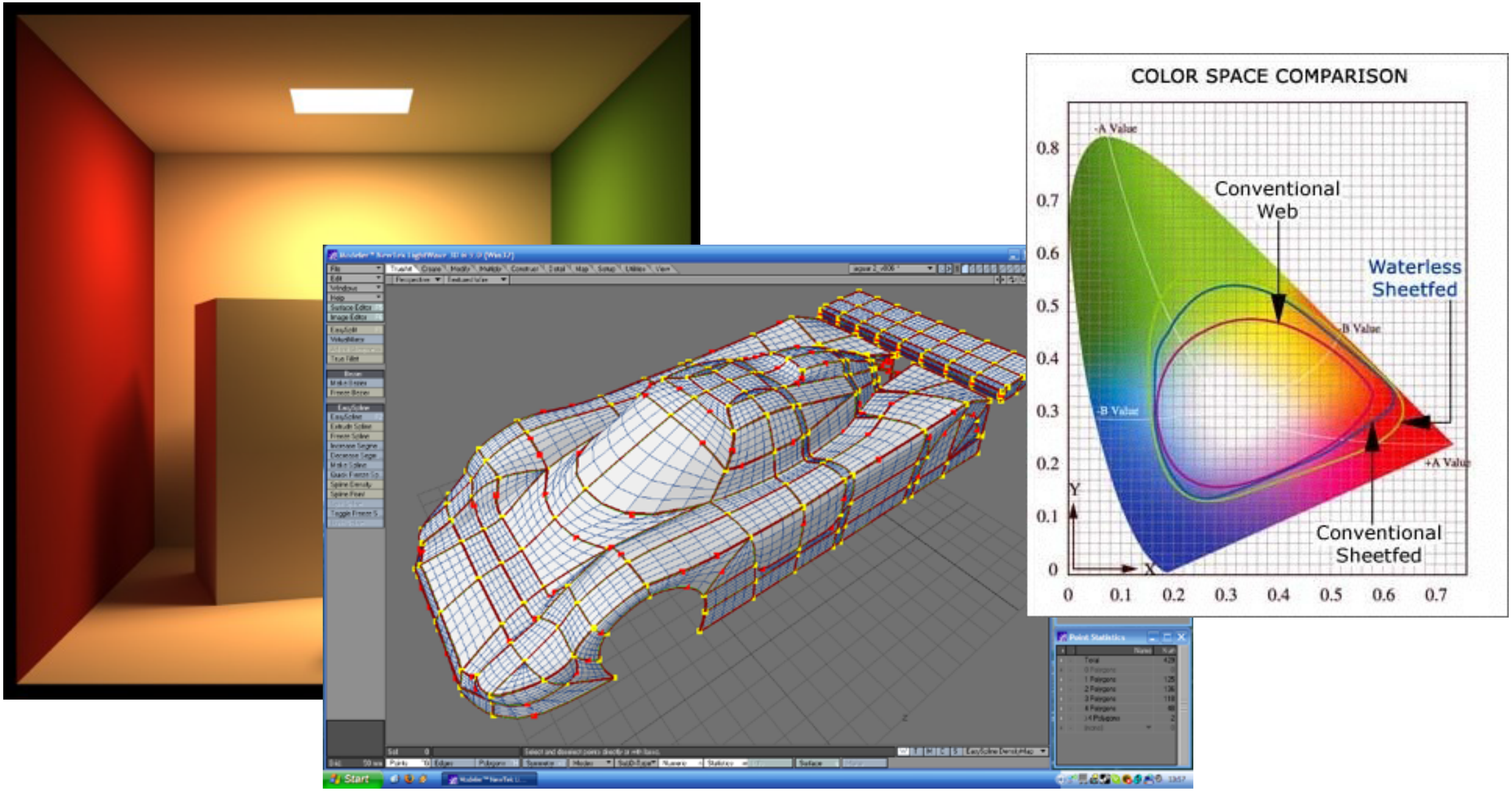


Lecture 4 : Coordinate systems and transforms in 2D (with an introduction to transforms in Canvas)

Tuesday September 21st 2021

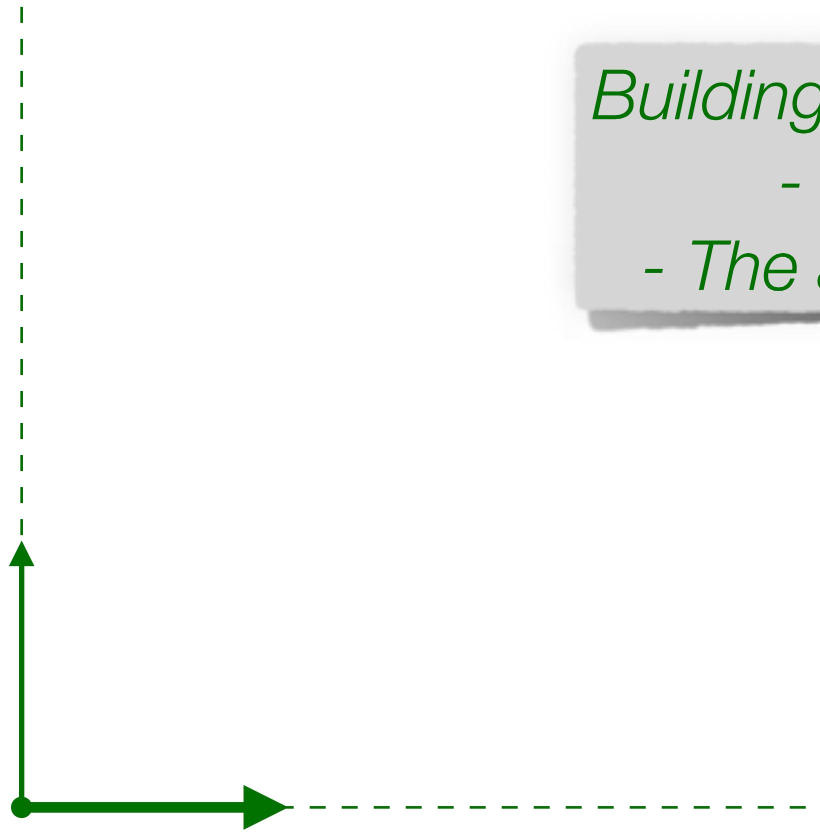
Administrative stuff

- Programming Assignment Due on Friday!
- Post questions on Piazza, or drop in to office hours!
    - Hope it has been working so far ?!?
- (Blatantly repetitive reminder …)
  Do try your hardest to not do your development in JSBin. It's deceptively convenient, but it is a big trap going into the future (you want to have a proper debugging environment).

Today's lecture

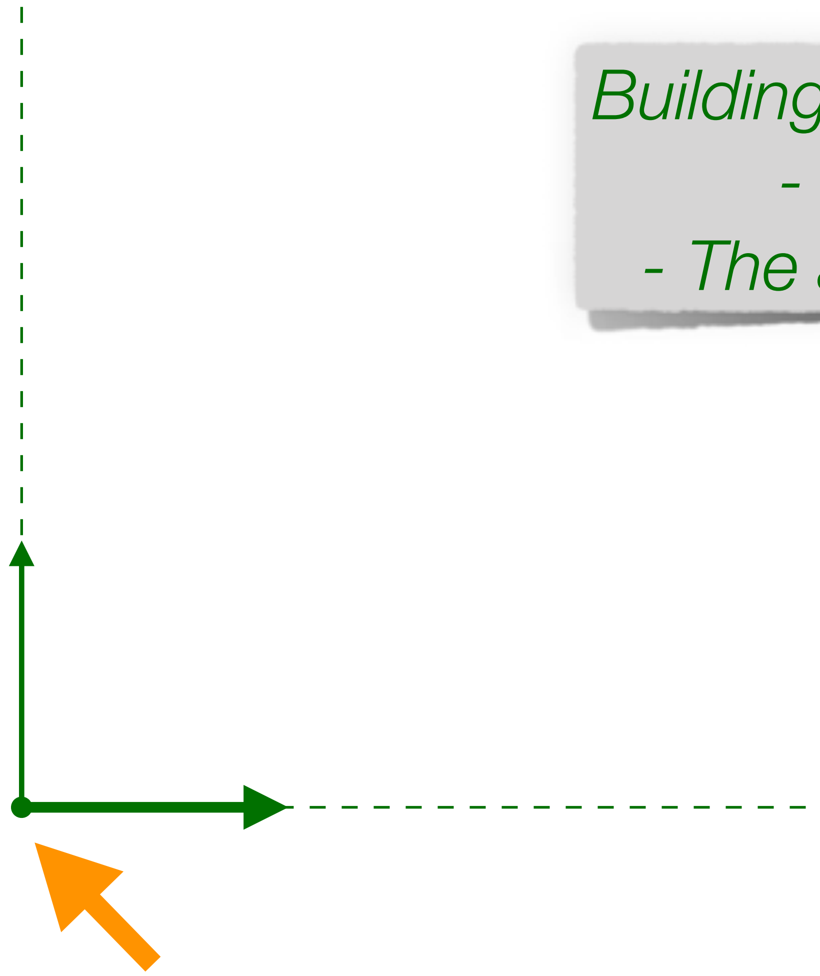
- Discussion of coordinate systems, and their utility in modeling geometric shapes
  - Heed this advice:
    always define convenient coordinate systems to work in
- Intro to transforms between coordinate systems
  - Description of elementary transforms:
    Translation, Scale, Rotation
  - Motivation of studying and using transforms
  - Chaining transforms together
  - How are transforms implemented in Canvas?

Coordinate systems and transforms



Building blocks of a coordinate system
- The origin (“origin point”)
- The axis vectors (2 in 2D, 3 in 3D)

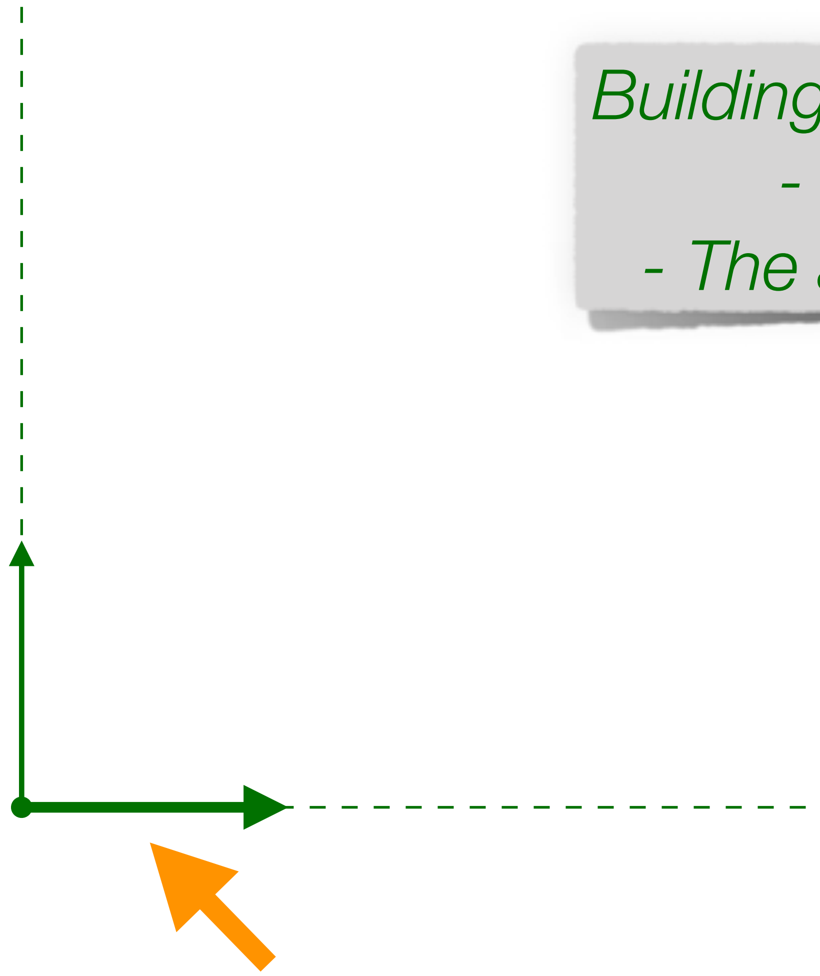
Coordinate systems and transforms



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

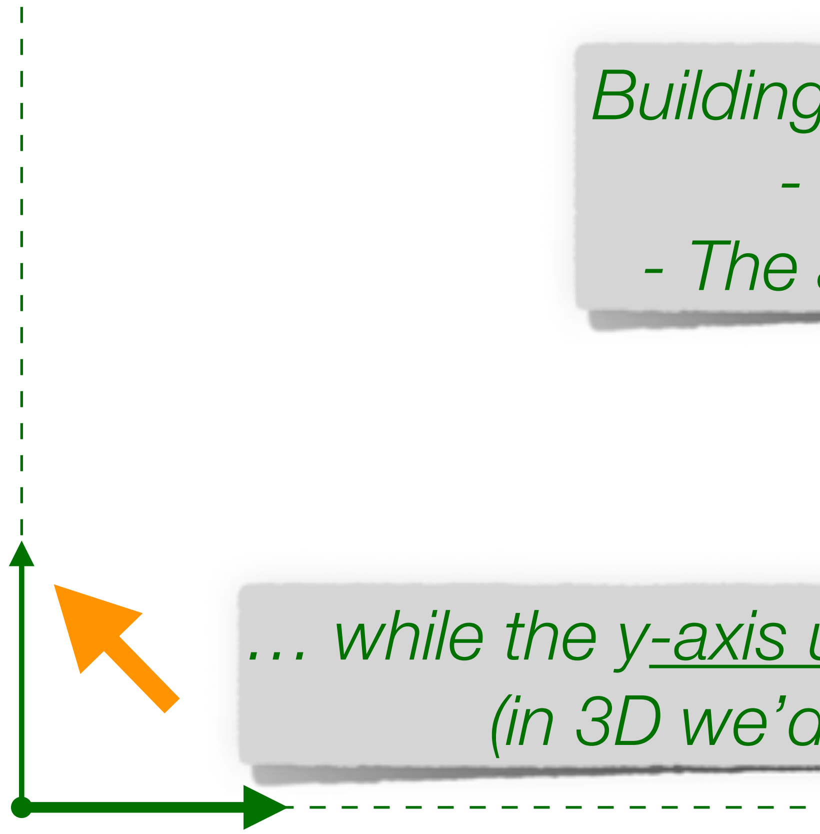


Building blocks of a coordinate system
- The origin (“origin point”)
- The axis vectors (2 in 2D, 3 in 3D)

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

Coordinate systems and transforms

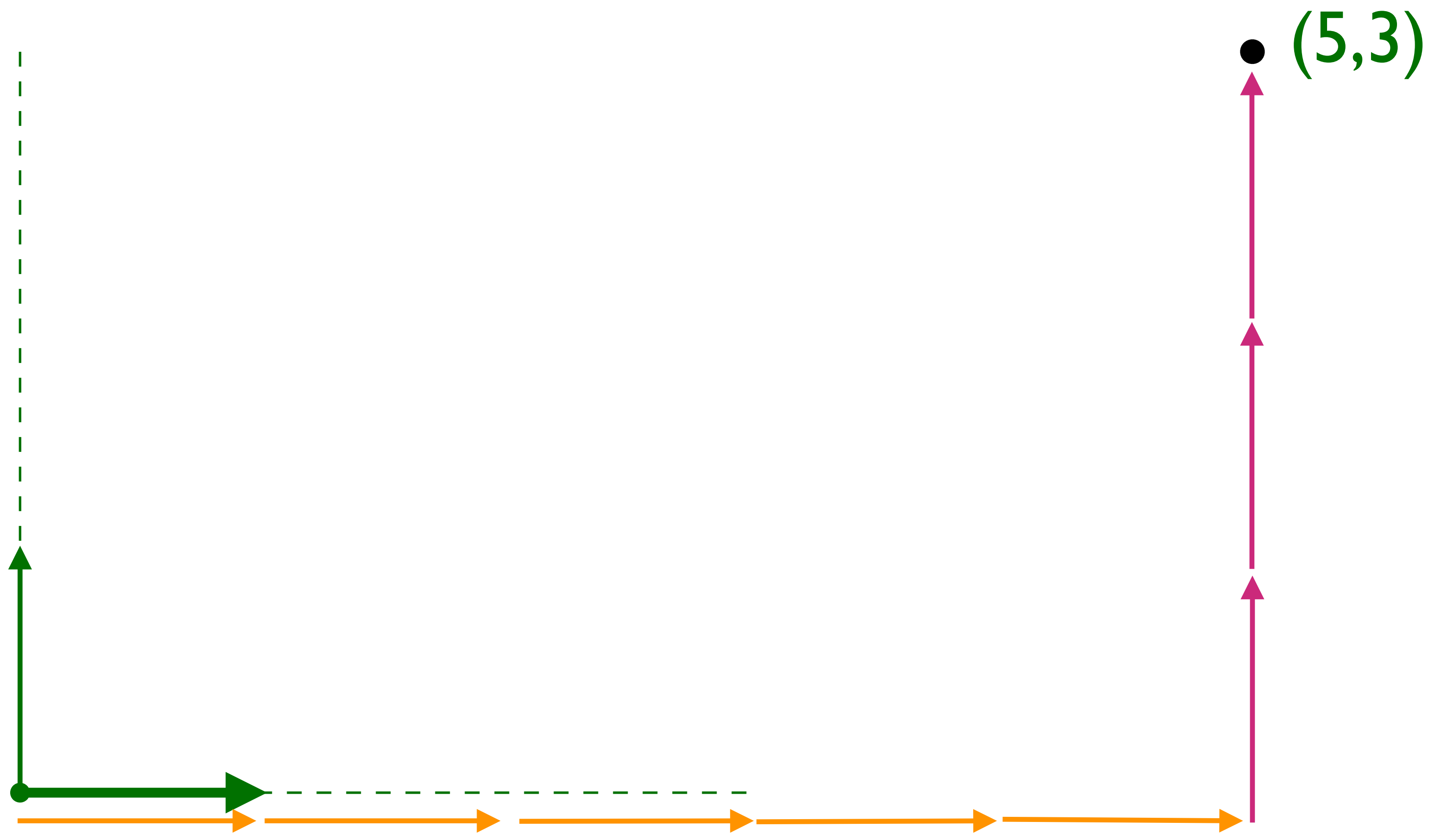
Building blocks of a coordinate system
- The origin ("origin point")
- The axis vectors (2 in 2D, 3 in 3D)



… while the y-axis unit vector is the lighter one
(in 3D we'd have a z-axis, too)

Coordinate systems and transforms

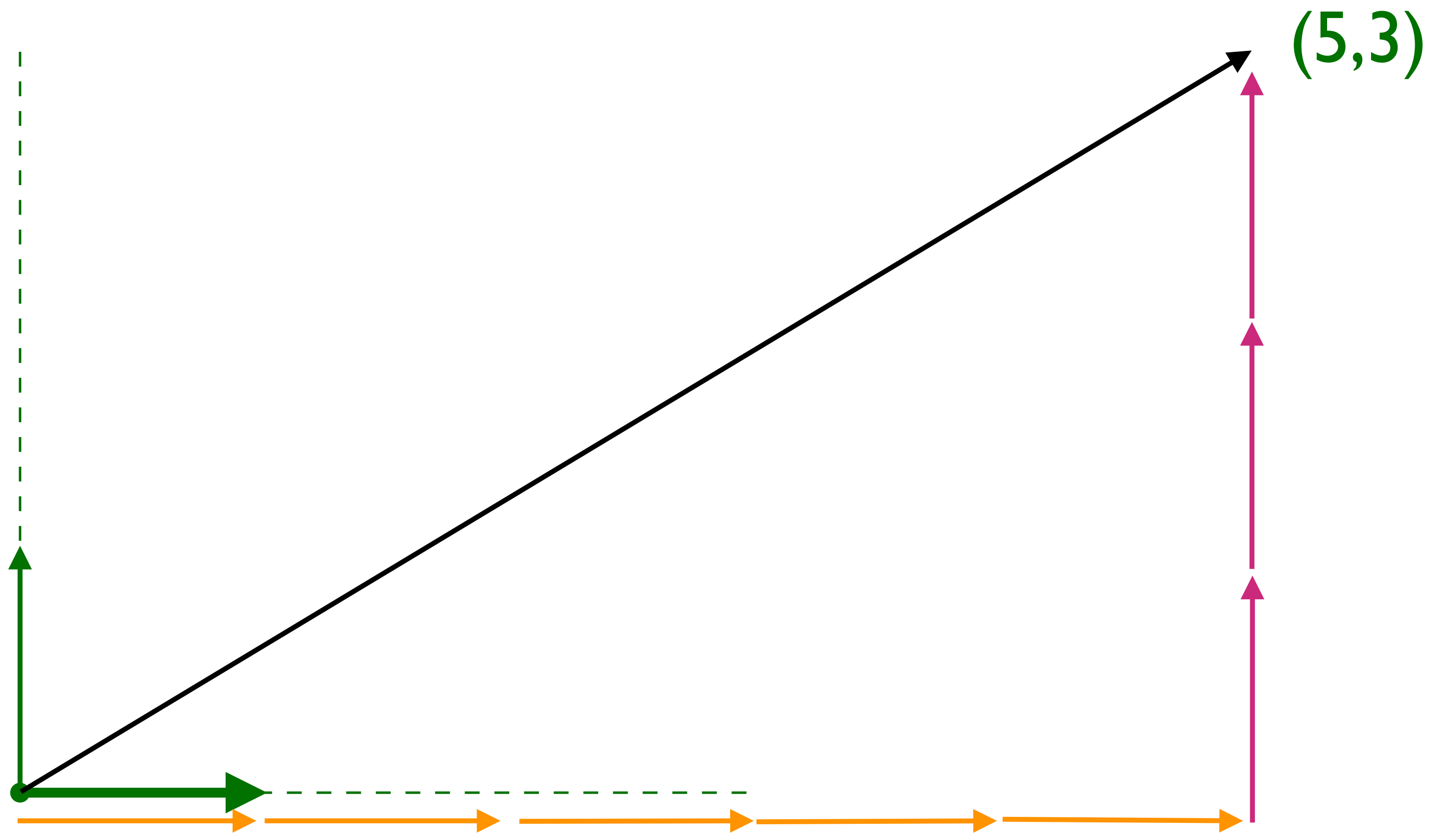
Once we have defined a coordinate system, points (and vectors) can acquire numerical identifiers …



… 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

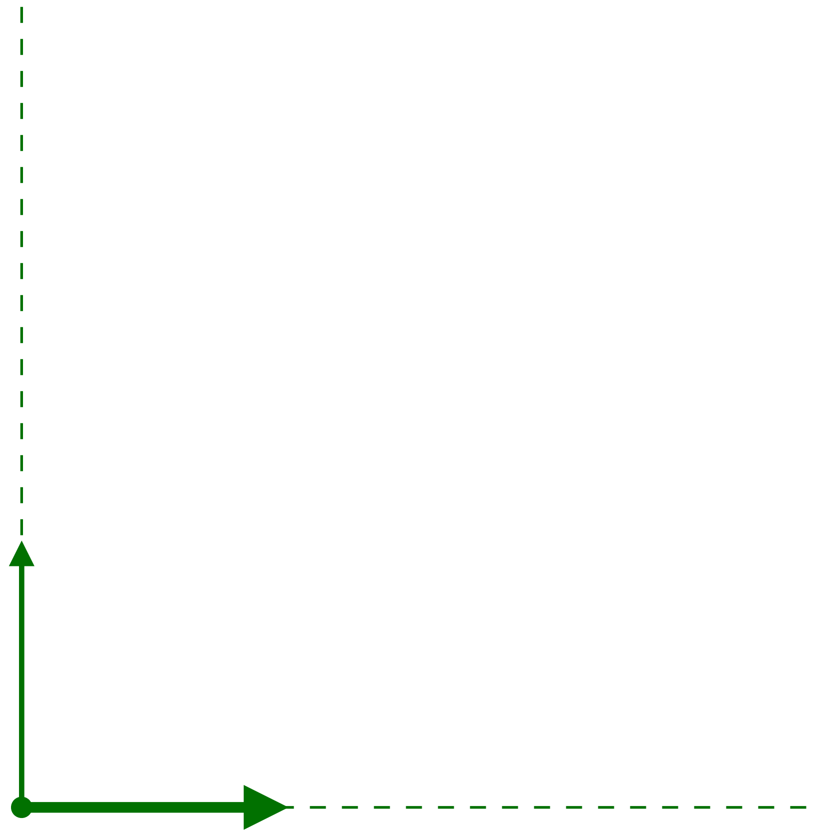
Once we have defined a coordinate system, points (and vectors) can acquire numerical identifiers …



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

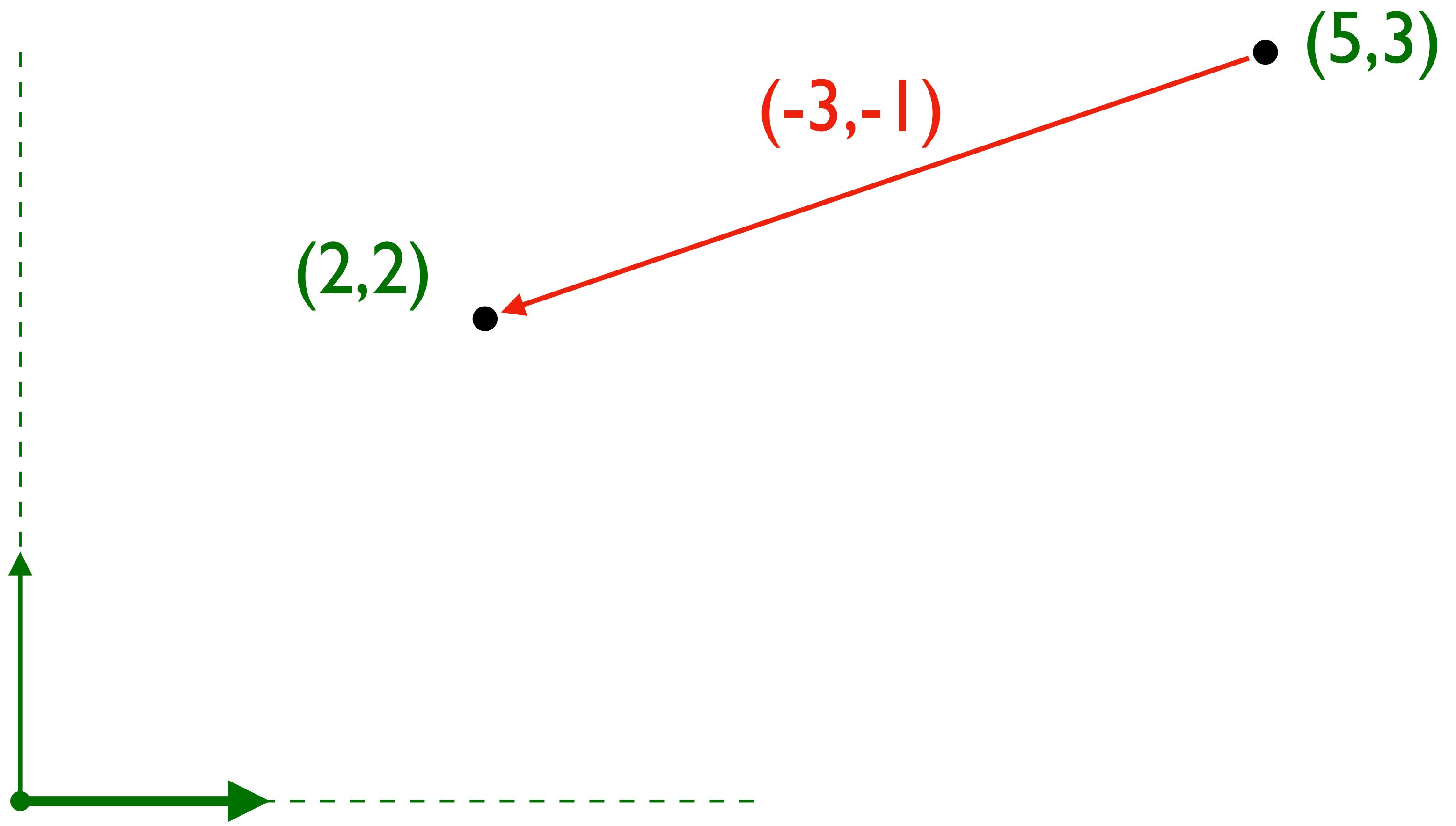
Coordinate systems and transforms

Coordinates help make geometric operations computable using algebraic operations …



Coordinate systems and transforms

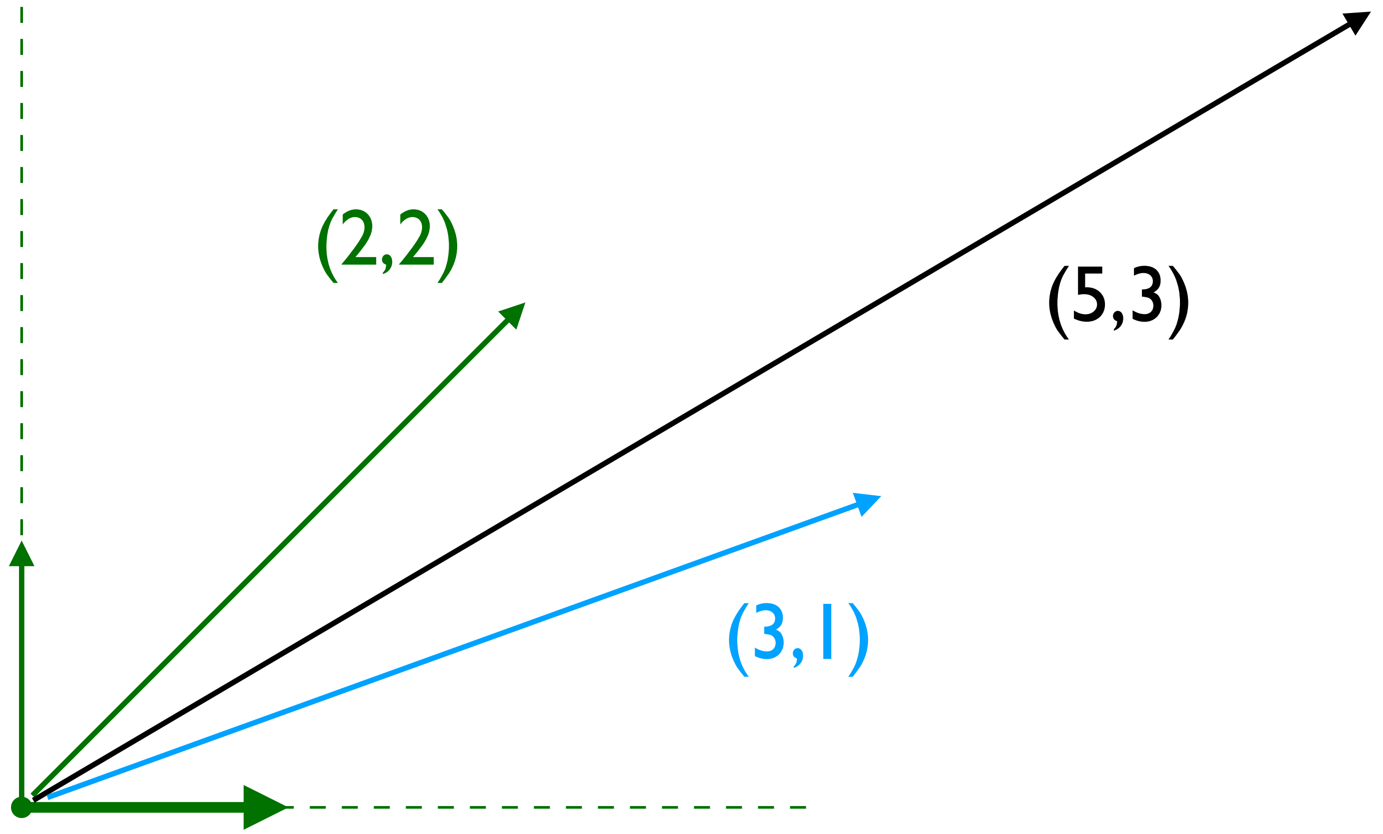
Coordinates help make geometric operations computable using algebraic operations …



… 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

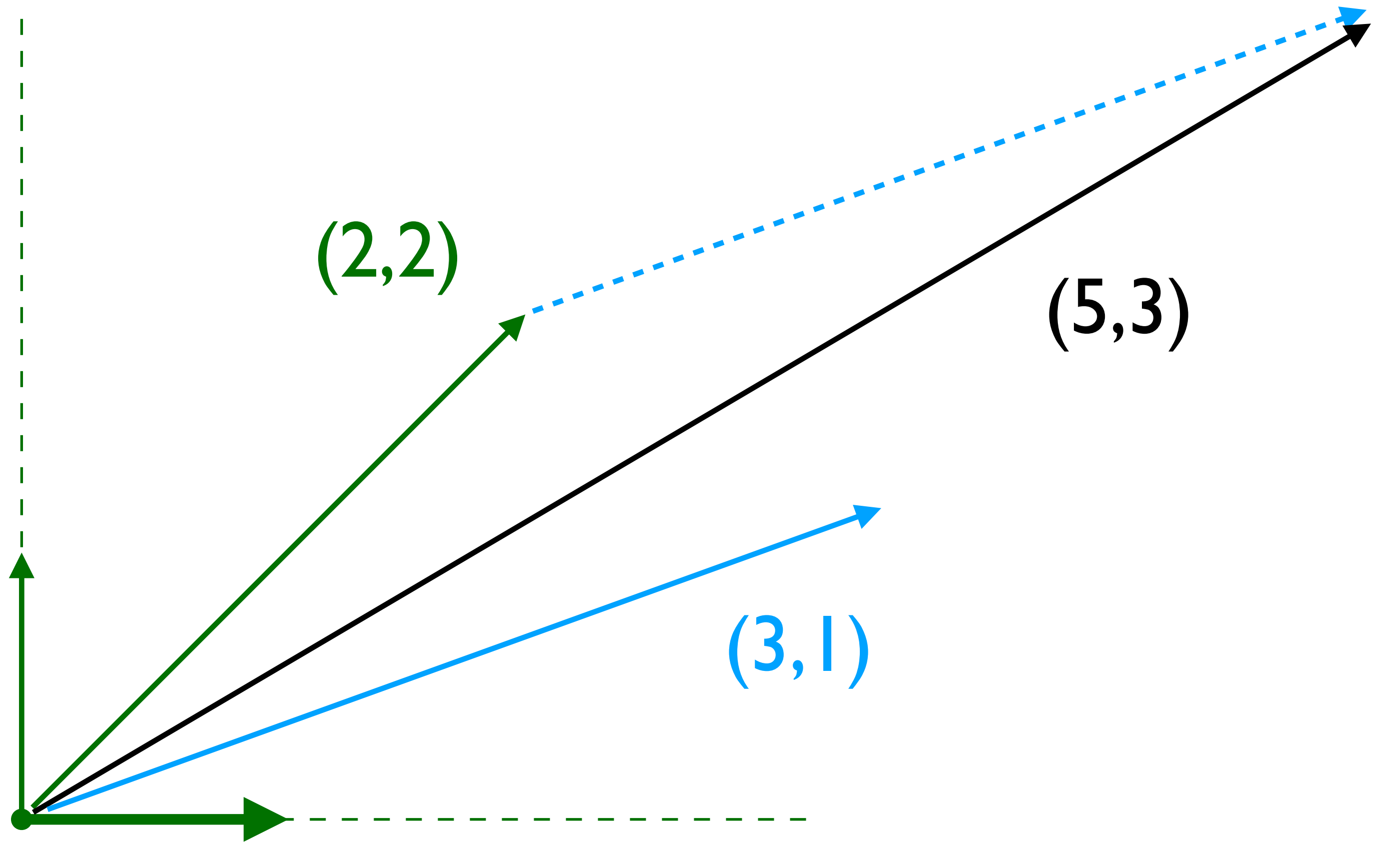
Coordinates help make geometric operations computable using algebraic operations …



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

Coordinate systems and transforms

Coordinates help make geometric operations computable using algebraic operations …

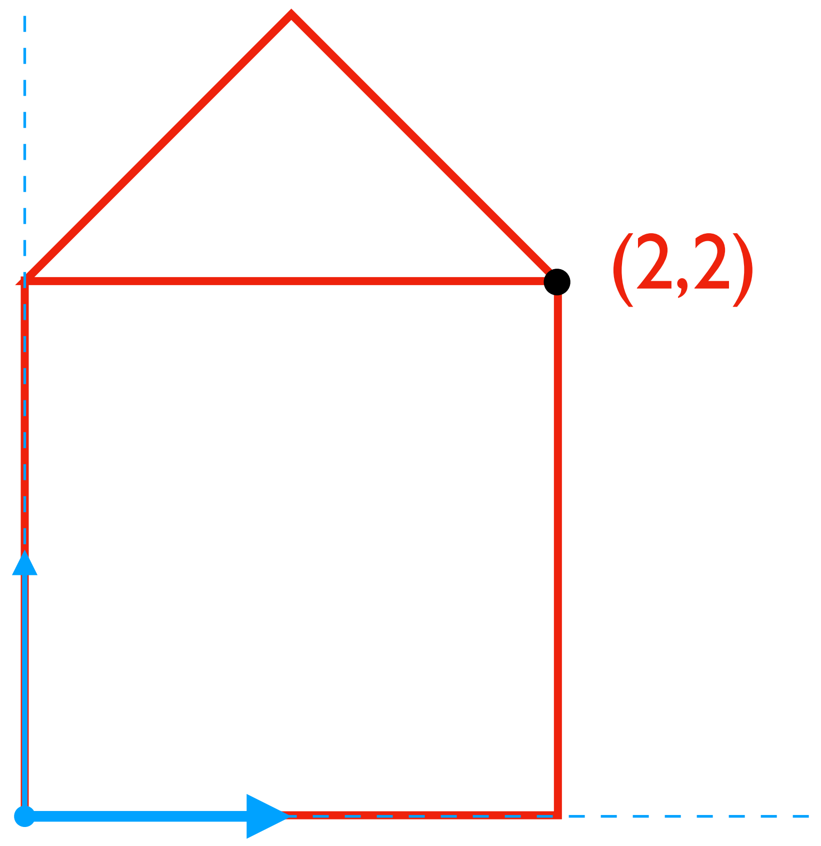


…or adding two vectors together (remember rectangle rule?)

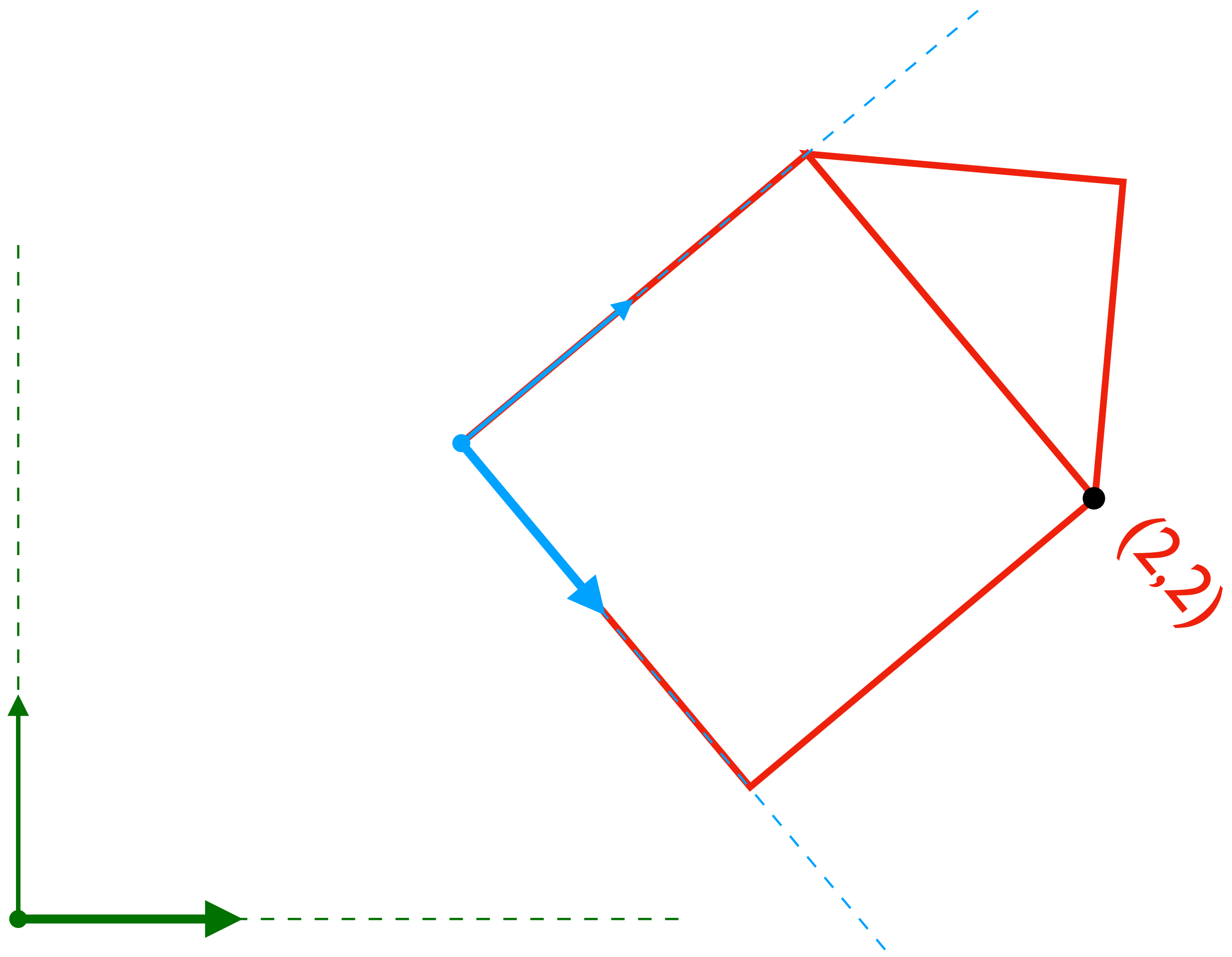
Coordinate systems and transforms

- 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

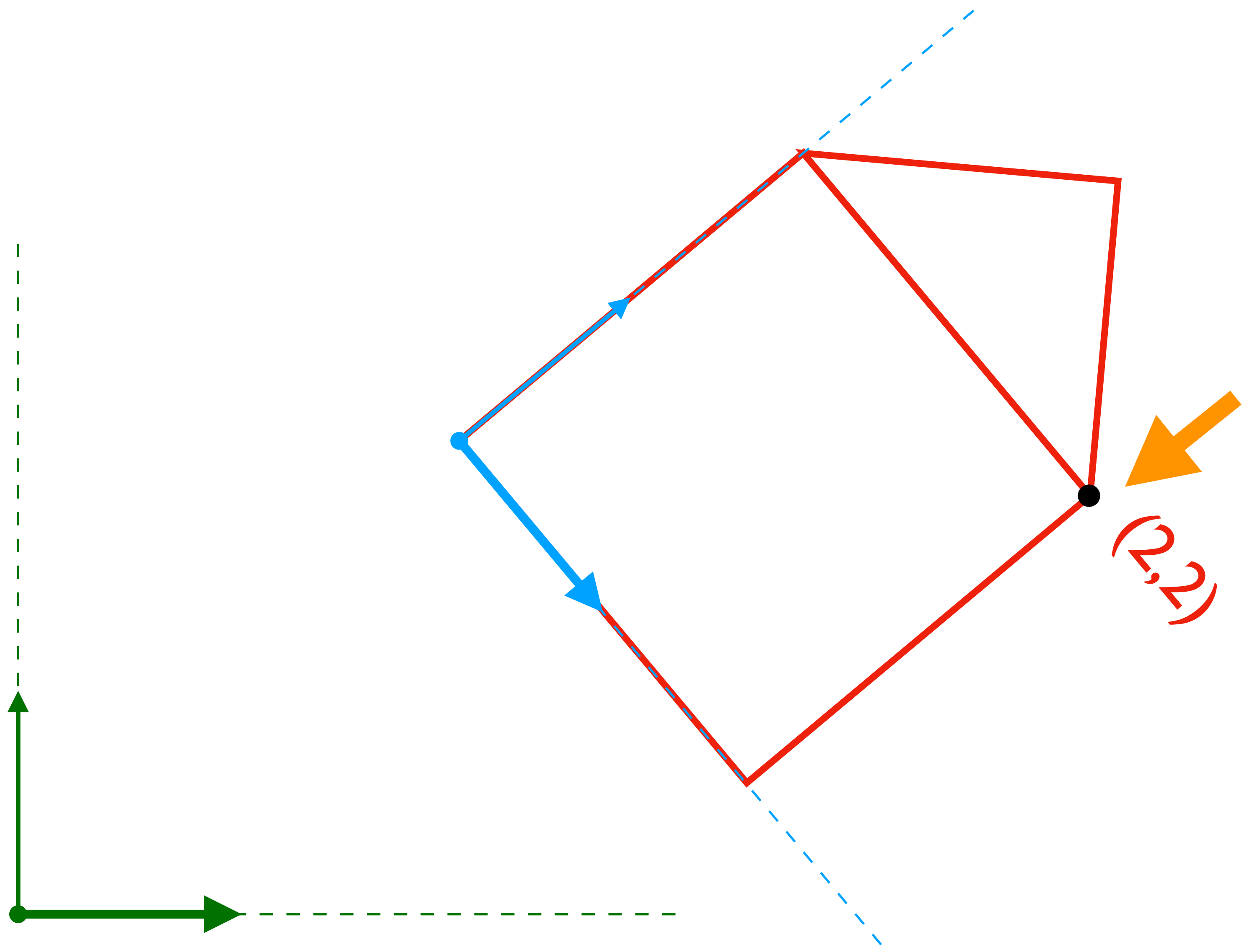
Coordinate systems and transforms



Coordinate systems and transforms

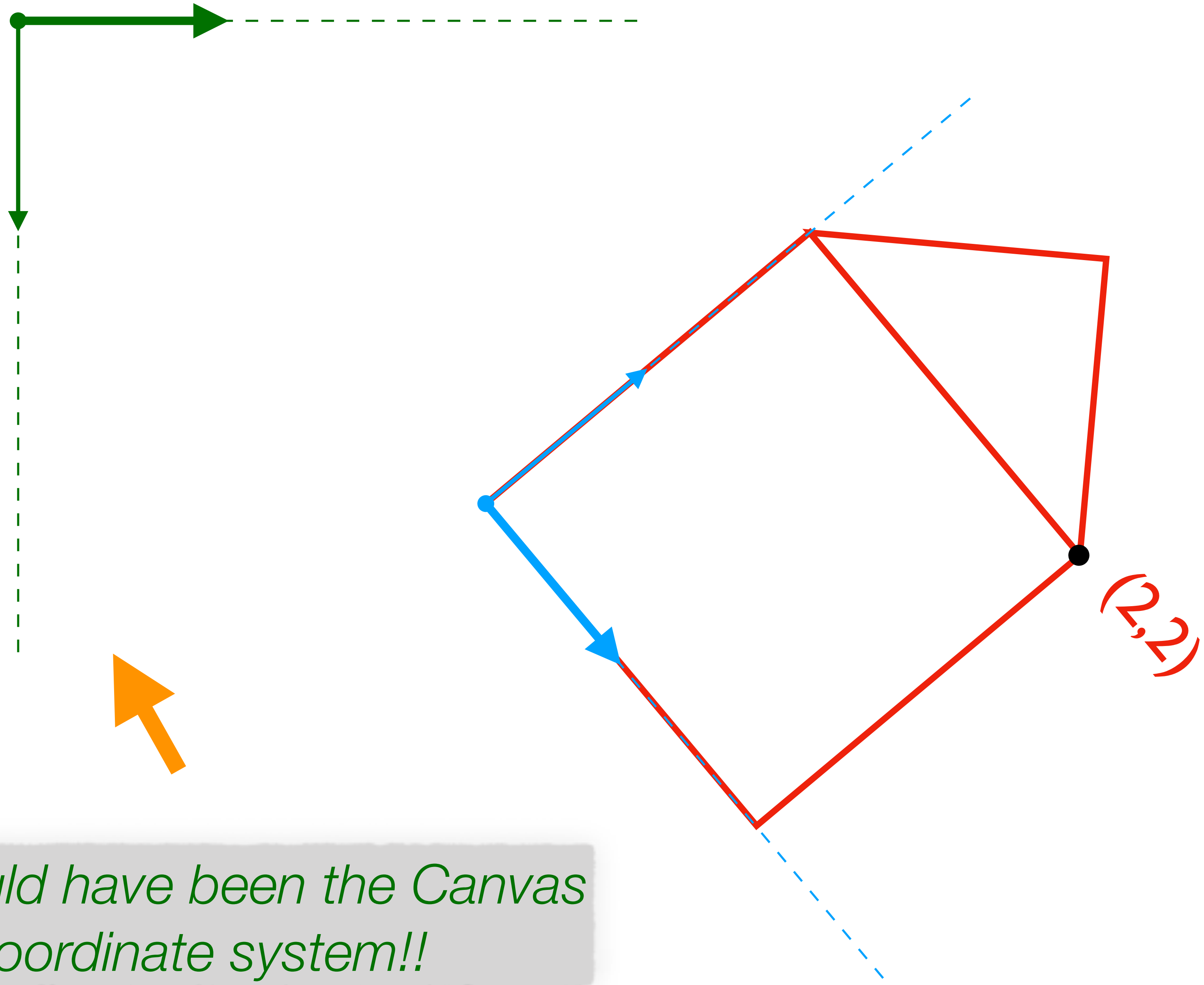


Coordinate systems and transforms



What are the "green" coordinates of this point?

Coordinate systems and transforms



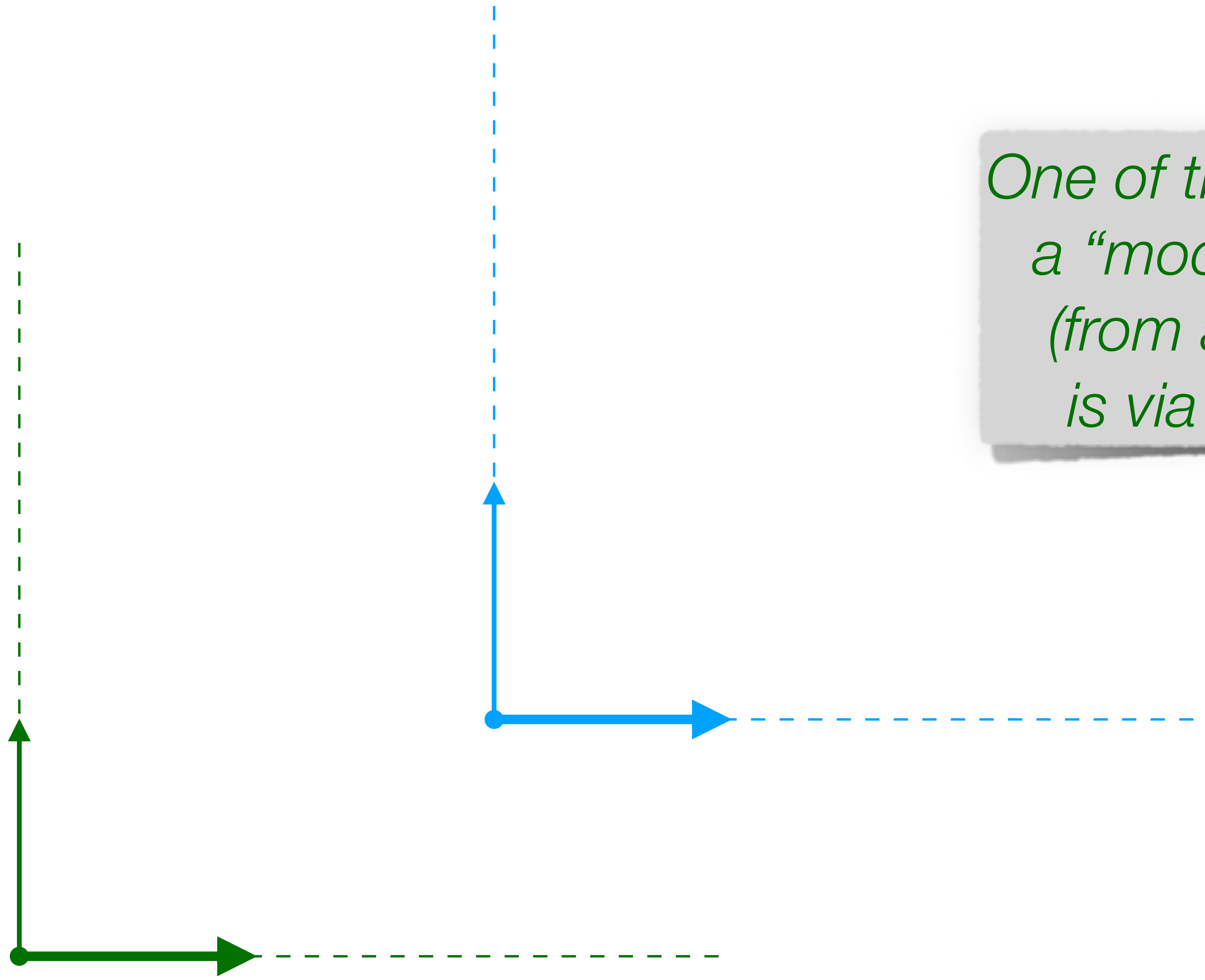
This could have been the Canvas coordinate system!!

A great motivation: Hierarchical modeling

jsbin.com/wovupusife

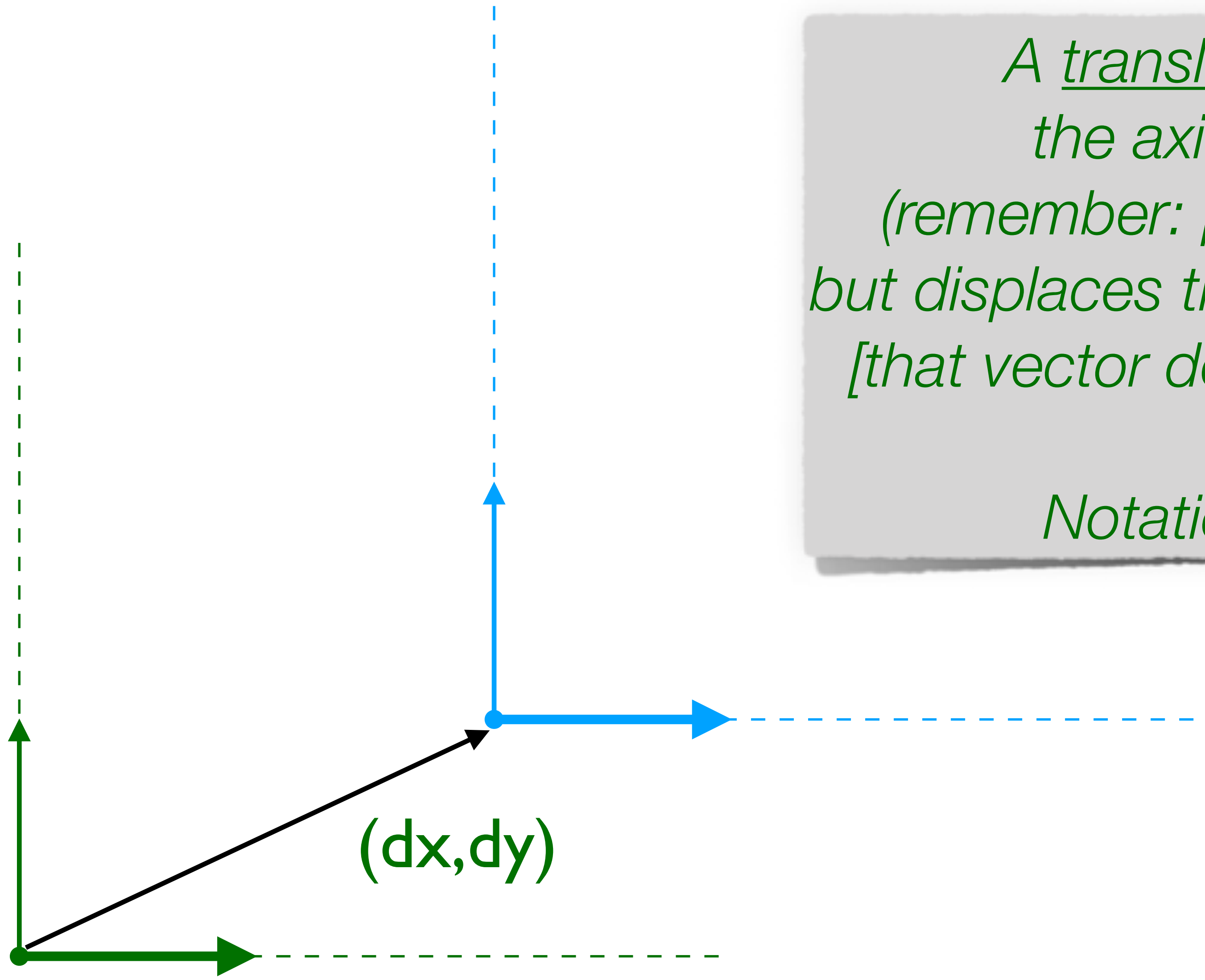
- Very often we model scenes that include repeated instances of very similar shapes, but:
  - The different instances can include displaced, rotated, or stretched versions of a "master copy"
  - The placement of some instances could be subordinate to the placement of others
  - There might be a "natural" 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!)

Elementary transforms



One of the simplest ways to create a "modified" coordinate system (from a starting/reference one) is via a translation transform.

Elementary transforms



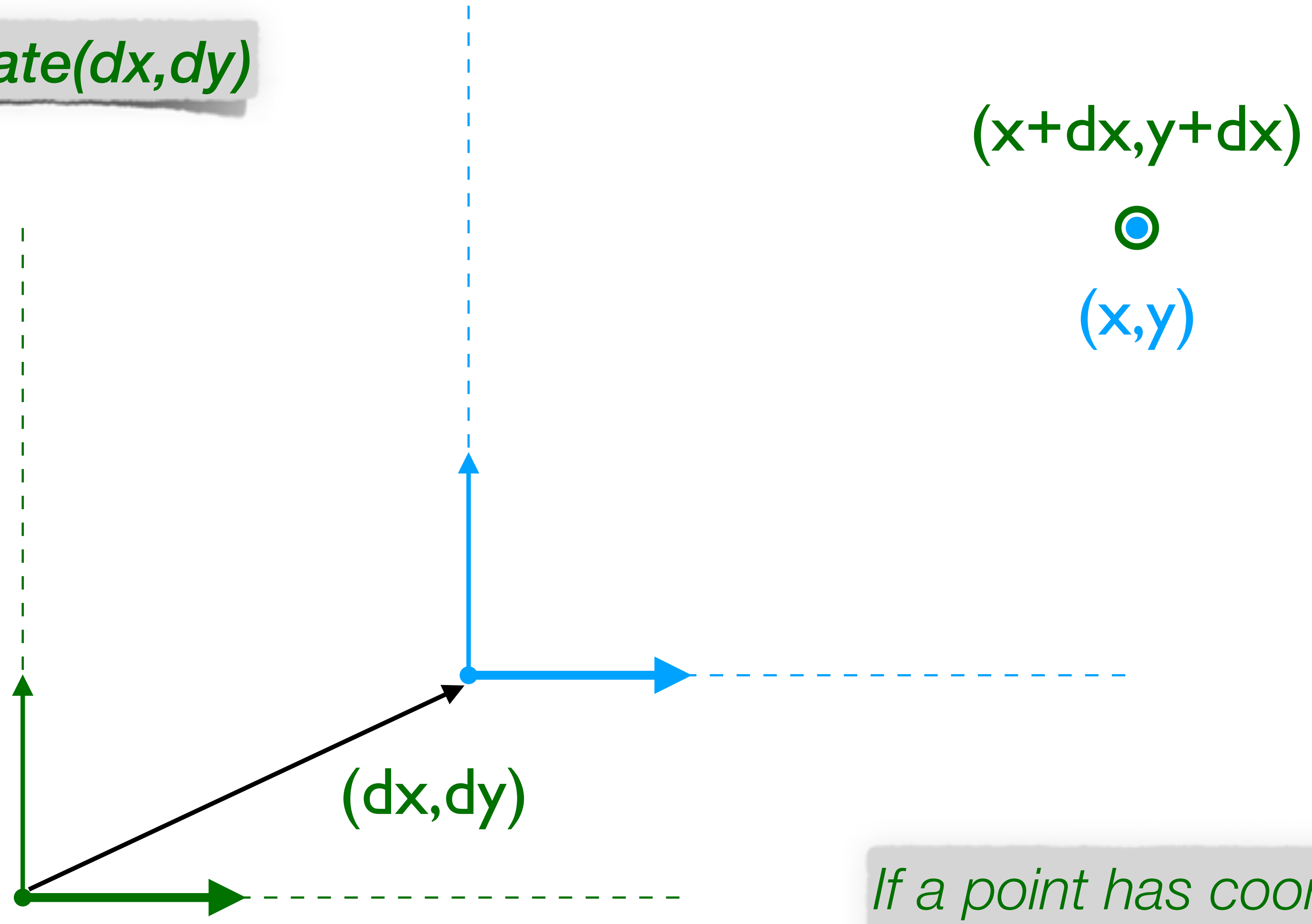
A translation transform leaves
the axis vectors unchanged
(remember: parallel shift doesn't count)
but displaces the origin by the vector (dx,dy)
[that vector defined in the master system]

Notation: ***Translate(dx,dy)***

Green coordinate system is the reference/master
The blue coordinate system is the transformed one

Elementary transforms

Translate(dx,dy)



If a point has coordinates (x,y) in the transformed system, it will have coordinates (x+dx, y+dy) in the original/reference one

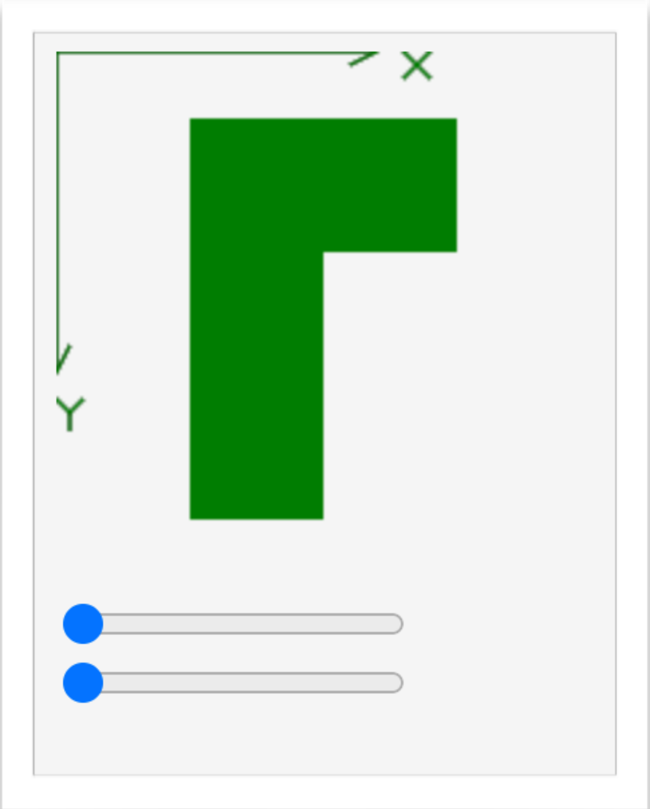
Translations in Canvas

jsbin.com/fesukexori

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>
```



demo.js

```
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;
```

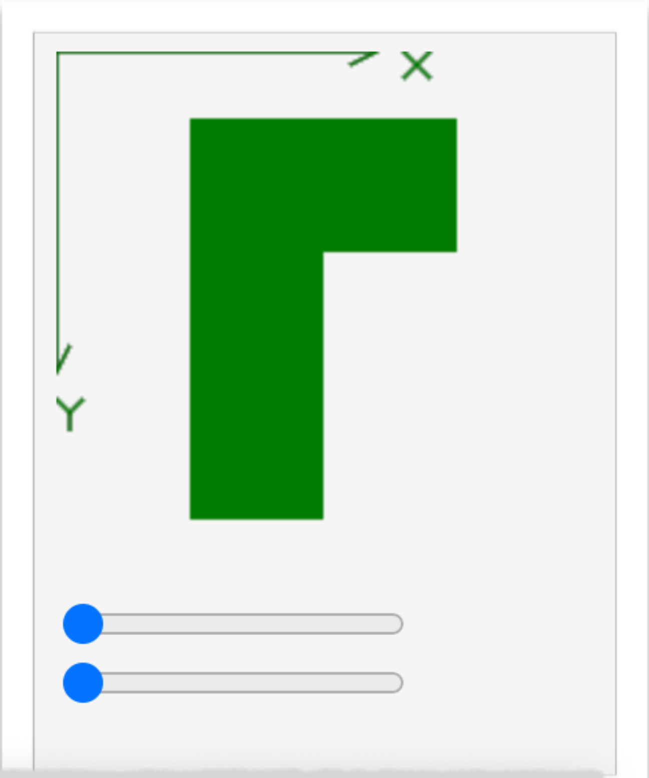
Translations in Canvas

jsbin.com/fesukexori

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)

demo.js

```
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;
```

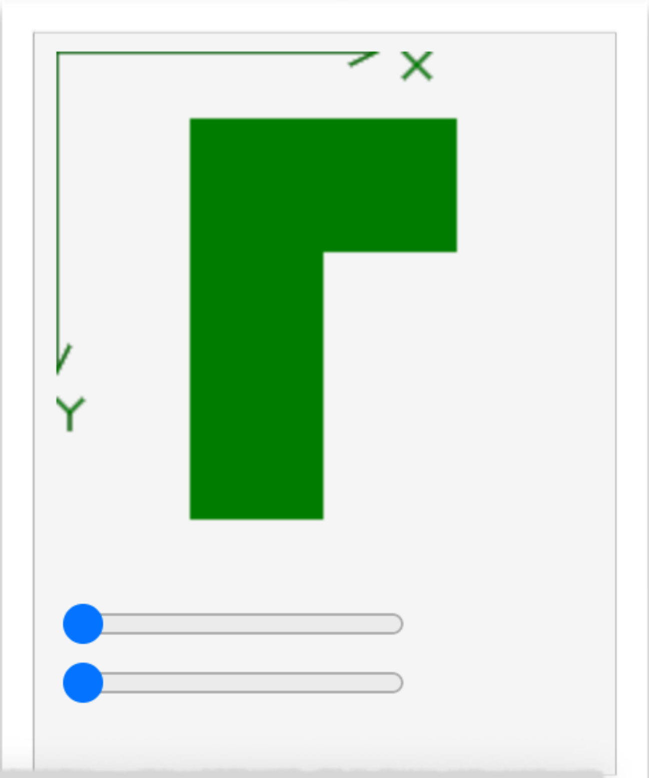
Translations in Canvas

jsbin.com/fesukexori

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 (refer to, tutorial)

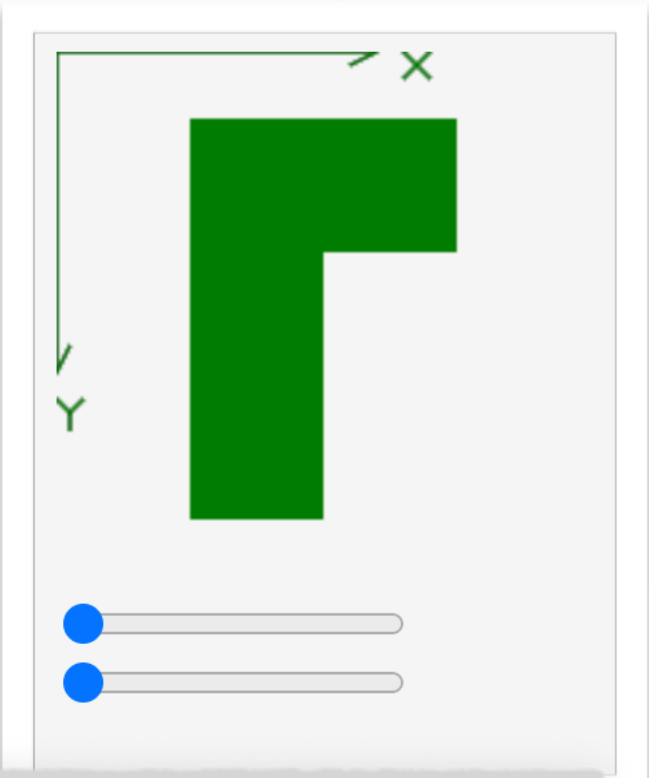
demo.js

```
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;
```

Translations in Canvas

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>
```



Canvas provides implementation of a translation transform!

jsbin.com/fesukexori

Week2/Demo2

demo.js

```
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;
```

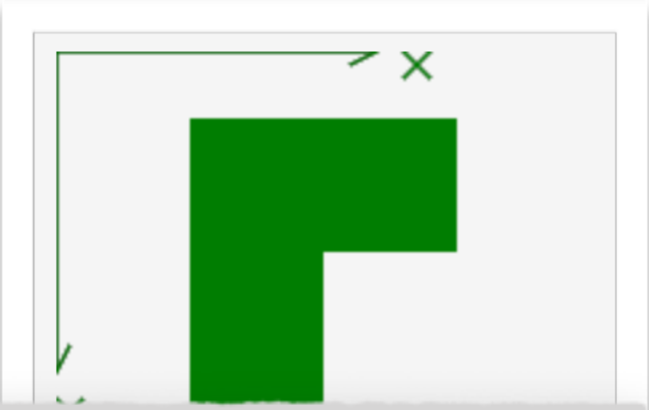
Translations in Canvas

jsbin.com/fesukexori

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>
```

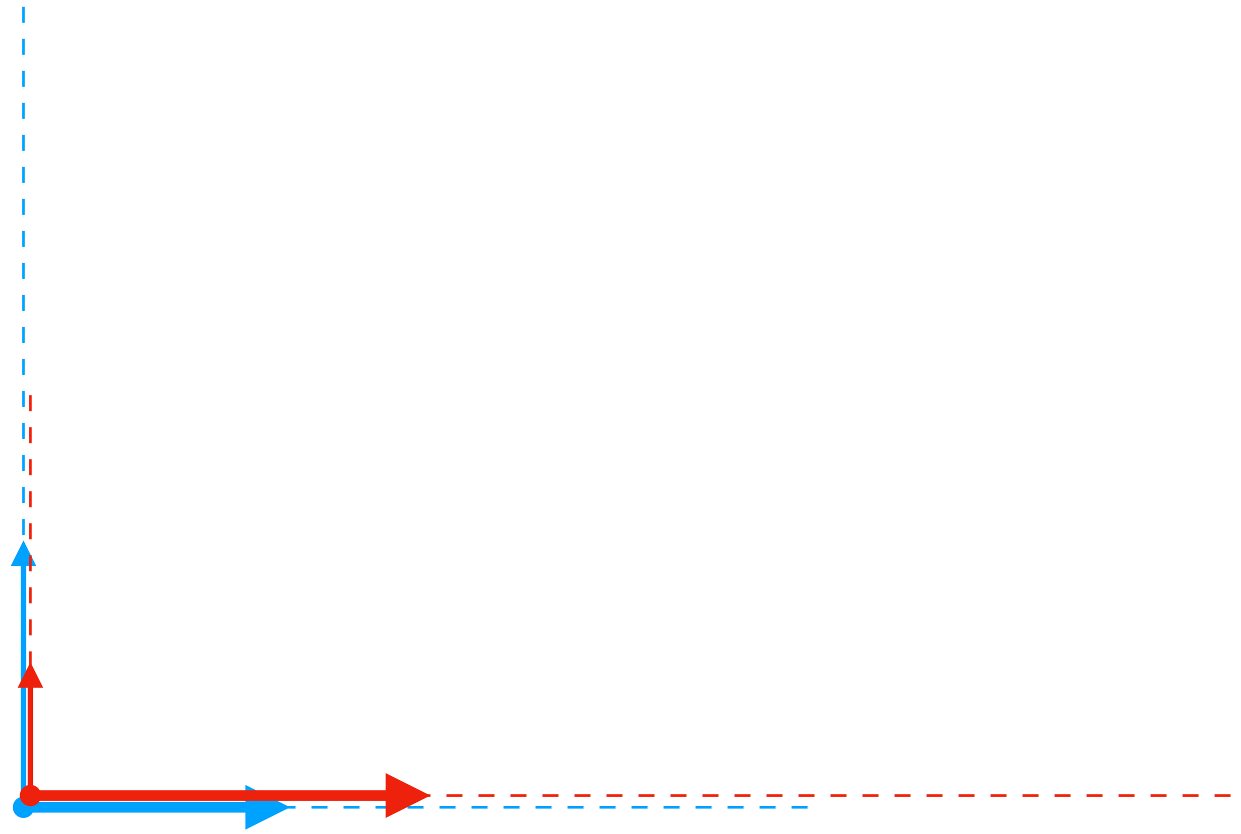


save/restore: consider them as saving the currently applied transform (or none, if no transform applied) and reverting to the same transform later (more in discussion of hierarchical modeling)

demo.js

```
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;
```

Coordinate systems and transforms

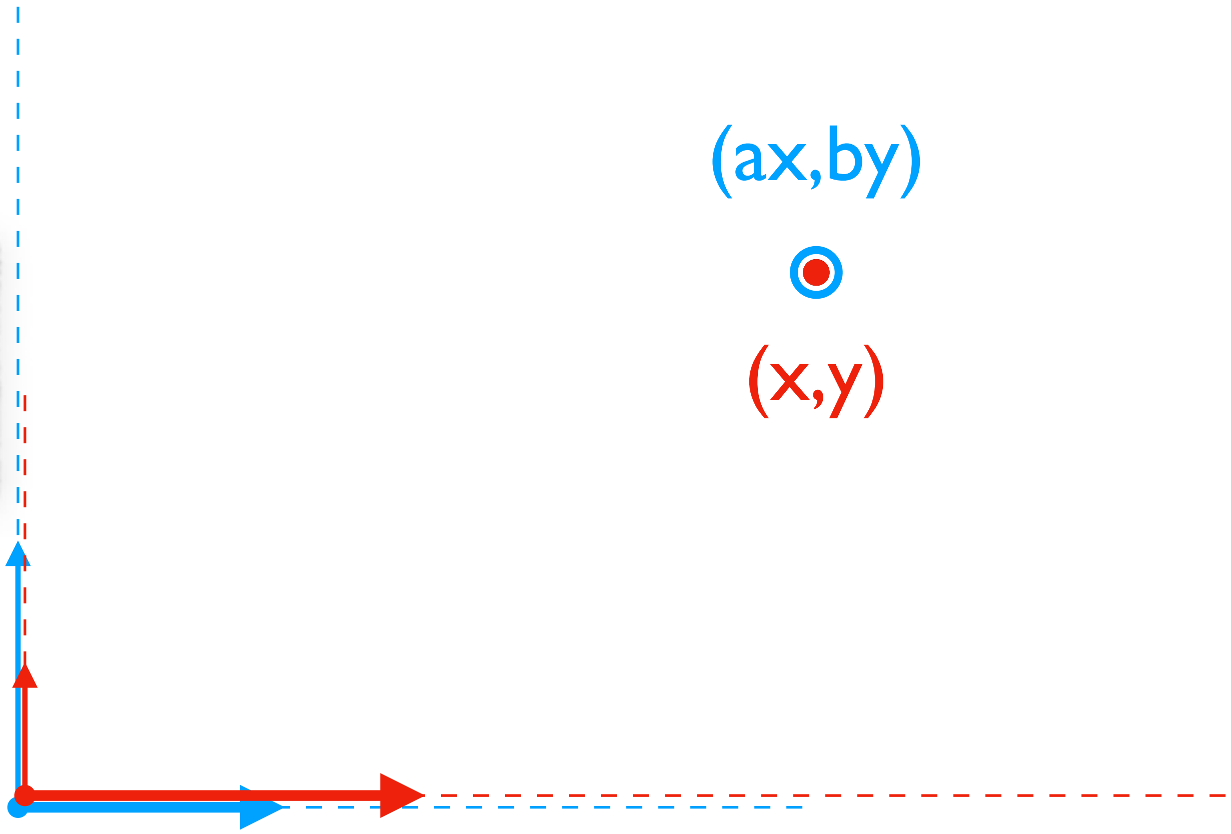


A scaling transformation leaves the origin unchanged, but shrinks or grows the axis vectors by a given factor

Coordinate systems and transforms

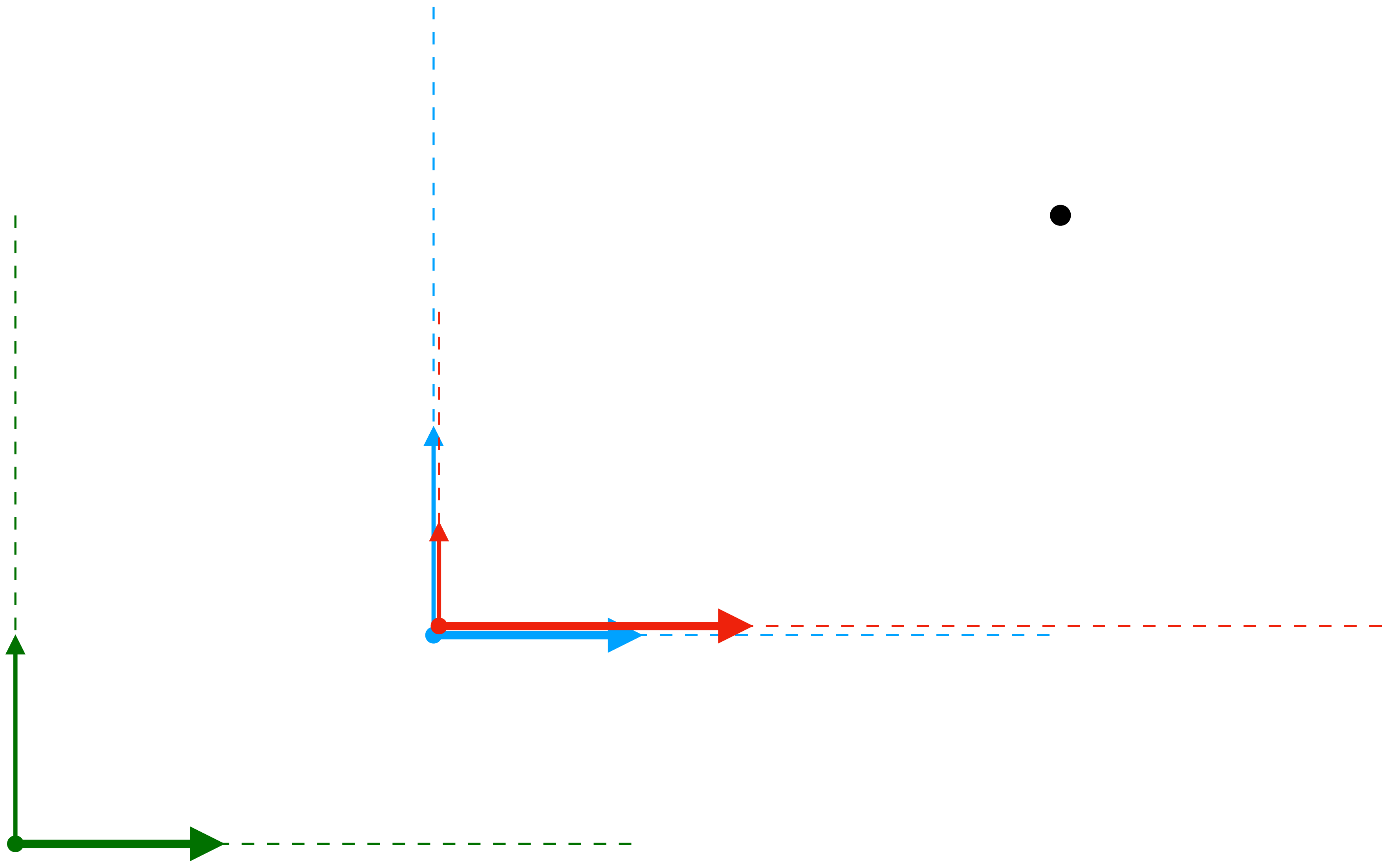
Scale(a,b)

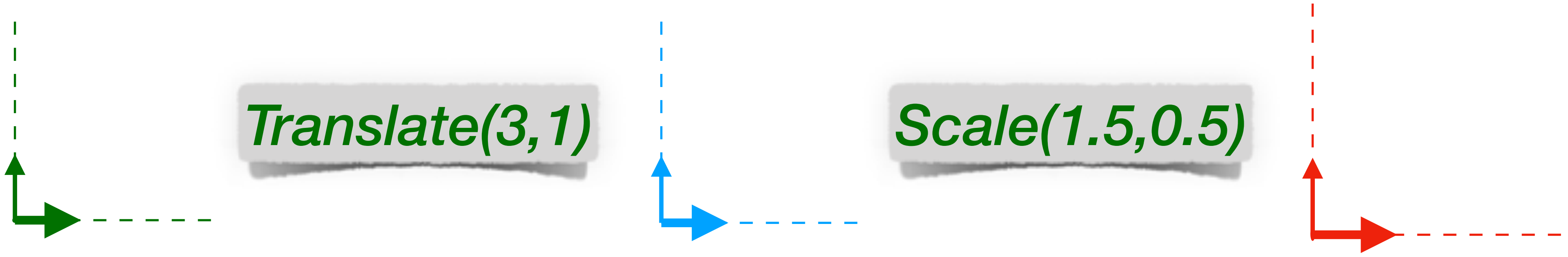
*[in this example **Scale(1.5,0.5)**]*



If a point has coordinates (x,y) in the transformed system, it will have coordinates (ax, by) in the original/reference one

Chaining transforms together

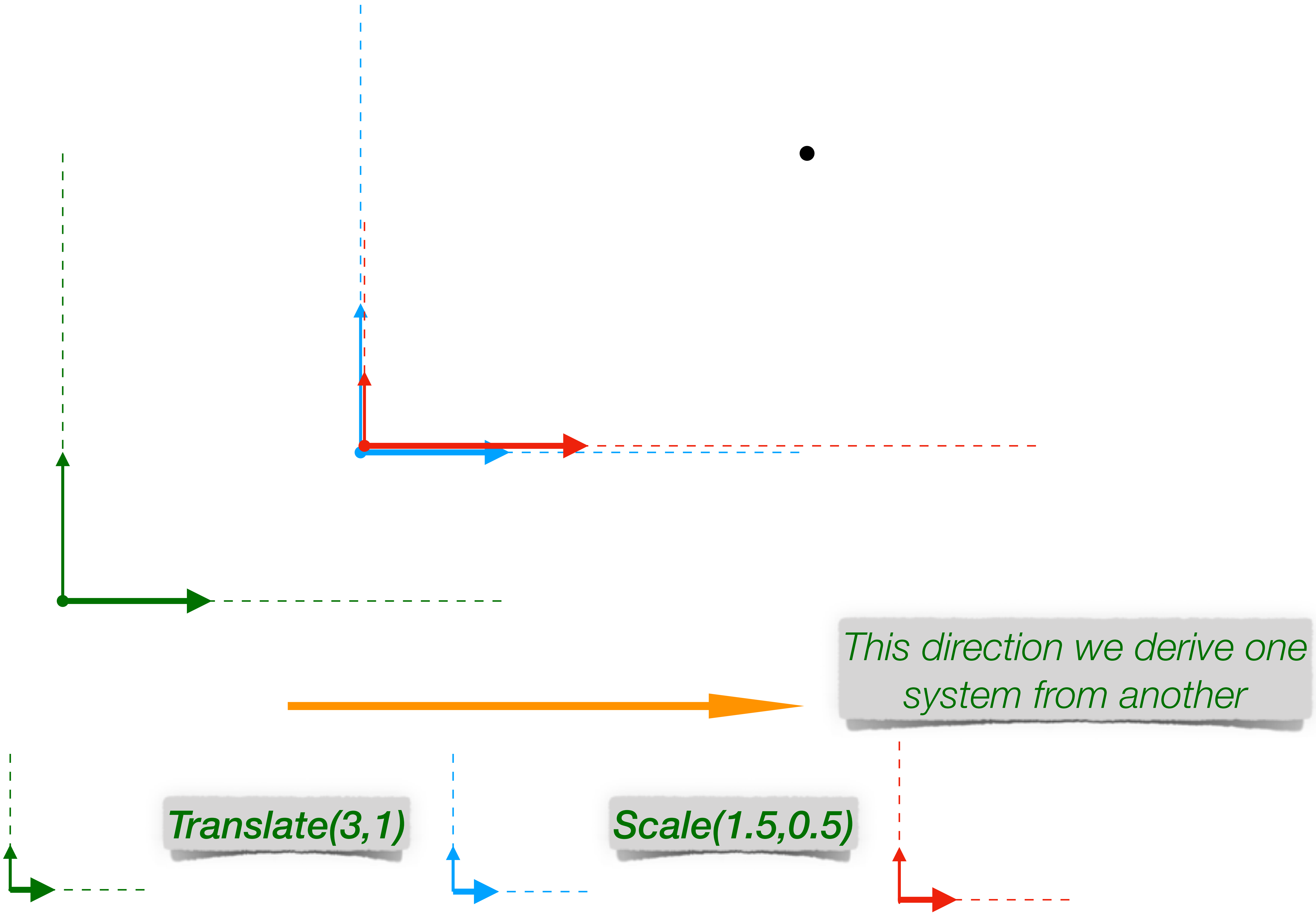




Translate(3,1)

Scale(1.5,0.5)

Chaining transforms together

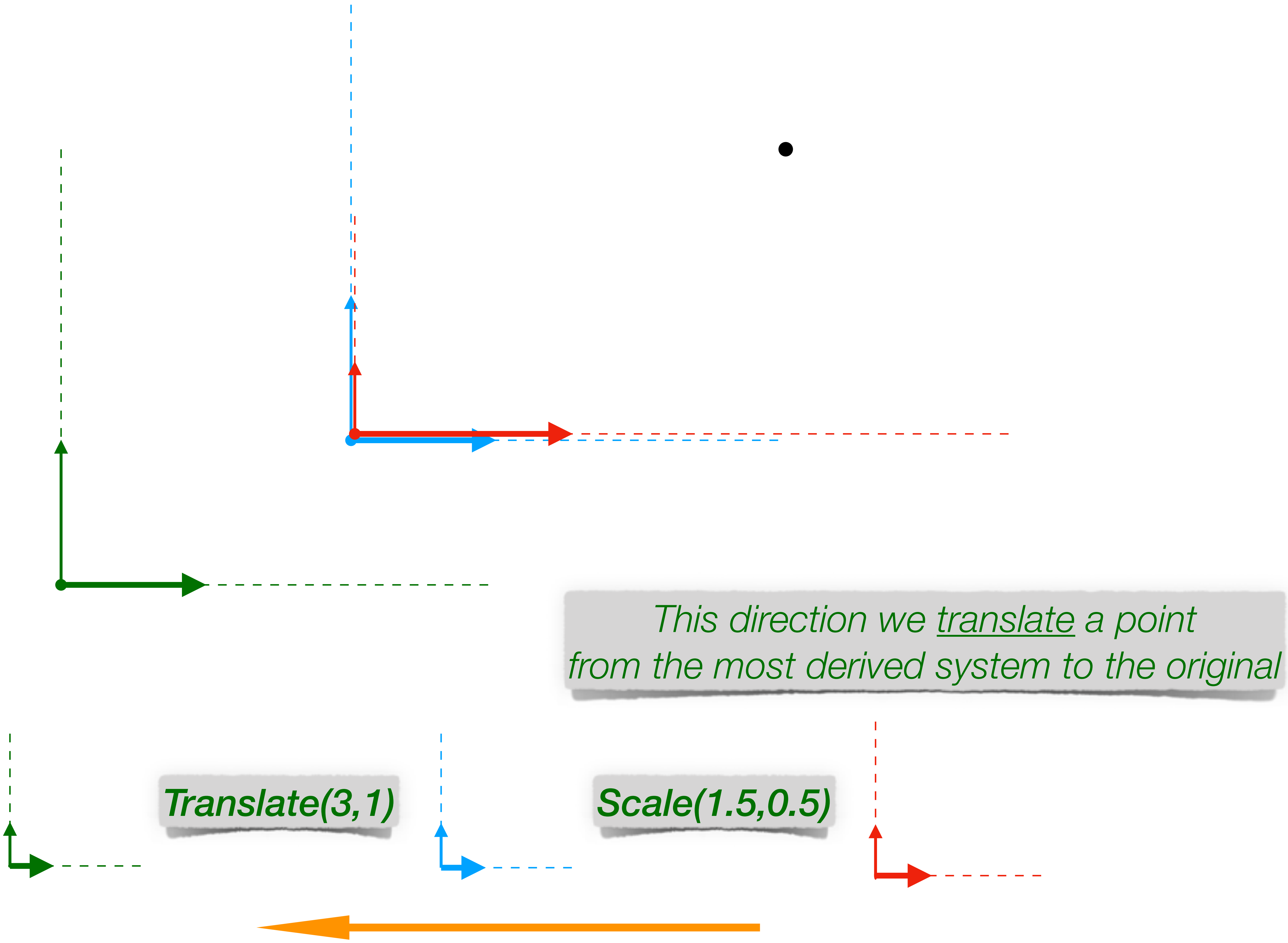


This direction we derive one system from another

Translate(3,1)

Scale(1.5,0.5)

Chaining transforms together

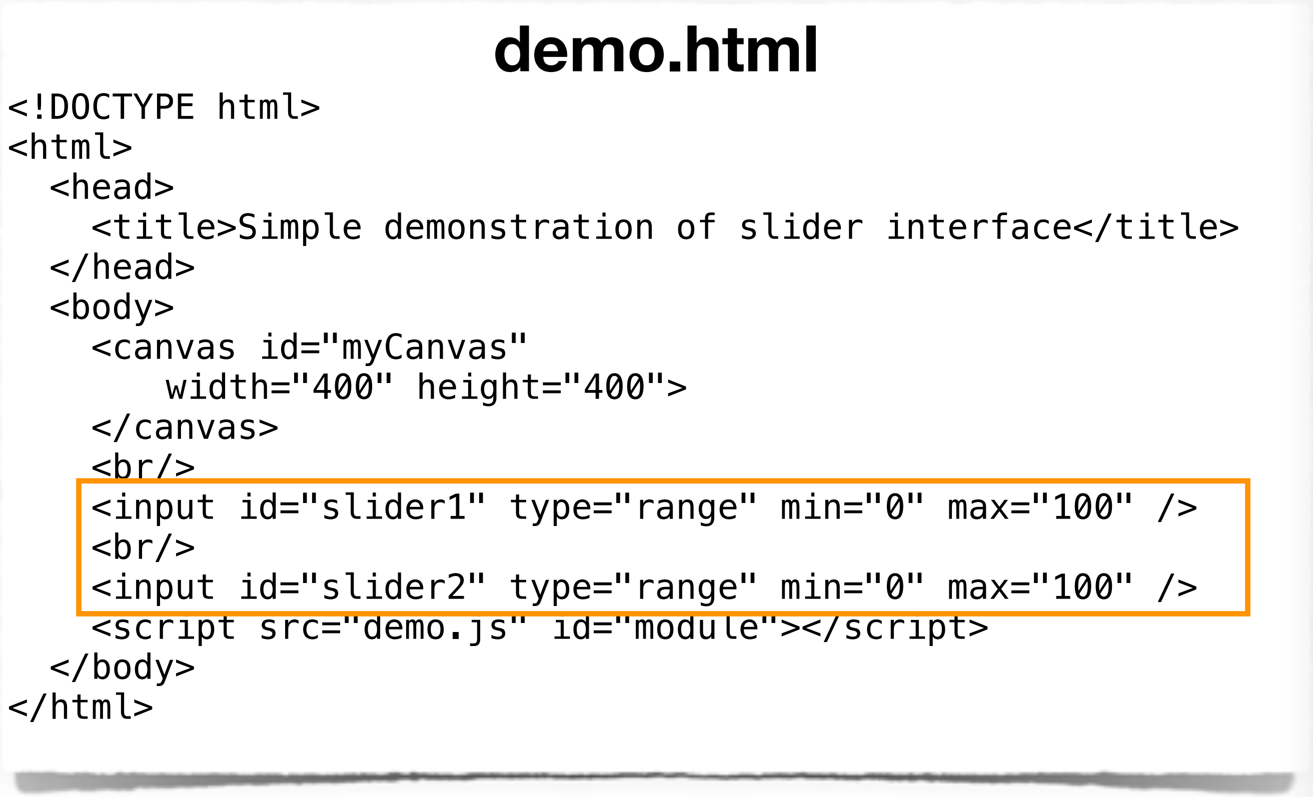


This direction we translate a point from the most derived system to the original

Translate(3,1)

Scale(1.5,0.5)

Translate/Scale in Canvas

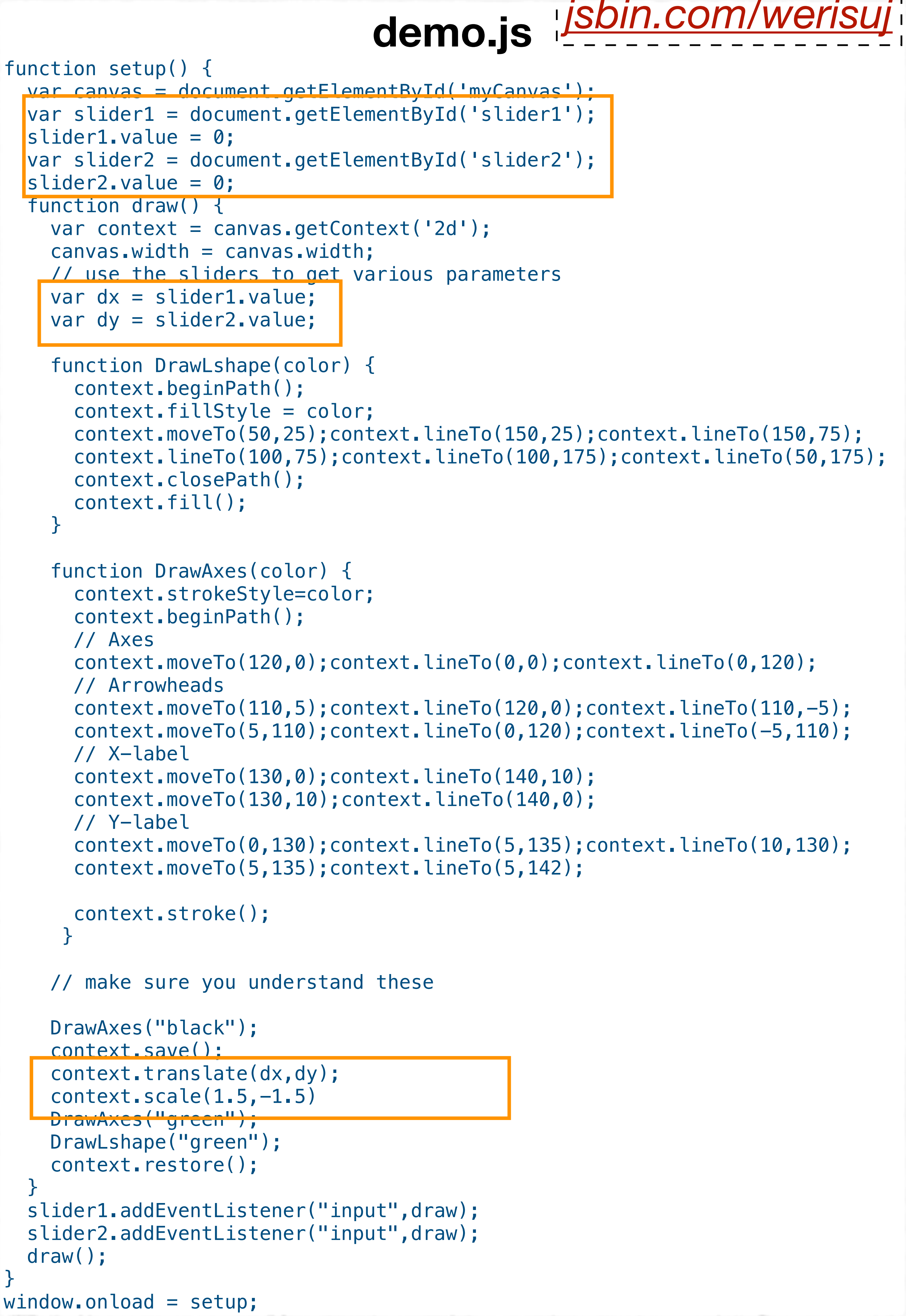


demo.html

```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>
```

Observe that order matters!

jsbin.com/werisuj

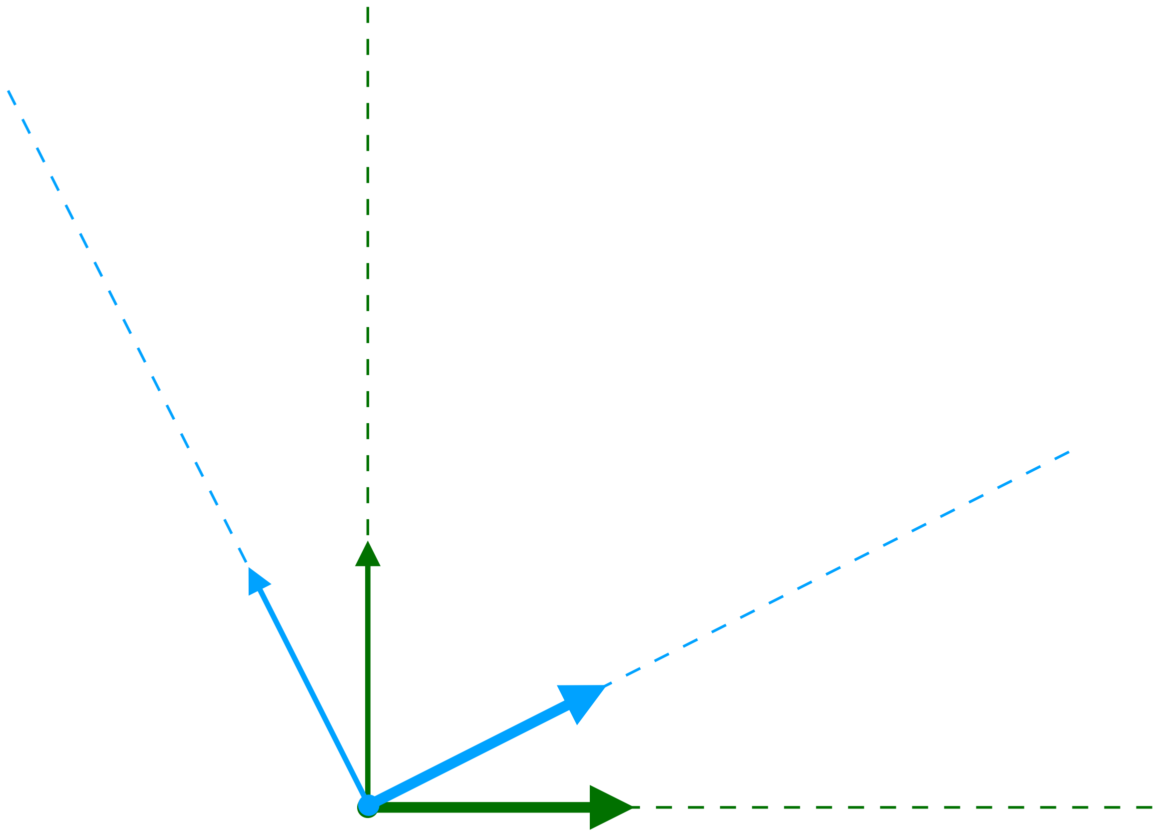


demo.js

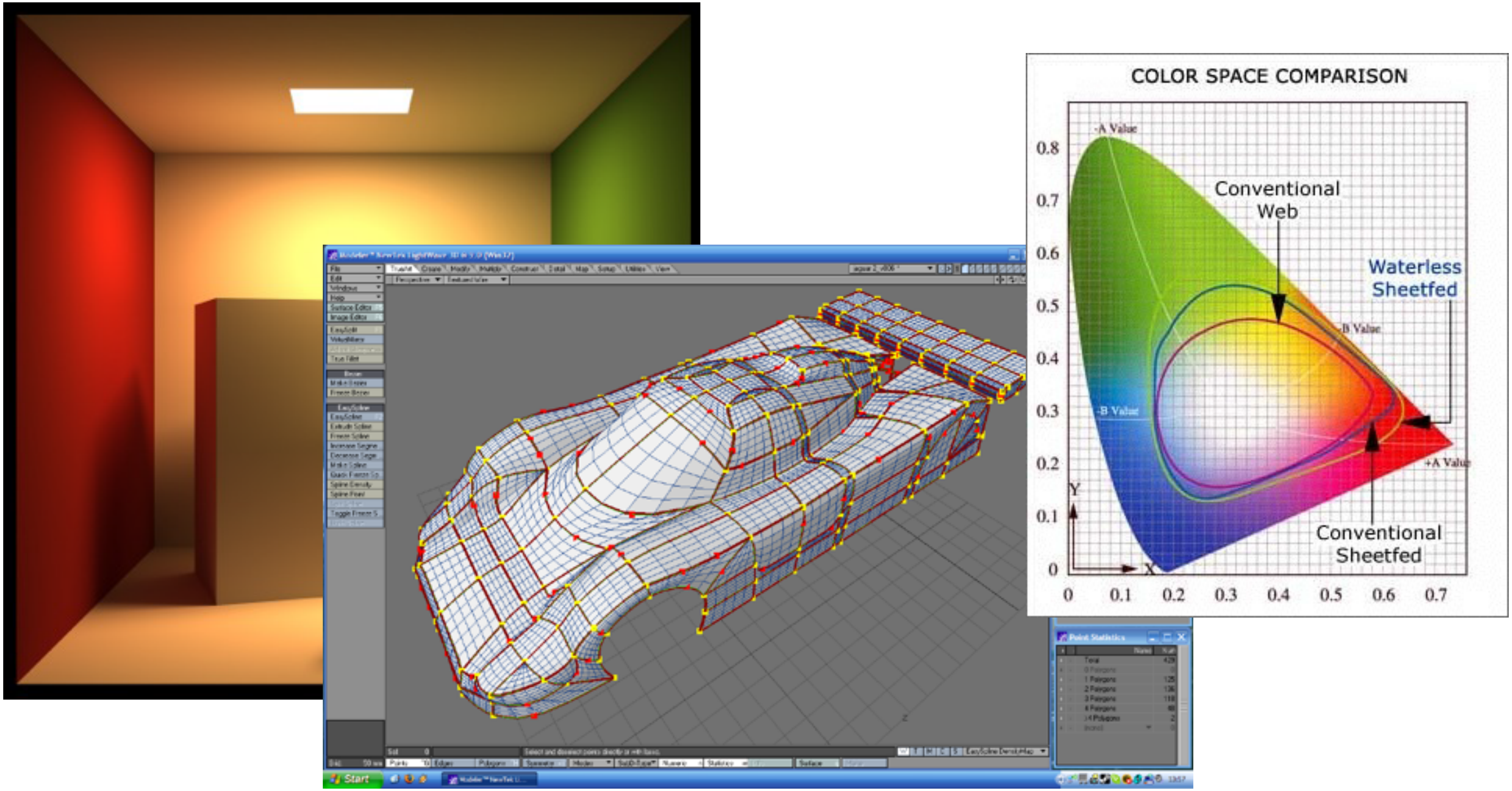
```javascript
function setup() {
 var canvas = document.getElementById('myCanvas');
 var slider1 = document.getElementById('slider1');
 slider1.value = 0;
 var slider2 = document.getElementById('slider2');
 slider2.value = 0;
 function draw() {
  var context = canvas.getContext('2d');
  canvas.width = 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);
  context.scale(1.5,-1.5)
  DrawAxes("green");
  DrawLshape("green");
  context.restore();
 }
 slider1.addEventListener("input",draw);
 slider2.addEventListener("input",draw);
 draw();
}
window.onload = setup;
```

Elementary transforms

jsbin.com/hozeyar



*A rotational transform keeps the origin intact, and rotates the axis vectors around it Implemented by **rotate(angle)** in canvas (use angle in radians!)*



Lecture 4 : Coordinate systems and transforms in 2D (with an introduction to transforms in Canvas)

Tuesday September 21st 2021