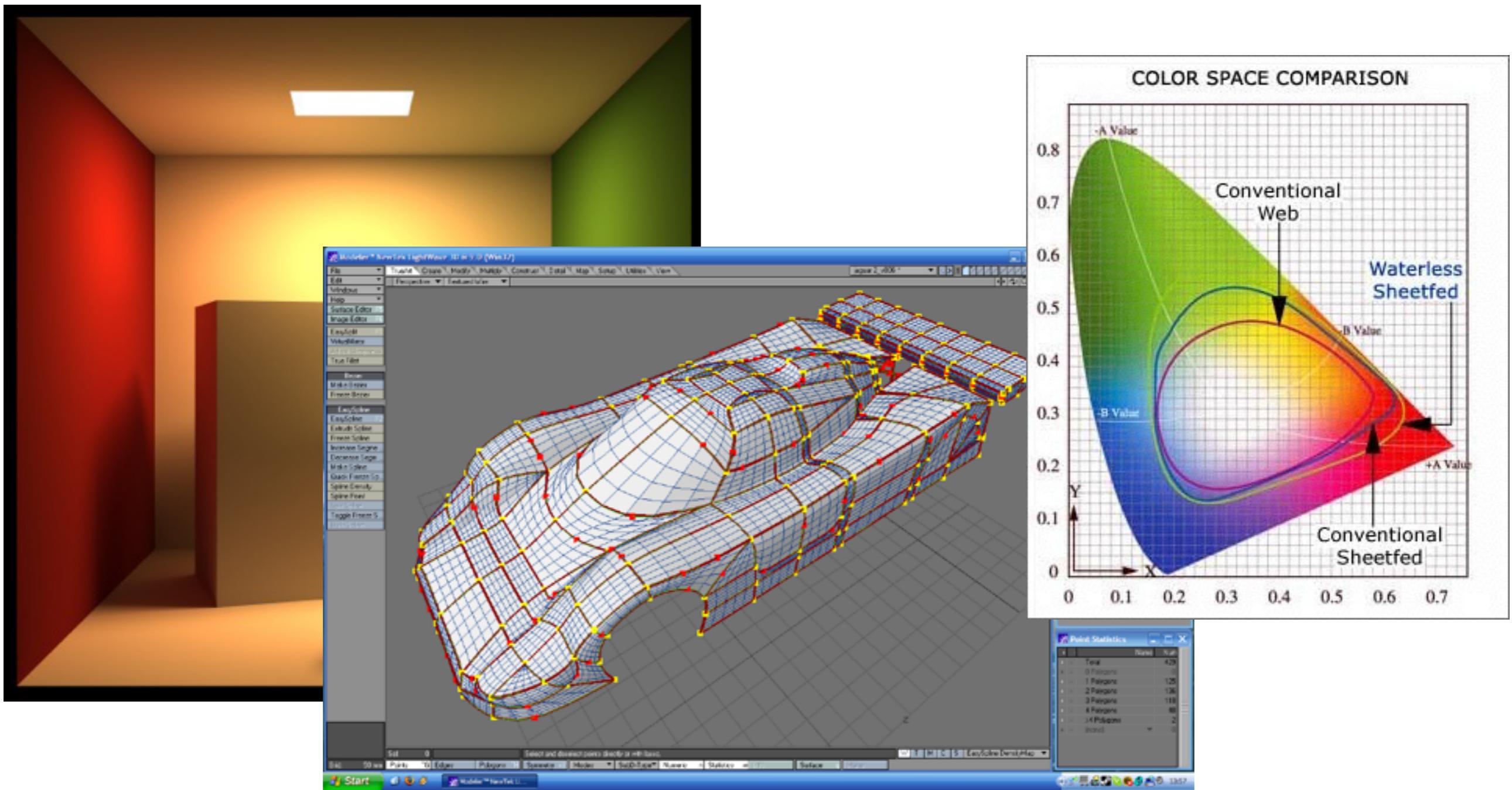
    // make sure you understand these
    DrawAxes("black");
    context.save();
    context.translate(dx,dy);
    DrawAxes("green");
    DrawLshape("green");
    context.restore();
  }
  slider1.addEventListener("input",draw);
  slider2.addEventListener("input",draw);
  draw();
}
window.onload = setup;
```

Additional examples (flash preview)

- Simple animation
 - Specify what action (drawing) is to be taken when the browser is ready to “refresh” its visual contents
 - Creates a recurrent action trigger, that we can use to create an animation

Additional examples (flash preview)

- Hierarchical modeling
 - Shapes subordinate to the placement of other shapes
 - Uses several coordinate frames for drawing
 - Nested hierarchy of transforms
 - Illustration of the Canvas transform stack
 - (all of these to be discussed next week)

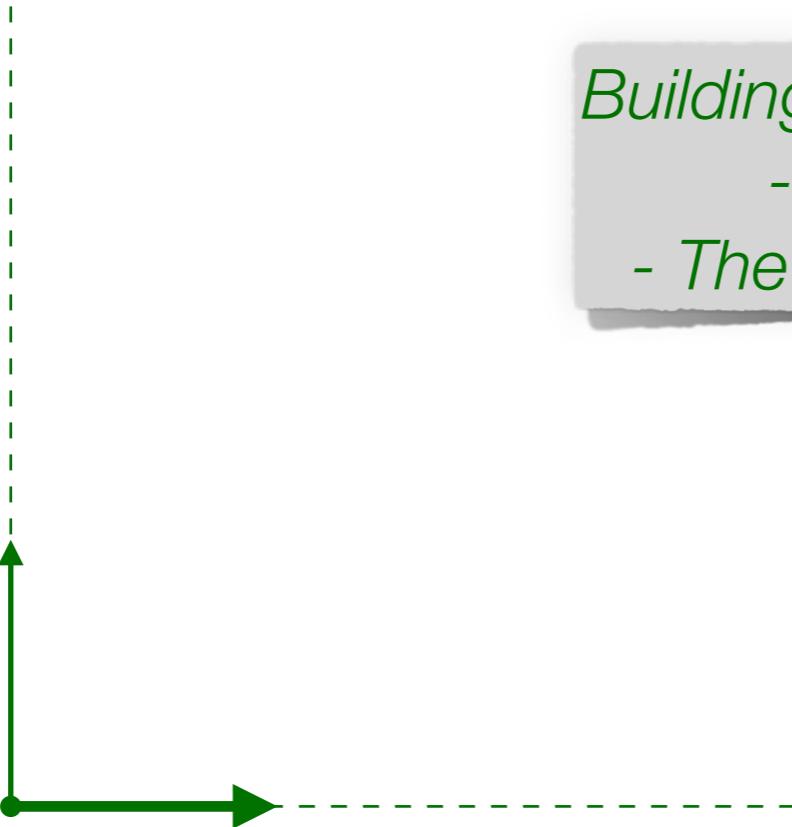


Lecture 3 : More features in Canvas 2D drawing (input elements, events/triggers, intro to transforms)

Thursday September 14th 2023

(Overflow slides;
likely we'll cover them!)

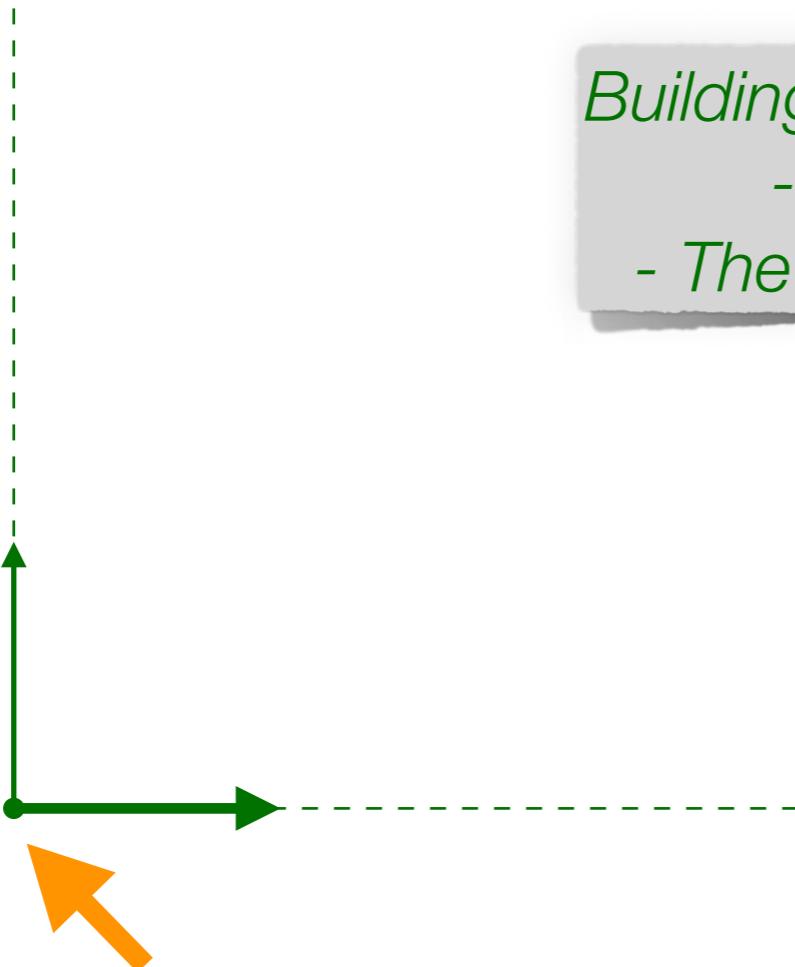
Coordinate systems and transforms



Building blocks of a coordinate system

- *The origin (“origin point”)*
- *The axis vectors (2 in 2D, 3 in 3D)*

Coordinate systems and transforms

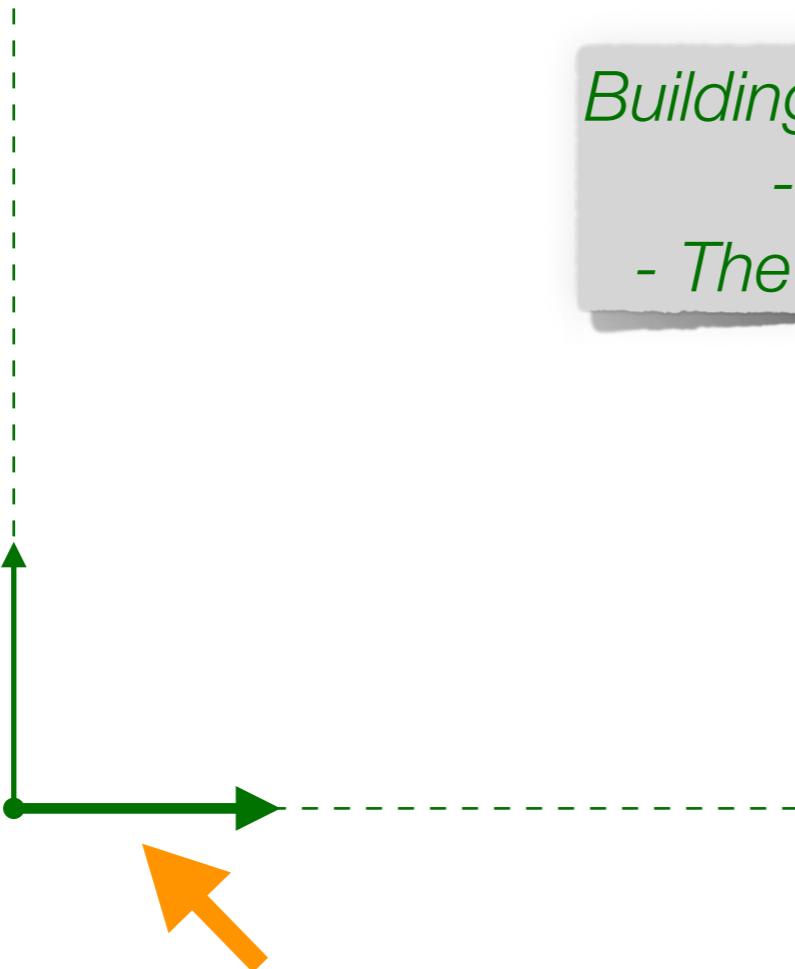


Building blocks of a coordinate system

- *The origin (“origin point”)*
- *The axis vectors (2 in 2D, 3 in 3D)*

The point we call the origin will be depicted as a circle

Coordinate systems and transforms

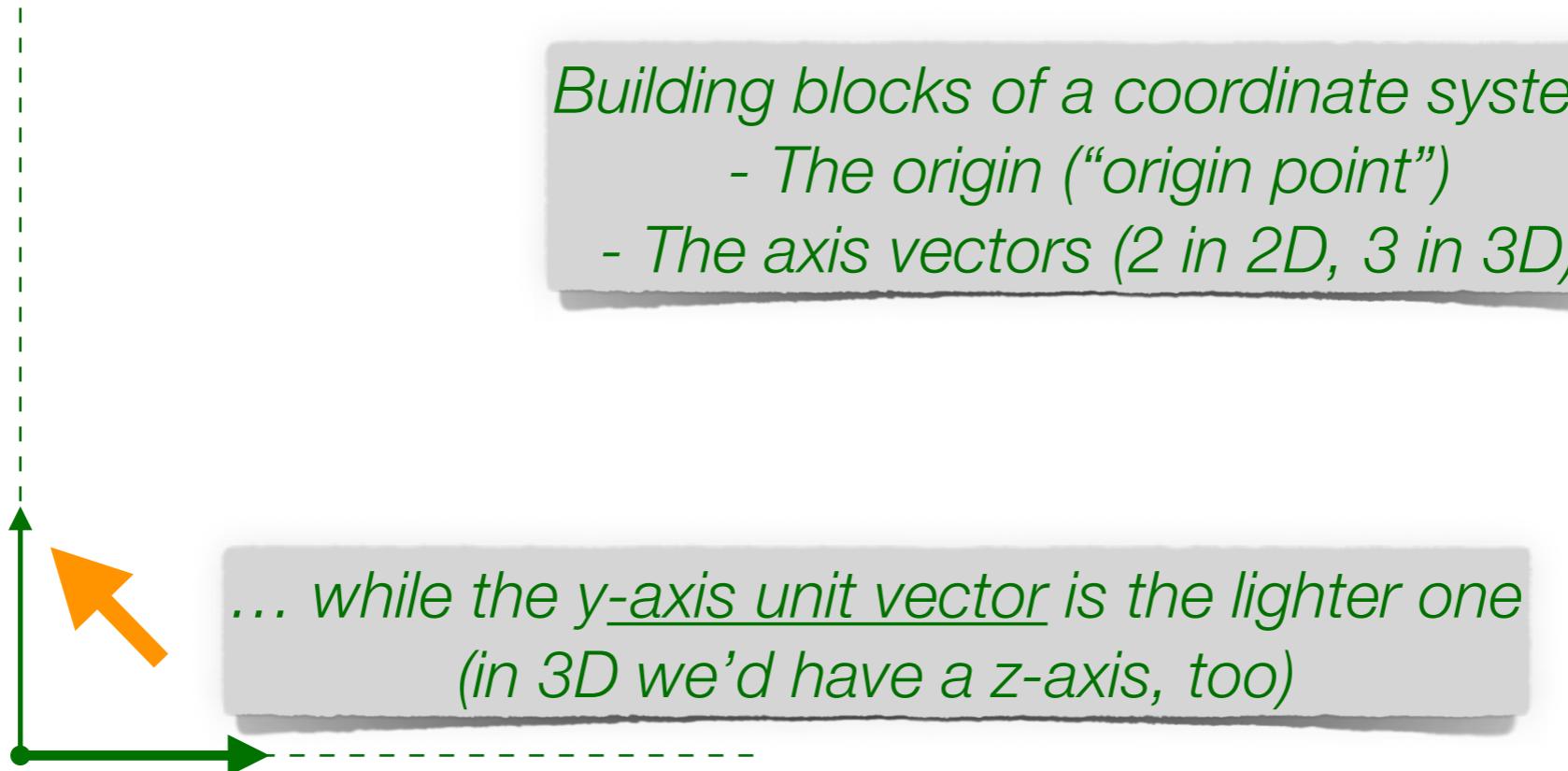


Building blocks of a coordinate system

- *The origin (“origin point”)*
- *The axis vectors (2 in 2D, 3 in 3D)*

The x-axis unit vector will be shown as the thicker of the two arrows ...

Coordinate systems and transforms



Coordinate systems and transforms

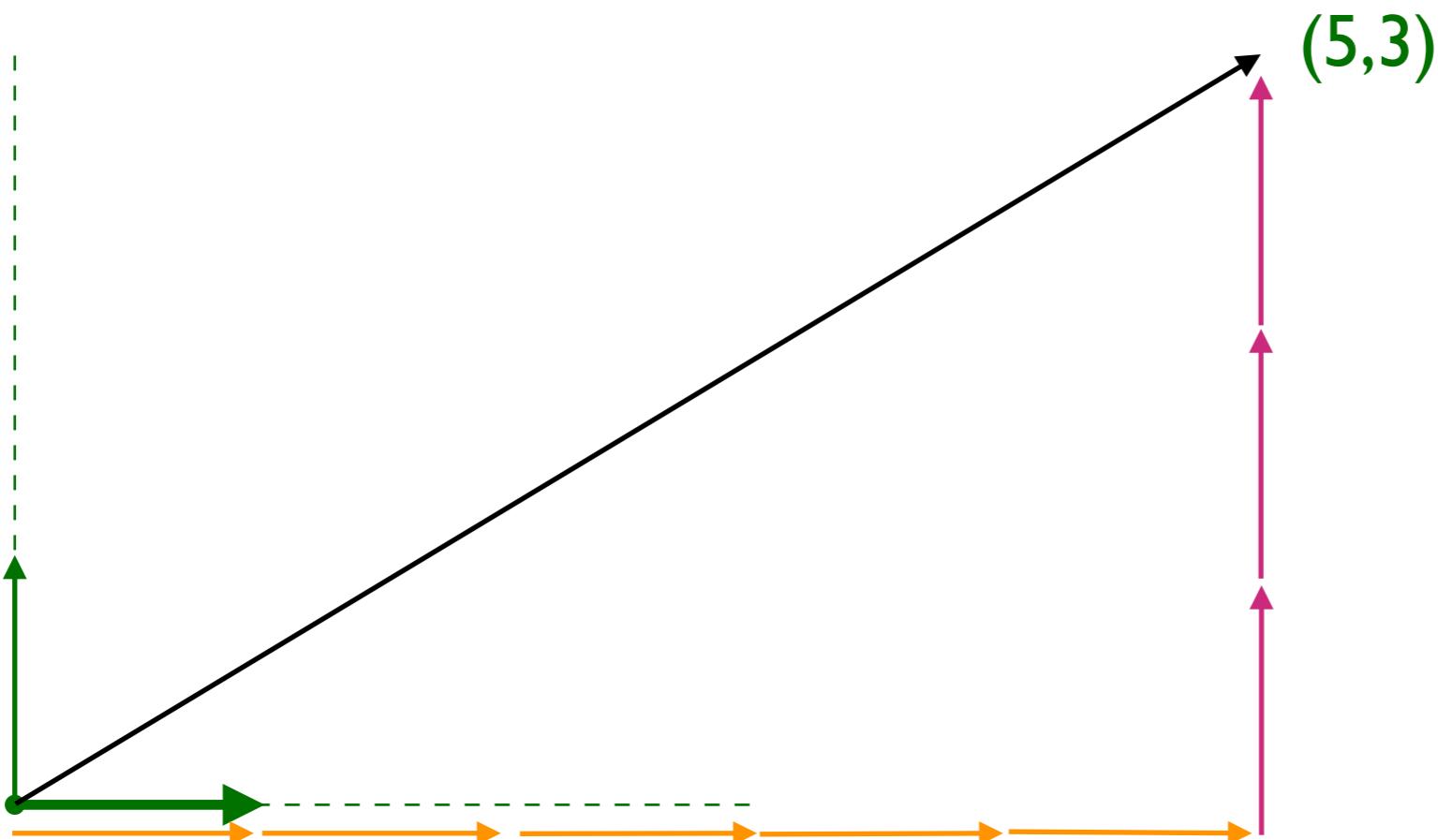
Once we have defined a coordinate system, points (and vectors) can acquire numerical identifiers ...



... a point identified as the coordinate (a,b) can be reached from the origin by stepping a -times along the x-axis vector and b -times along the y-axis vector

Coordinate systems and transforms

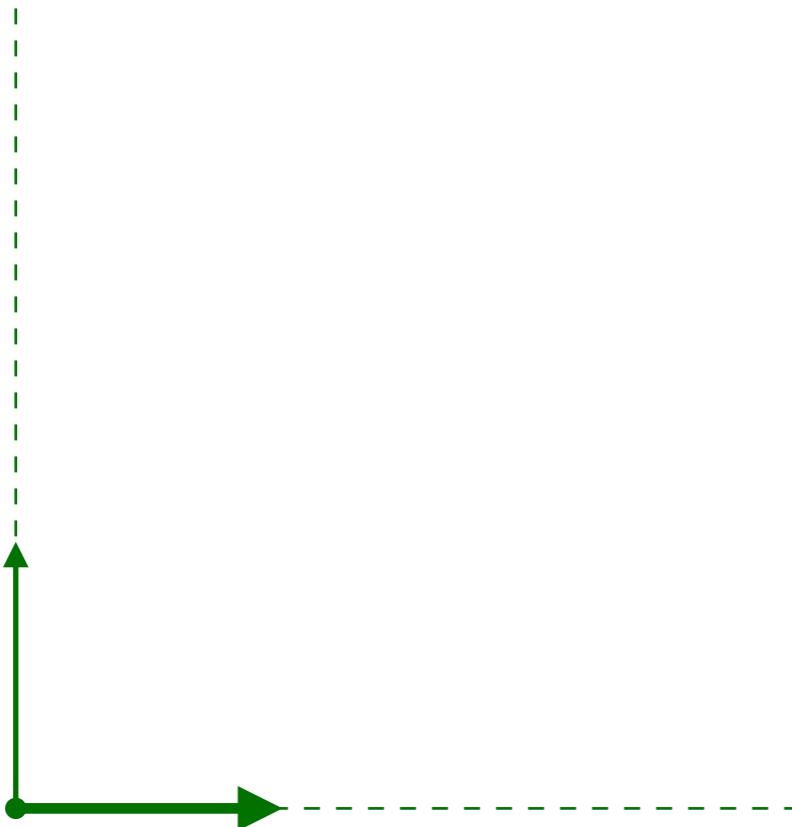
Once we have defined a coordinate system, points (and vectors) can acquire numerical identifiers ...



*... and the same coordinate notation identifies vectors
(seen here drawn from the origin)*

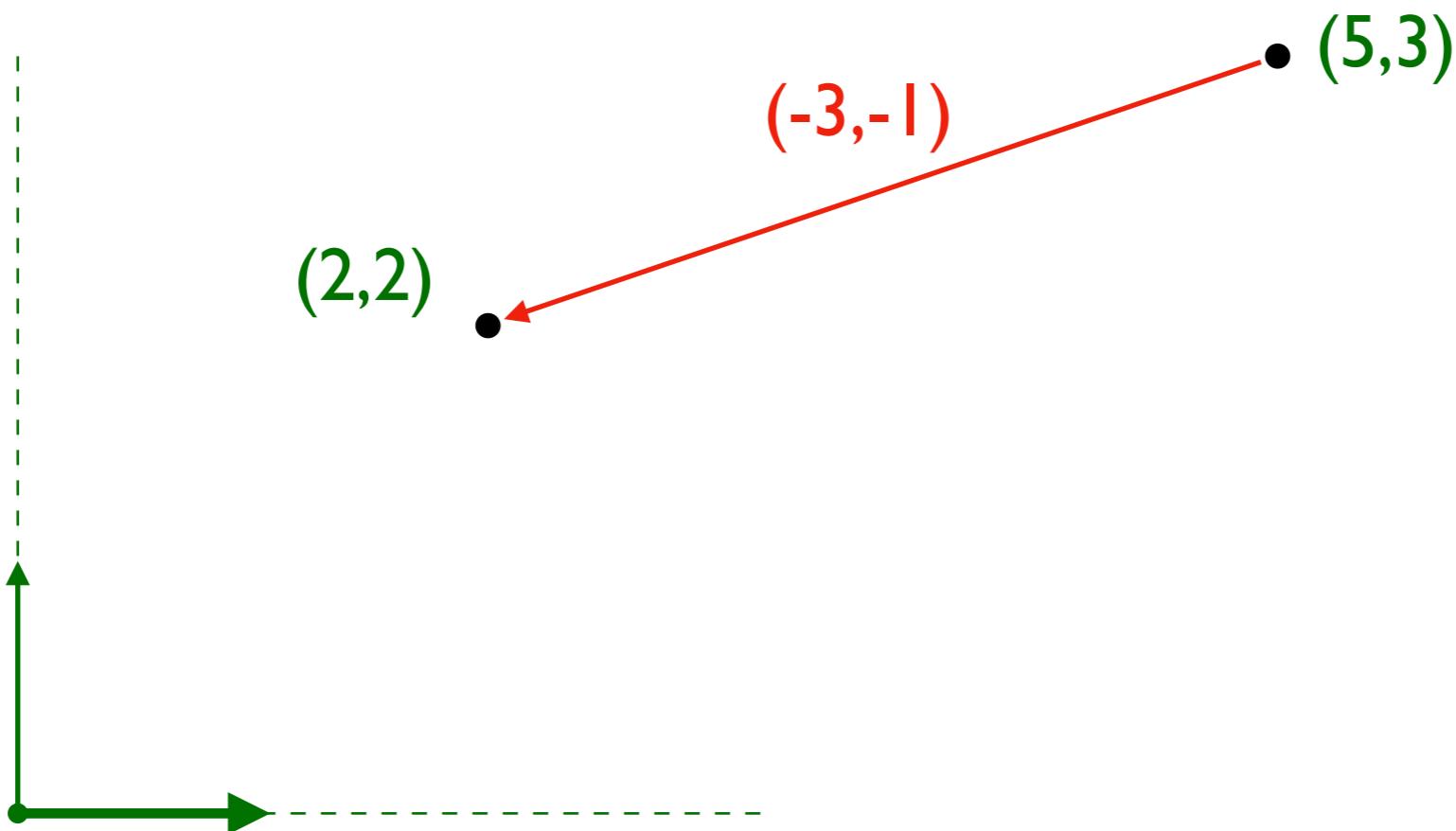
Coordinate systems and transforms

Coordinates help make geometric operations computable using algebraic operations ...



Coordinate systems and transforms

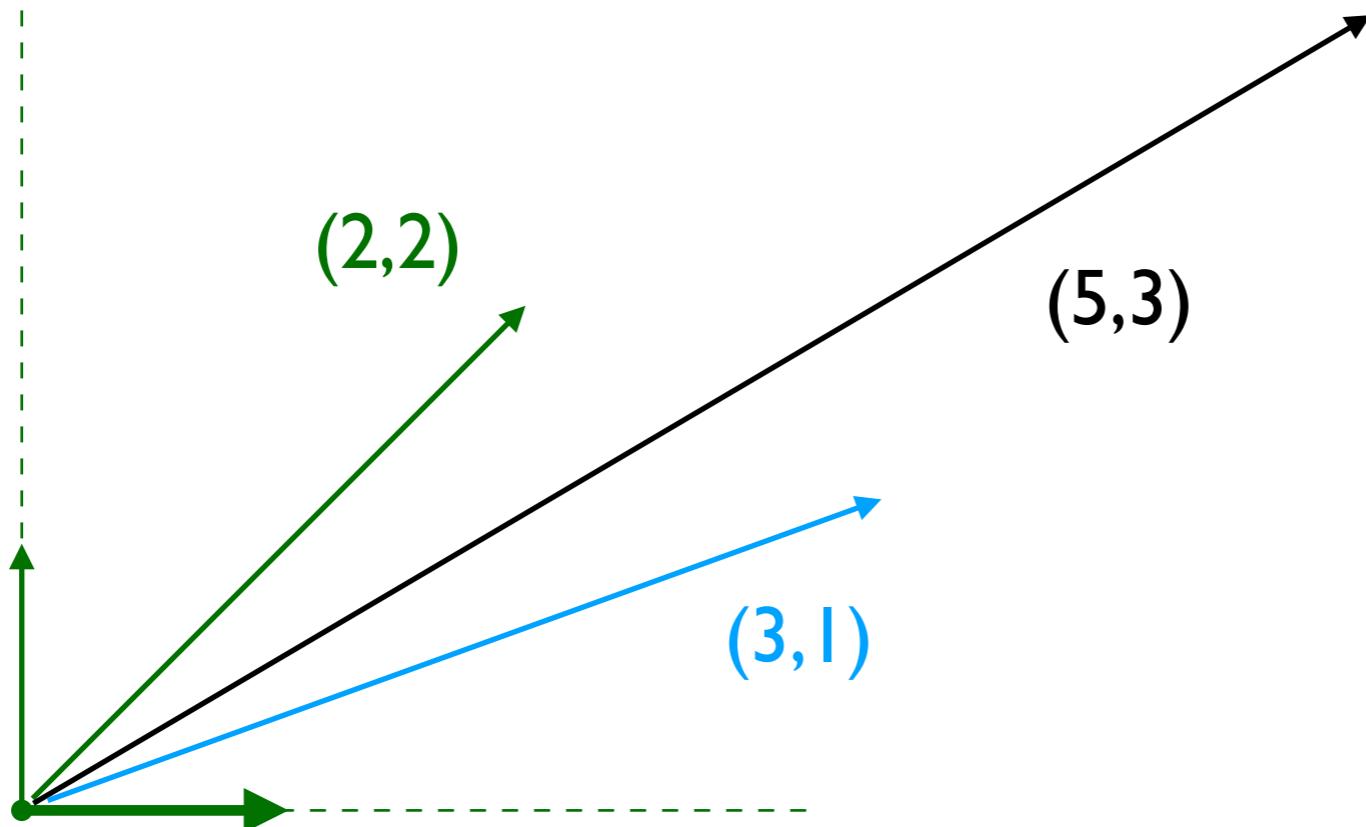
Coordinates help make geometric operations computable using algebraic operations ...



*... such as moving from a point in the direction of a vector
(just add the coordinate notations of point and vector)*

Coordinate systems and transforms

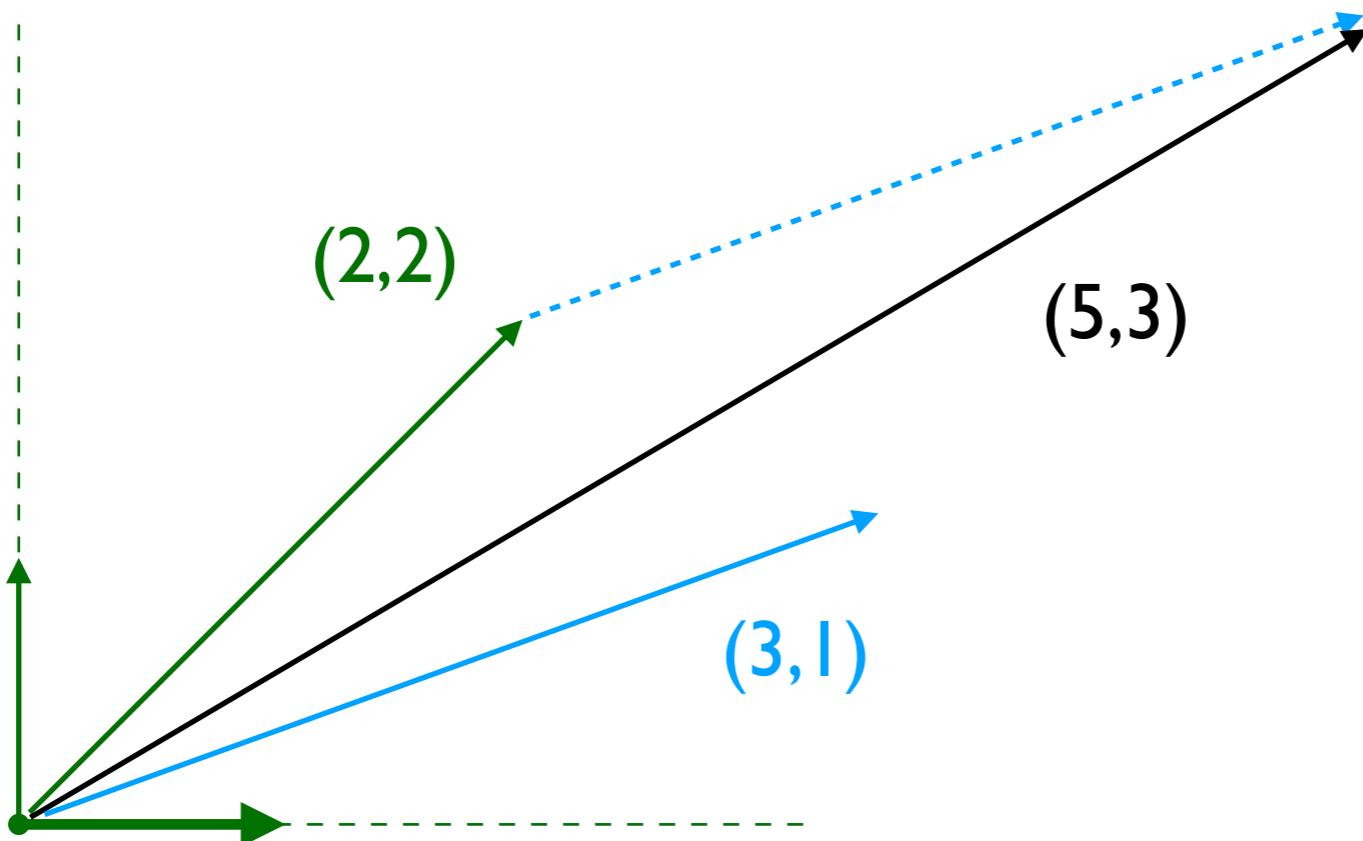
Coordinates help make geometric operations computable using algebraic operations ...



...or adding two vectors together (remember rectangle rule?)

Coordinate systems and transforms

Coordinates help make geometric operations computable using algebraic operations ...

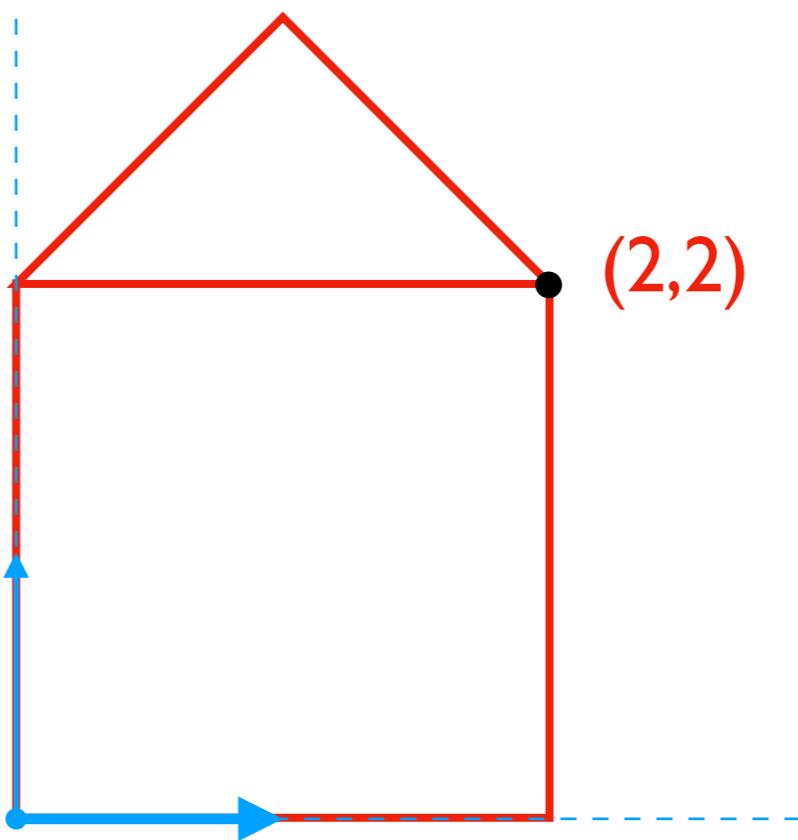


...or adding two vectors together (remember rectangle rule?)

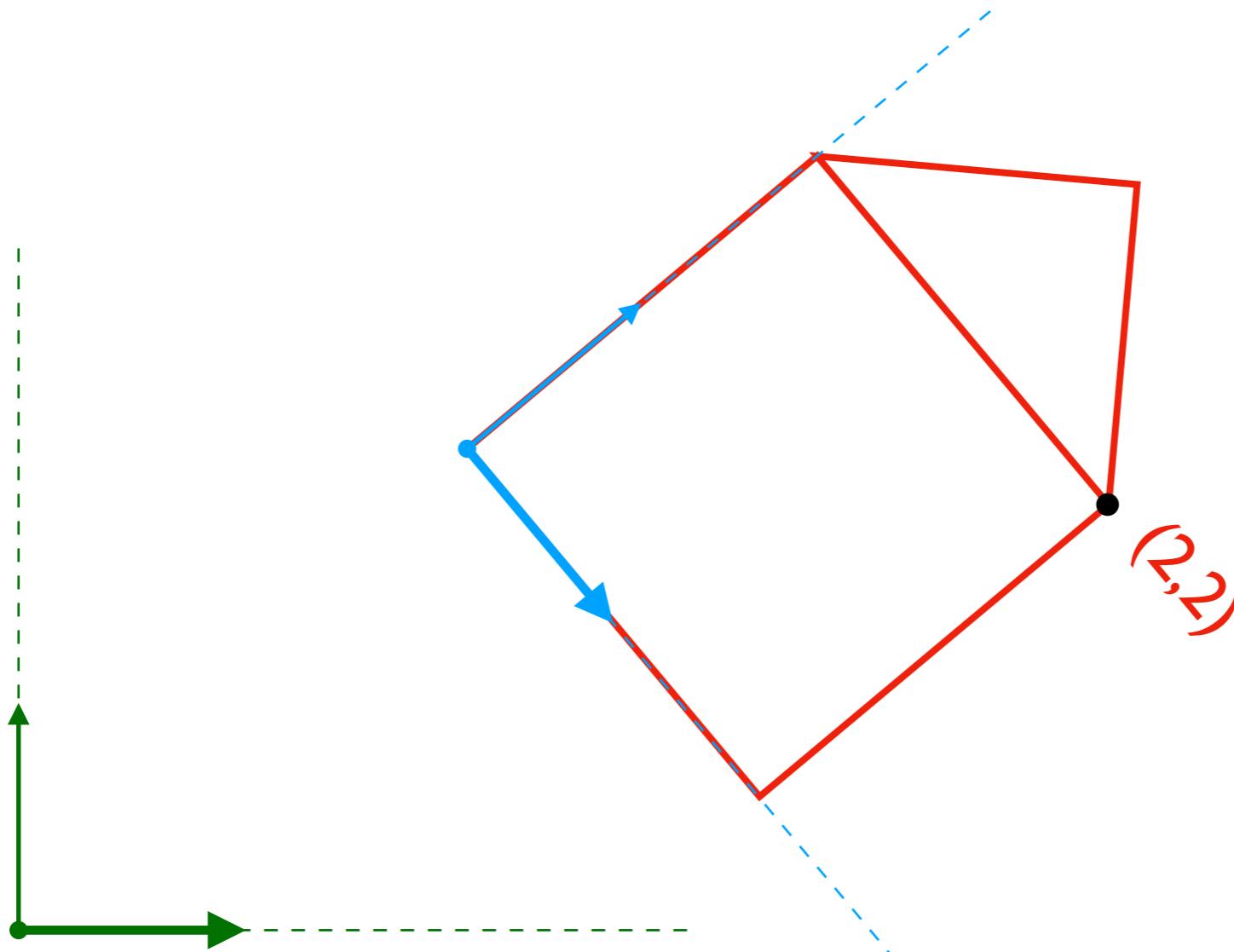
Coordinate systems and transforms

- There are instances where multiple coordinate systems might be present in a scene (either because they inherently exist, or because we chose to create them)
- It could easily be the case that drawing some shape (more accurately: prescribing the geometry of a given shape) is more easily done in one coordinate system rather than another
- We should always strive to work on the most convenient coordinate system for a given task

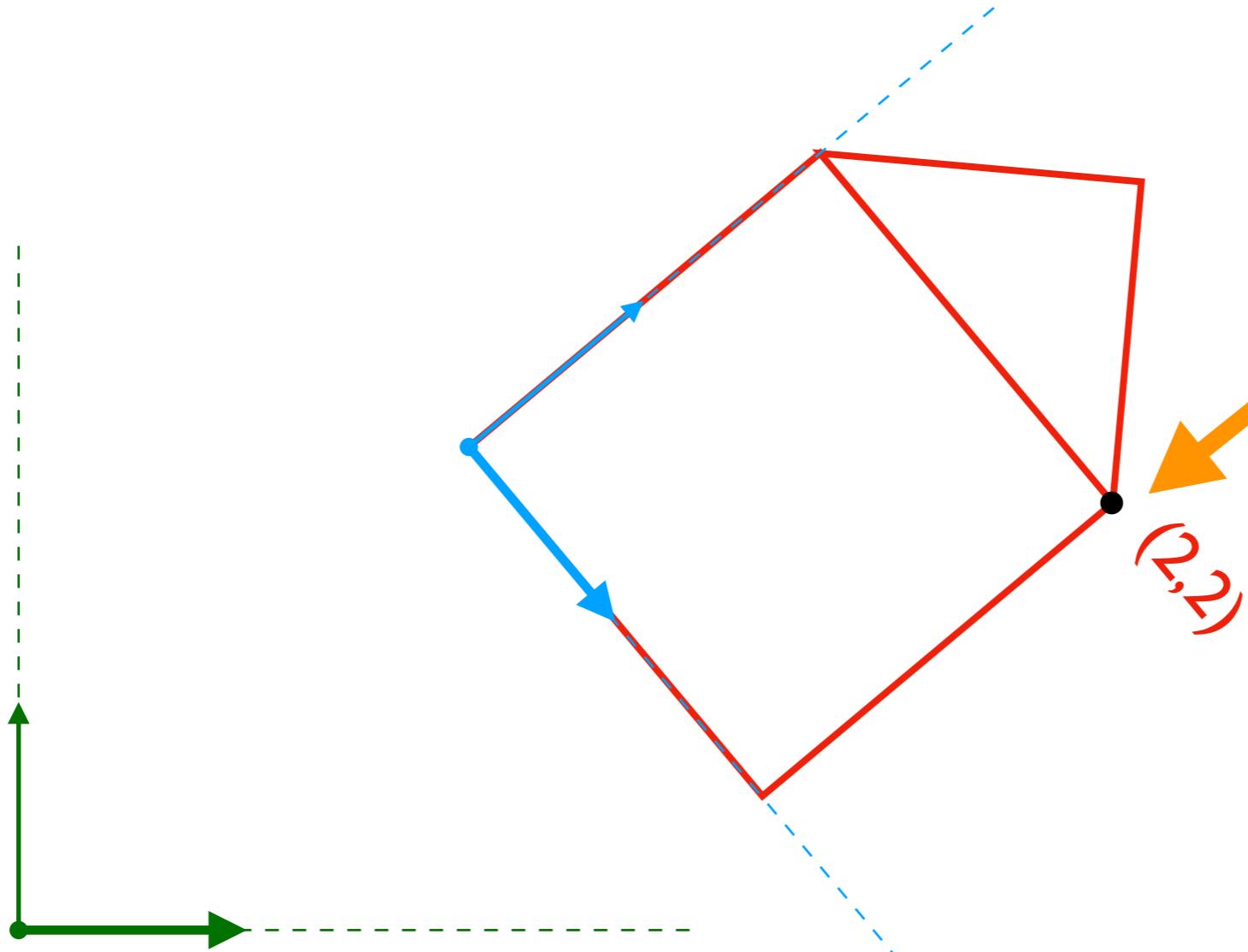
Coordinate systems and transforms



Coordinate systems and transforms

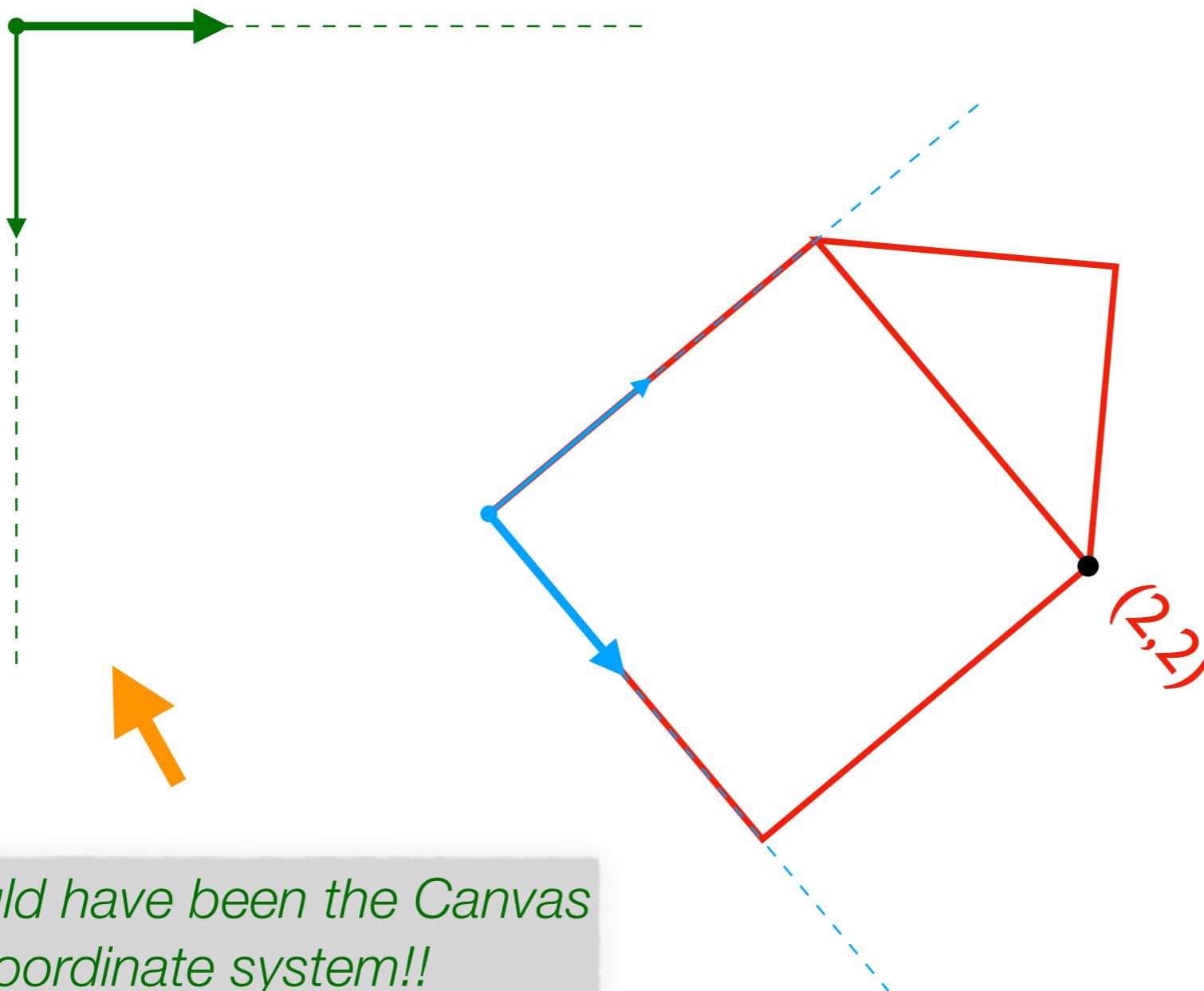


Coordinate systems and transforms



*What are the “green”
coordinates
of this point?*

Coordinate systems and transforms



*This could have been the Canvas
coordinate system!!*

A great motivation: Hierarchical modeling

- Very often we model scenes that include repeated instances of very similar shapes, but:
 - The different instances can include displaced, rotated, or stretched versions of a “master copy”
 - The placement of some instances could be subordinate to the placement of others
 - There might be a “natural” coordinate system to define the shape of the master instance, but that system might be very different from the intended placement
 - (More on hierarchical modeling in next lecture!)