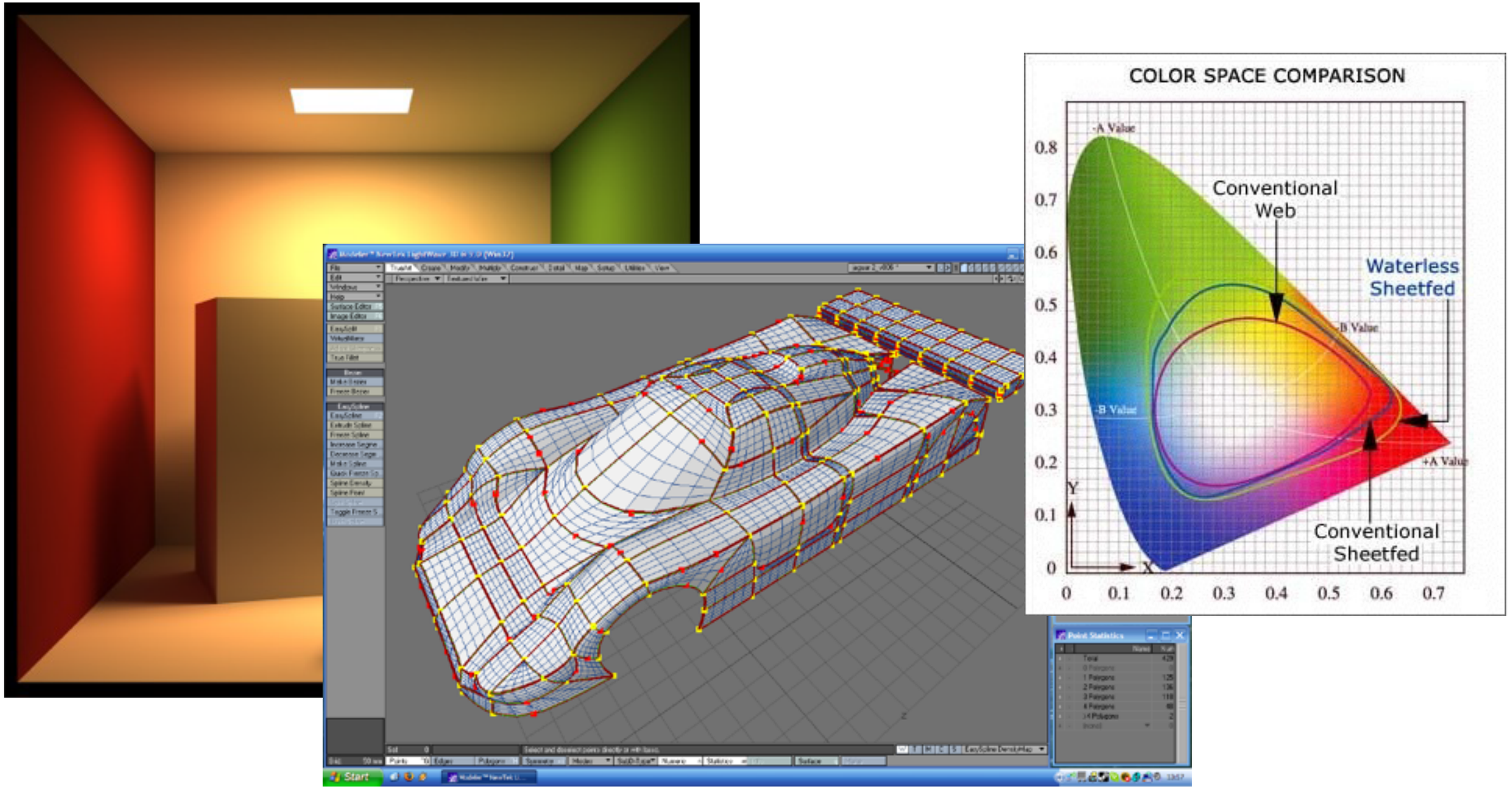


Lecture 5 : Transforms and Hierarchical Modeling in 2D (and overview of the Canvas transform stack)

Thursday September 23rd 2021

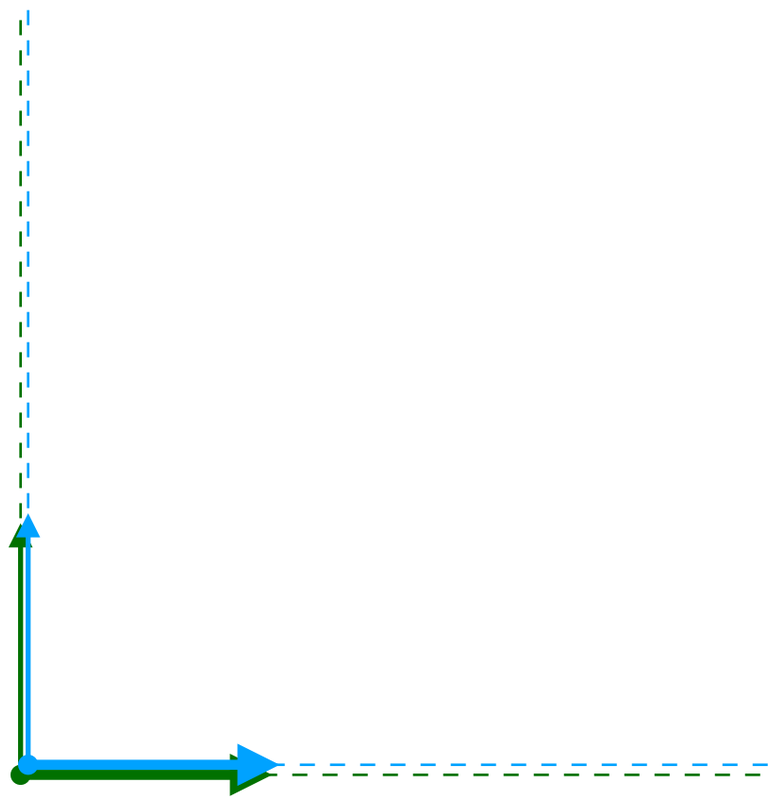
Today's lecture

- Review of elementary transforms (also in Canvas)
- Transform chains and intuition
    - Be mindful of two "competing" interpretations
      (a) The evolution of the current coordinate system, and
      (b) The mapping of drawing points to the original system
- Hierarchical modeling and the Canvas transform/stack
    - We'll see the true function of save()/restore() calls!
- Next week: introduction to the mathematical representation of coordinate transforms

Overview: Elementary Transforms

jsbin.com/hozeyar

Week3/Demo0

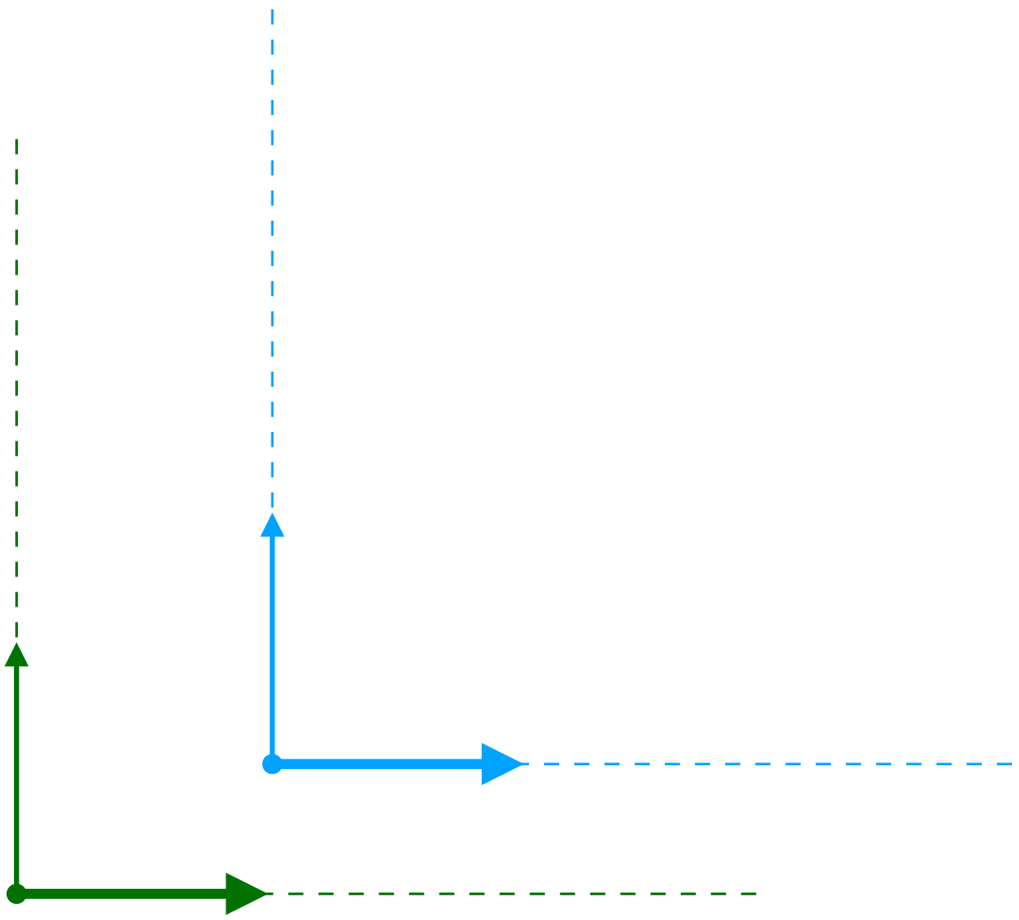


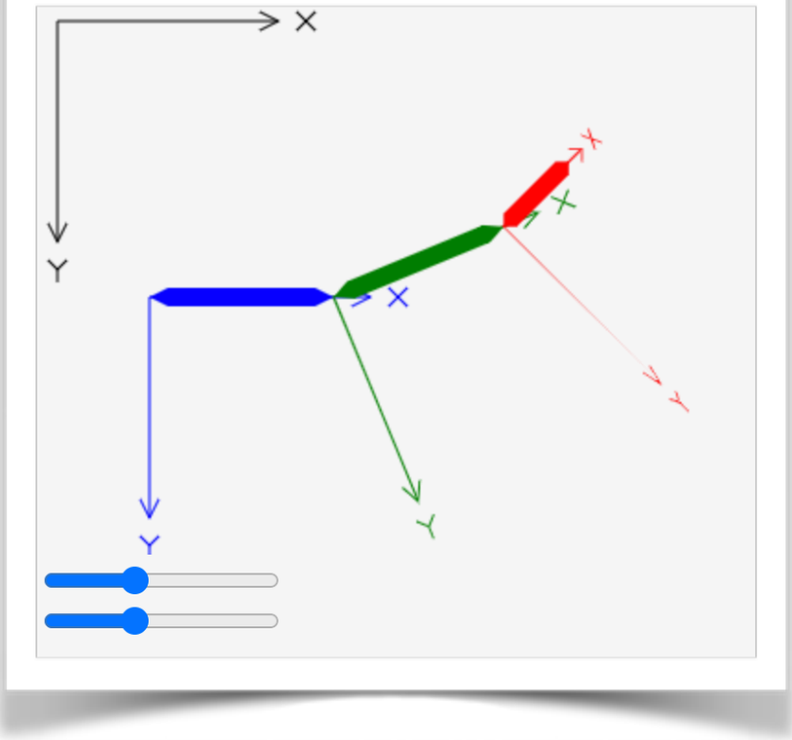
A translational transform leave the axis vectors unchanged, but shifts the origin

Overview: Elementary Transforms

jsbin.com/hozeyar

Week3/Demo0





A translational transform leave the axis vectors unchanged, but shifts the origin

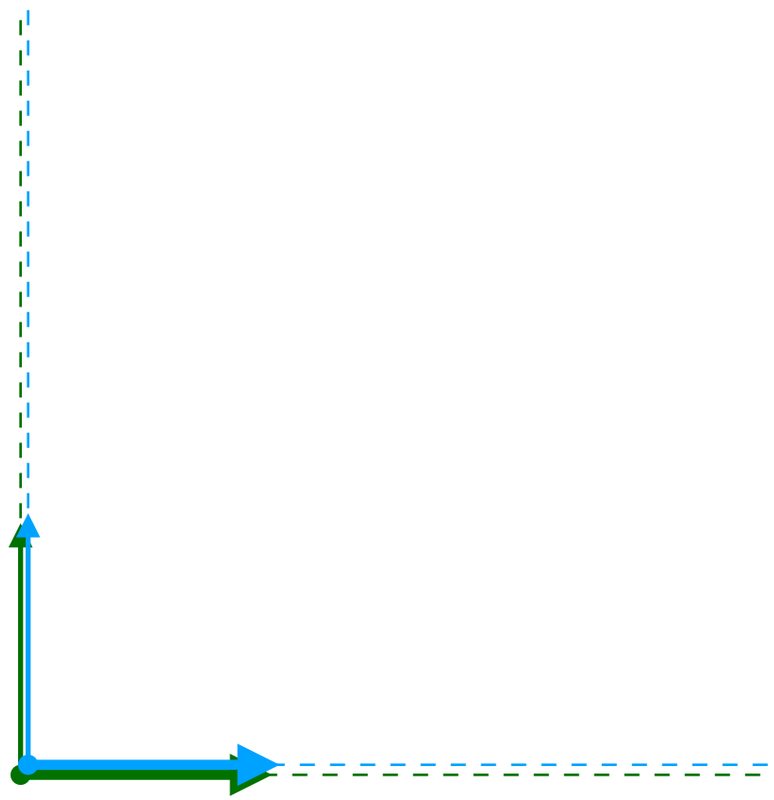
JavaScript

```
[...]

    // make sure you understand these
    axes("black");
    context.translate(50,150);
    linkage("blue");
    context.translate(100,0);
    context.rotate(theta1);
    linkage("green");
    context.translate(100,0);
    context.rotate(phi1);
    context.scale(0.5,1);
    linkage("red");

[...]
```

Overview: Elementary Transforms

jsbin.com/hozeyar

Week3/Demo0

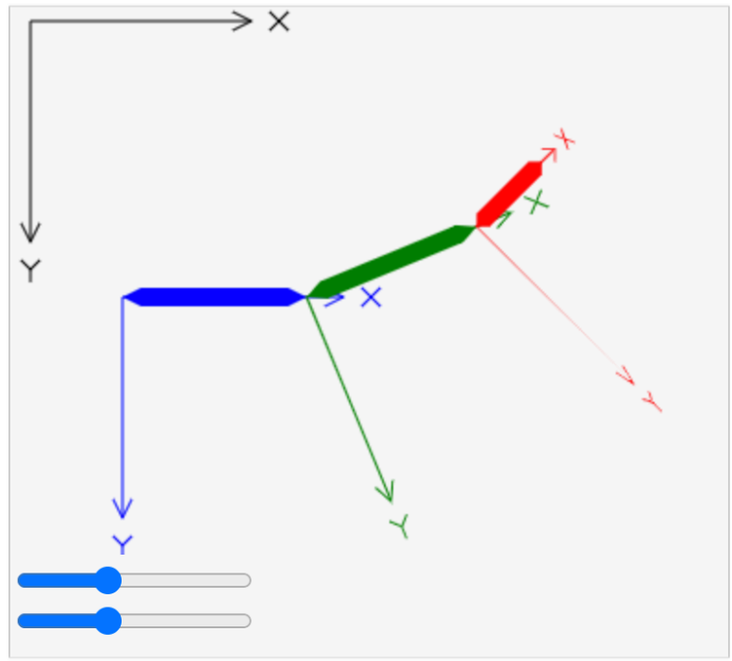


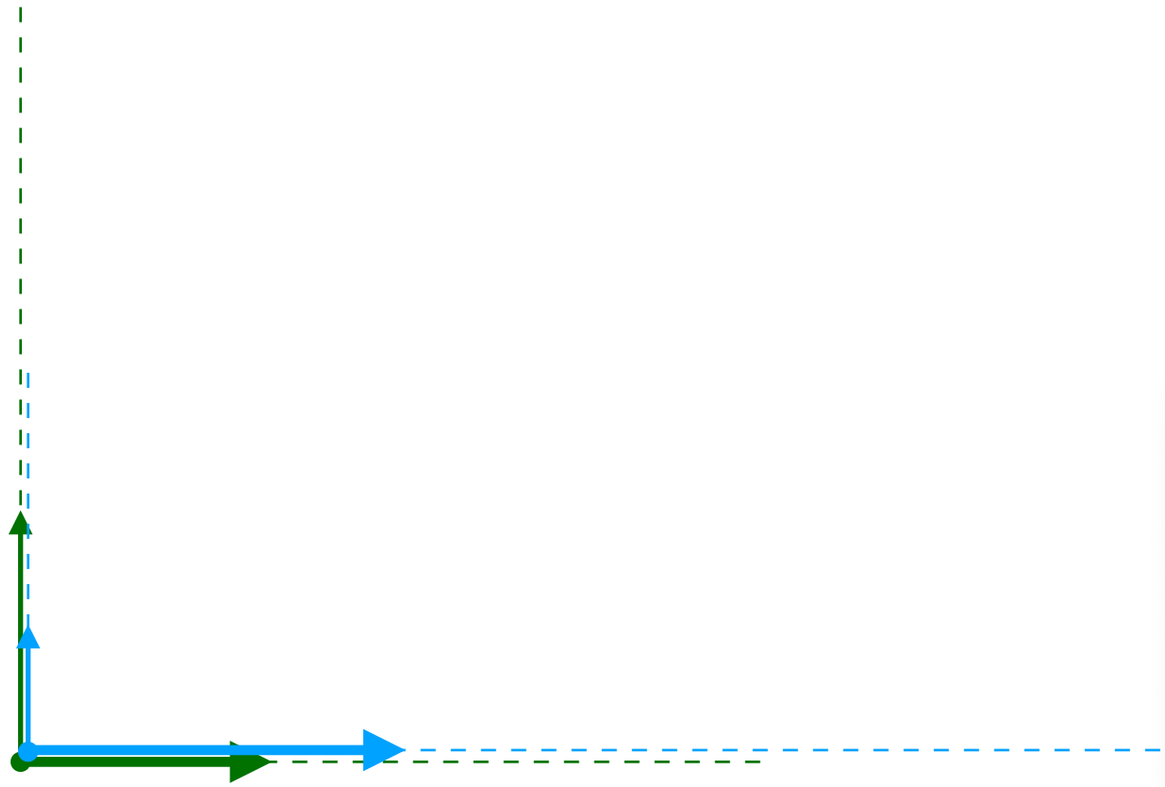
A scaling transform keeps the origin intact, but changes the length of axis vectors

Overview: Elementary Transforms

jsbin.com/hozeyar

Week3/Demo0





A scaling transform keeps the origin intact, but changes the length of axis vectors

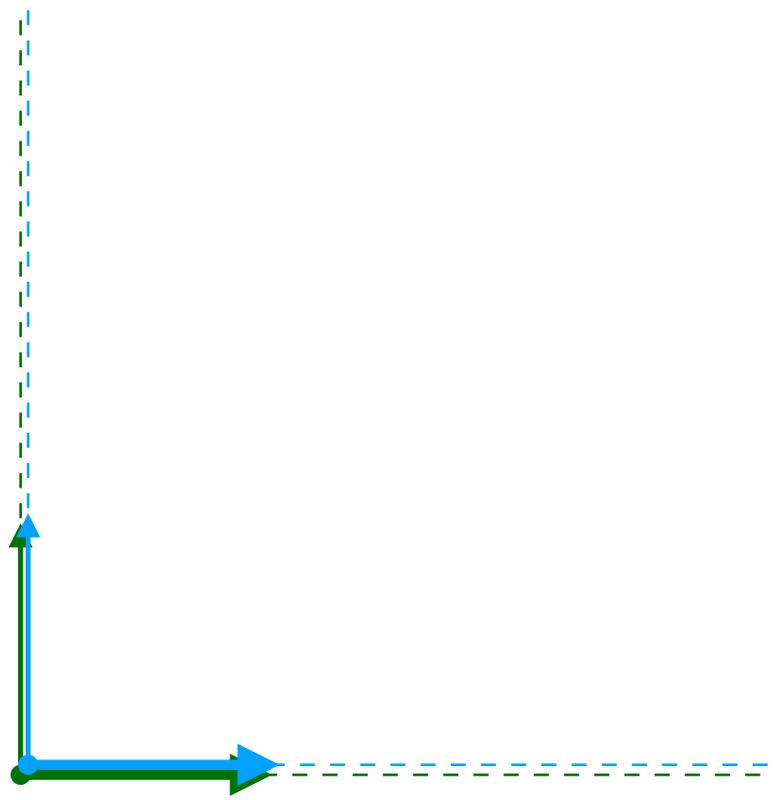
JavaScript

```
[...]

    // make sure you understand these
    axes("black");
    context.translate(50,150);
    linkage("blue");
    context.translate(100,0);
    context.rotate(theta1);
    linkage("green");
    context.translate(100,0);
    context.rotate(phi1);
    context.scale(0.5,1);
    linkage("red");

[...]
```

Overview: Elementary Transforms

jsbin.com/hozeyar

Week3/Demo0

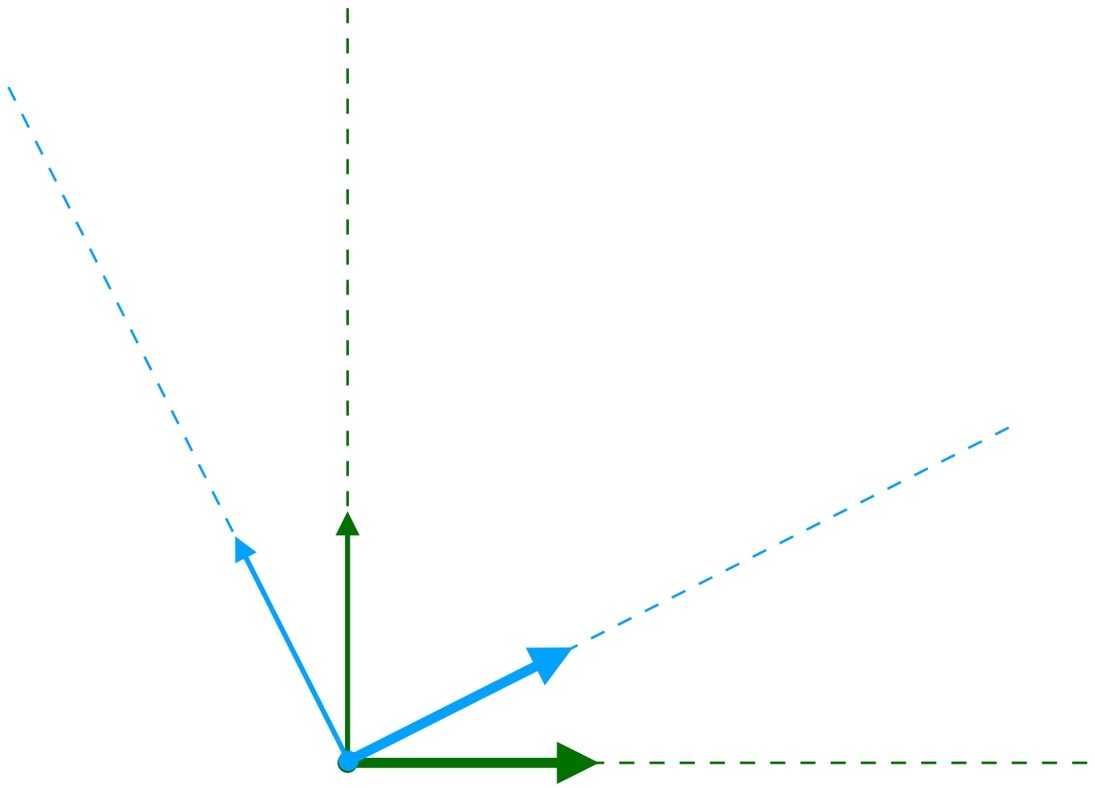


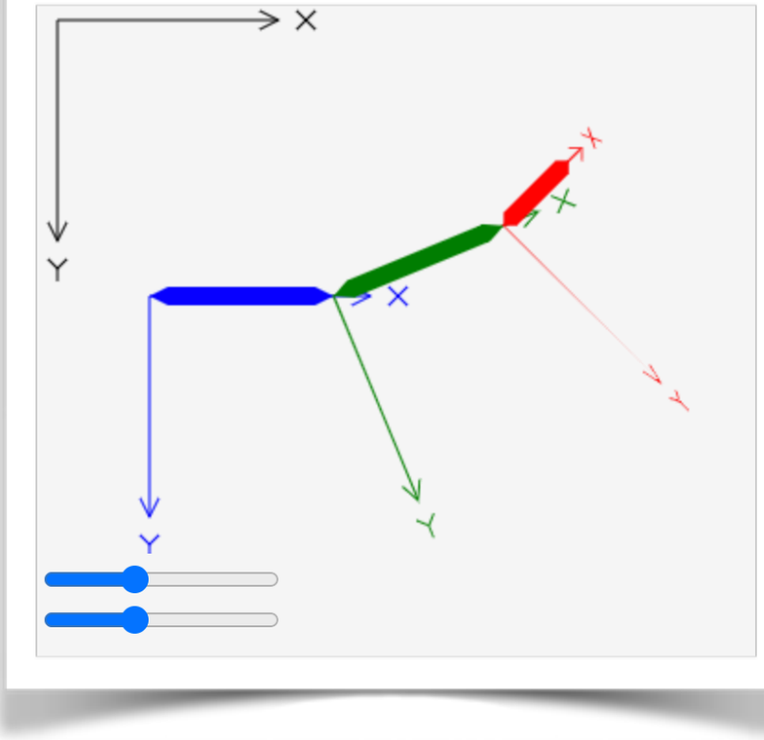
A rotational transform keeps the origin intact, and rotates the axis vectors around it

Overview: Elementary Transforms

jsbin.com/hozeyar

Week3/Demo0





A rotational transform keeps the origin intact, and rotates the axis vectors around it

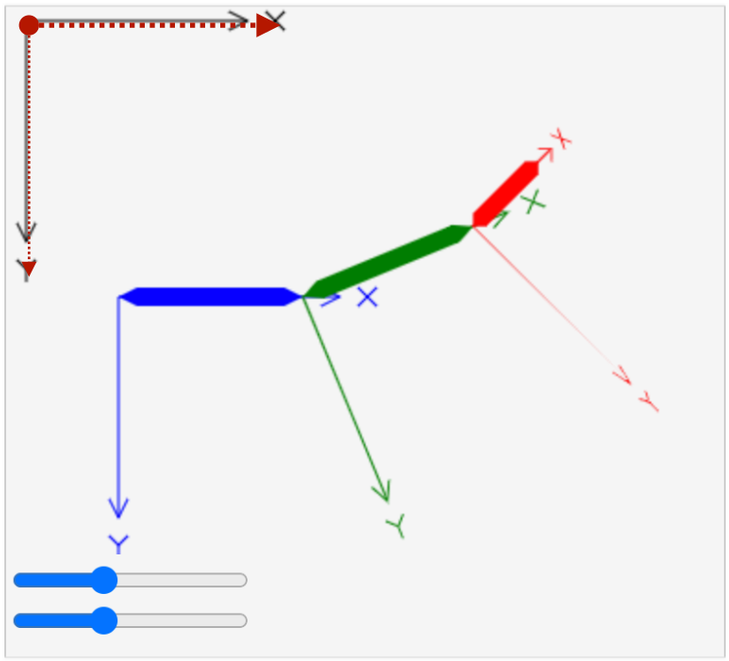
JavaScript

```
[...]

    // make sure you understand these
    axes("black");
    context.translate(50,150);
    linkage("blue");
    context.translate(100,0);
    context.rotate(theta1);
    linkage("green");
    context.translate(100,0);
    context.rotate(phi1);
    context.scale(0.5,1);
    linkage("red");

[...]
```

How are transforms combined?

jsbin.com/hozeyar

Week3/Demo0



JavaScript

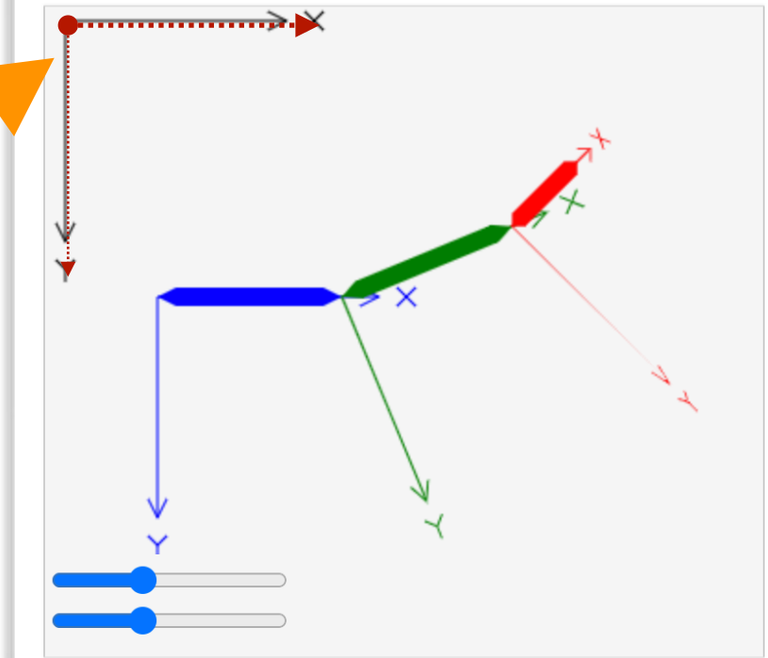
```
[...]

    // make sure you understand these
    axes("black");
    context.translate(50,150);
    linkage("blue");
    context.translate(100,0);
    context.rotate(theta1);
    linkage("green");
    context.translate(100,0);
    context.rotate(phi1);
    context.scale(0.5,1);
    linkage("red");

[...]
```

Canvas maintains a representation of the transform that relates the current coordinate system to the canvas coordinate system, aligned with the top-left corner

How are transforms combined?

jsbin.com/hozeyar

Week3/Demo0



JavaScript

```
[...]

    // make sure you understand these
    axes("black");
    context.translate(50,150);
    linkage("blue");
    context.translate(100,0);
    context.rotate(theta1);
    linkage("green");
    context.translate(100,0);
    context.rotate(phi1);
    context.scale(0.5,1);
    linkage("red");

[...]
```

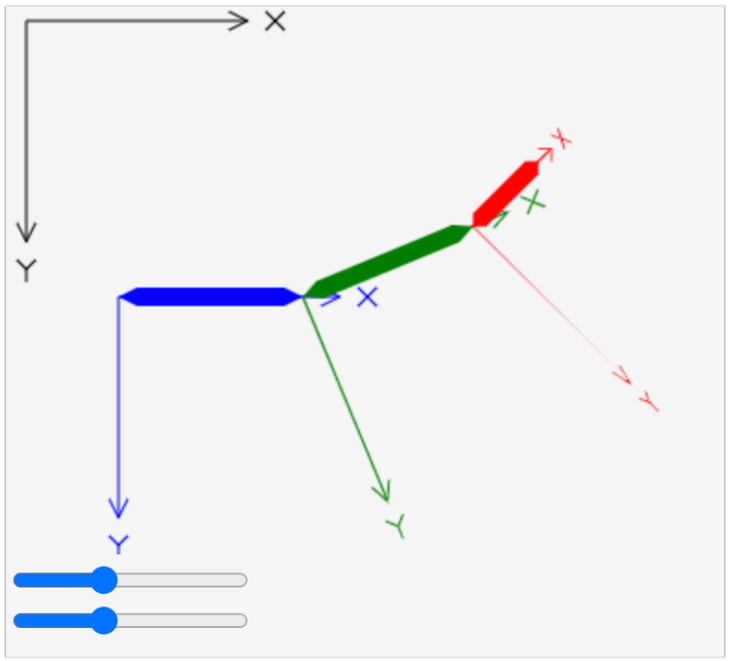
(no transforms yet!)

In the absence of any transform commands issued to the canvas drawing context, the original (canvas) system is the active one. We'll see how this evolves with transforms being issued … (the brown axes indicate the "current" system)

How are transforms combined?

jsbin.com/hozeyar

Week3/Demo0



JavaScript

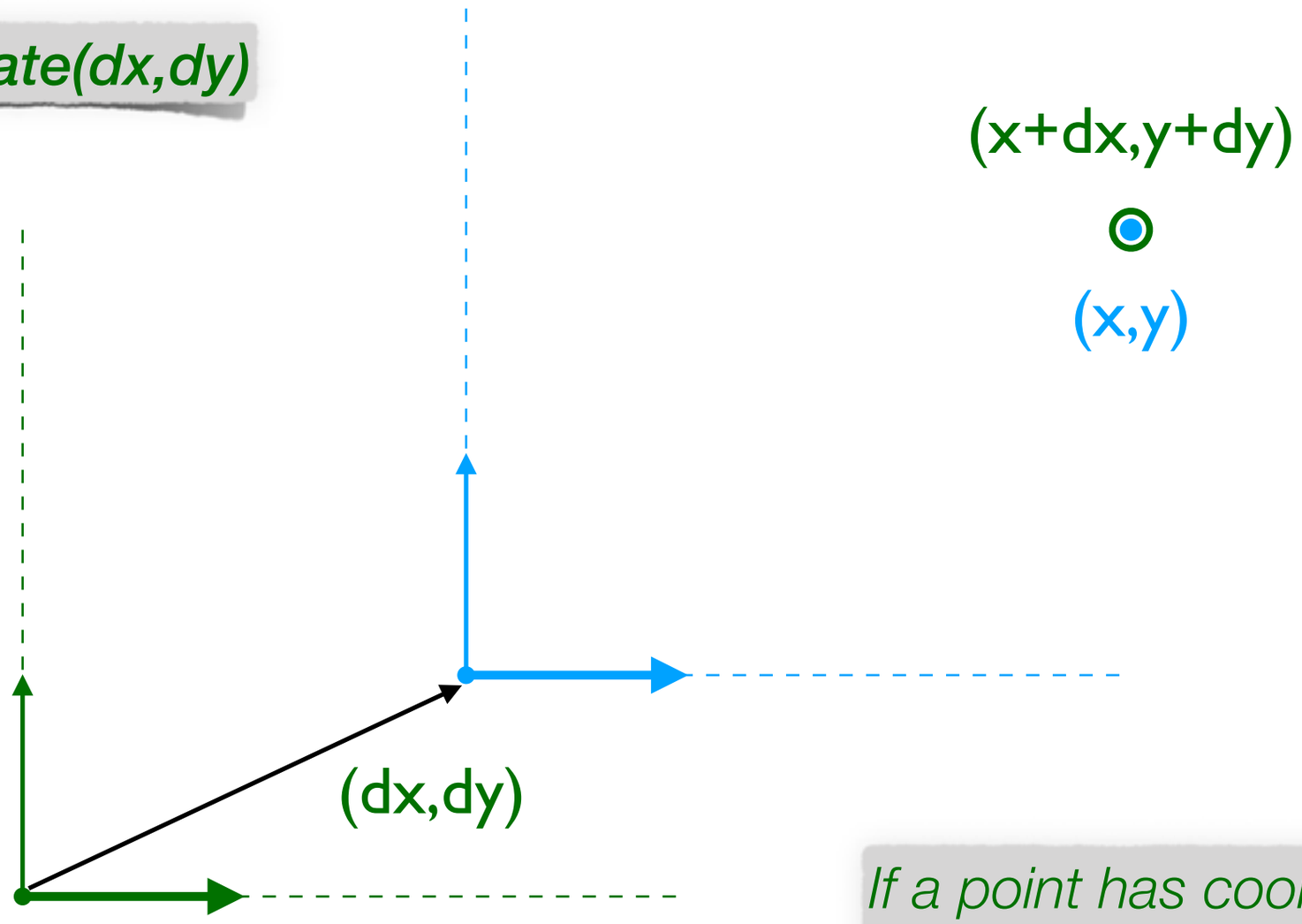
```
[...]

    // make sure you understand these
    axes("black");
    context.translate(50,150);
    linkage("blue");
    context.translate(100,0);
    context.rotate(theta1);
    linkage("green");
    context.translate(100,0);
    context.rotate(phi1);
    context.scale(0.5,1);
    linkage("red");

[...]
```

Transforms are introduced in this order
(current coordinate system is obtained by combining transforms left-to-right)

Points drawn in the current coordinate system are transformed back to canvas coordinates by applying the transforms in this order (right-to-left)

Remember: Transform semantics

Translate(dx,dy)



(x+dx,y+dy)

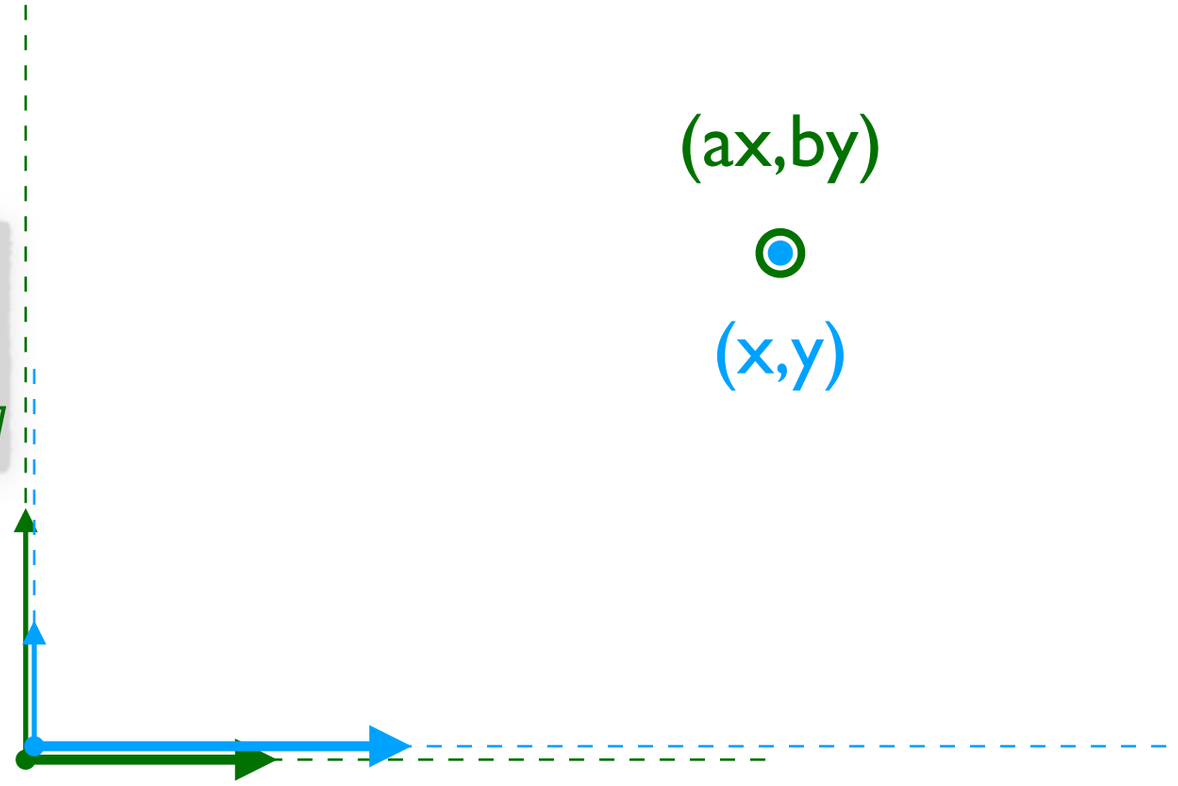
(x,y)

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

Remember: Transform semantics

Scale(a,b)

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



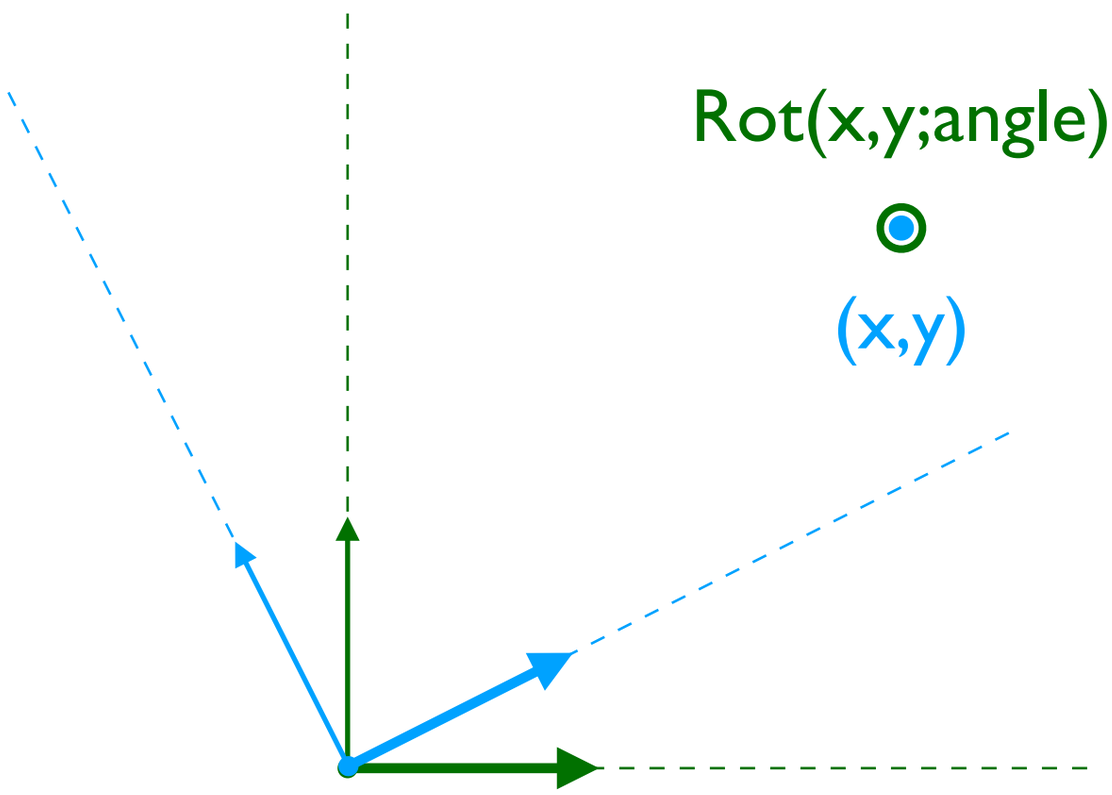
If a point has coordinates (x,y) in the transformed system, it will have coordinates (ax, by) in the original/reference one

Remember: Transform semantics

jsbin.com/hozeyar

Week3/Demo0

Rotate(angle)

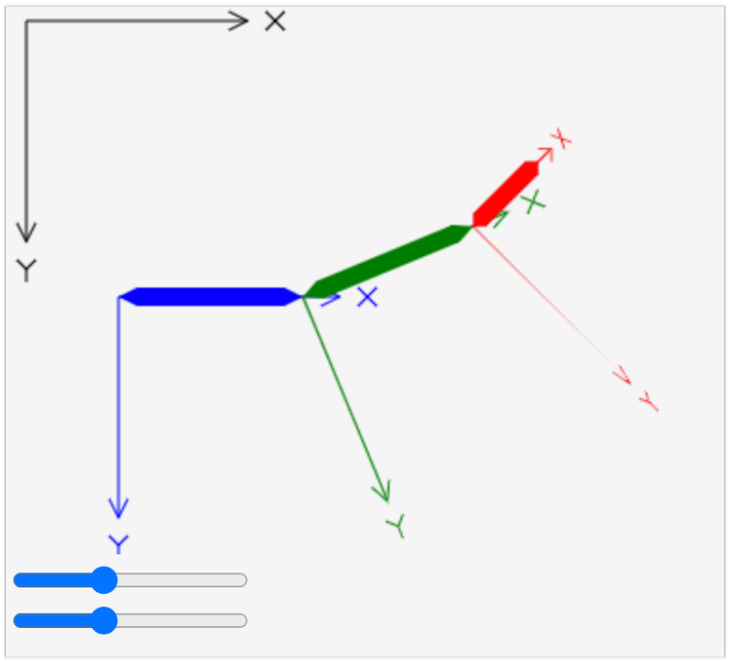


The semantics of rotations are a tiny bit more complex; we'll see them in algebraic terms later (next week …)

How are transforms combined?

jsbin.com/hozeyar

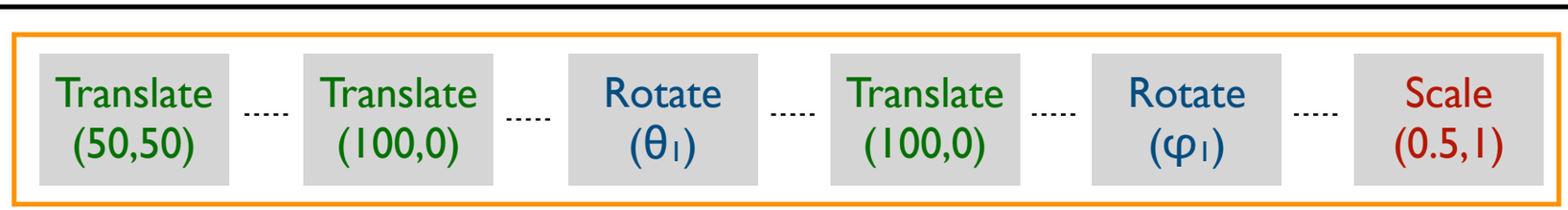
Week3/Demo0



JavaScript

```
[...]

    // make sure you understand these
    axes("black");
    context.translate(50,150);
    linkage("blue");
    context.translate(100,0);
    context.rotate(theta1);
    linkage("green");
    context.translate(100,0);
    context.rotate(phi1);
    context.scale(0.5,1);
    linkage("red");

[...]
```

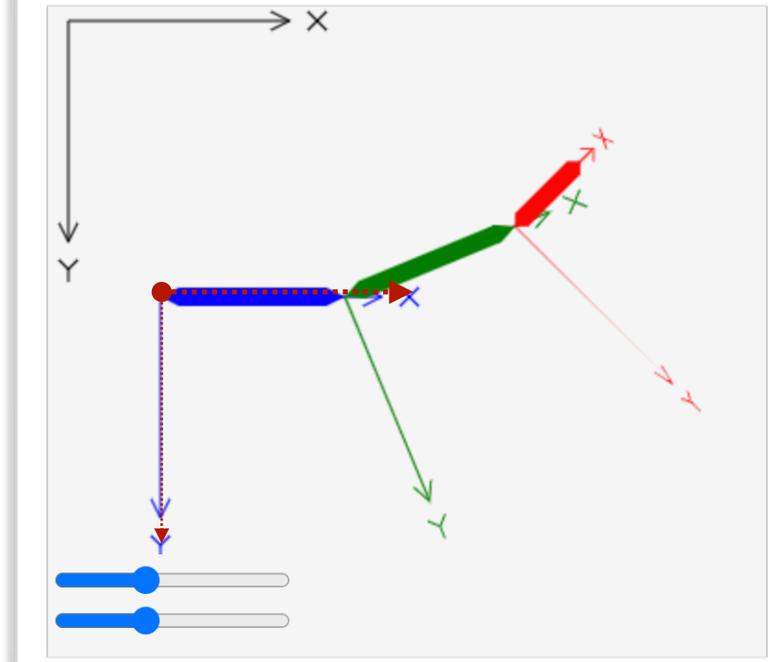
After transform commands have been issued, you can intuitively think of the context retaining a representation of the combined chain of them all (not exactly how it works … but an ok intuition for now)



How are transforms combined?

jsbin.com/hozeyar

Week3/Demo0



JavaScript

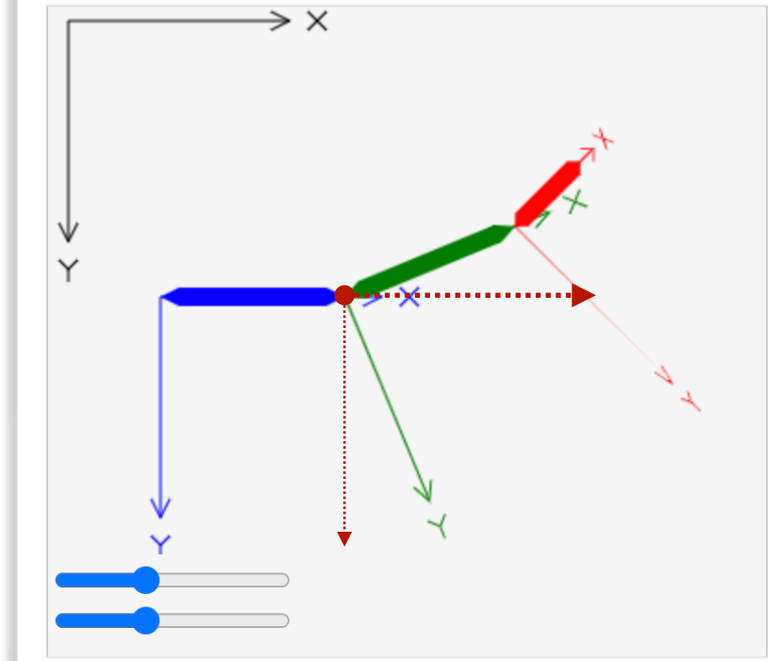
```
[...]

    // make sure you understand these
    axes("black");
    context.translate(50,150);
    linkage("blue");
    context.translate(100,0);
    context.rotate(theta1);
    linkage("green");
    context.translate(100,0);
    context.rotate(phi1);
    context.scale(0.5,1);
    linkage("red");

[...]
```

Translate
(50,50)

How are transforms combined?

jsbin.com/hozeyar

Week3/Demo0



JavaScript

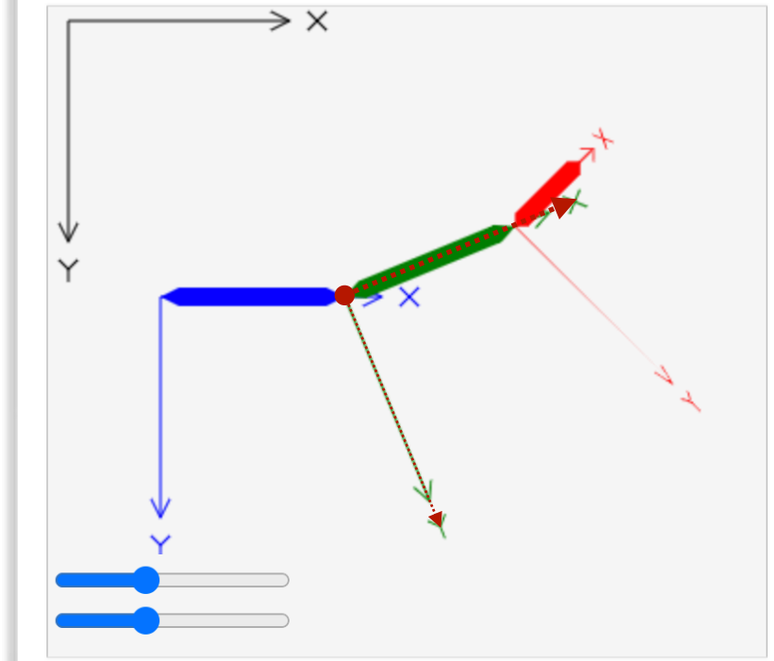
```
[...]

    // make sure you understand these
    axes("black");
    context.translate(50,150);
    linkage("blue");
    context.translate(100,0);
    context.rotate(theta1);
    linkage("green");
    context.translate(100,0);
    context.rotate(phi1);
    context.scale(0.5,1);
    linkage("red");

[...]
```

Translate (50,50) ..... Translate (100,0)

How are transforms combined?

jsbin.com/hozeyar

Week3/Demo0



JavaScript

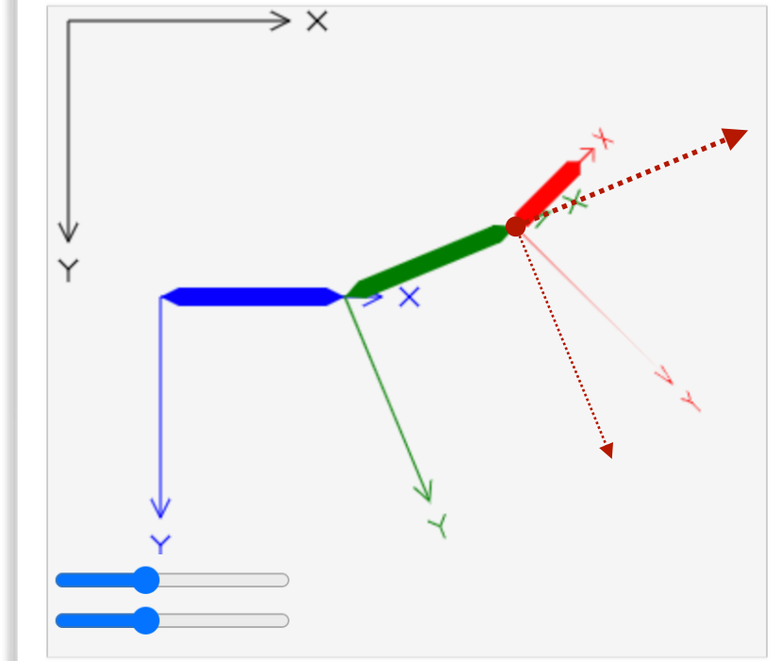
```
[...]

    // make sure you understand these
    axes("black");
    context.translate(50,150);
    linkage("blue");
    context.translate(100,0);
    context.rotate(theta1);
    linkage("green");
    context.translate(100,0);
    context.rotate(phi1);
    context.scale(0.5,1);
    linkage("red");

[...]
```

Translate (50,50) ----- Translate (100,0) ----- Rotate (θ_1)

How are transforms combined?

jsbin.com/hozeyar

Week3/Demo0



JavaScript

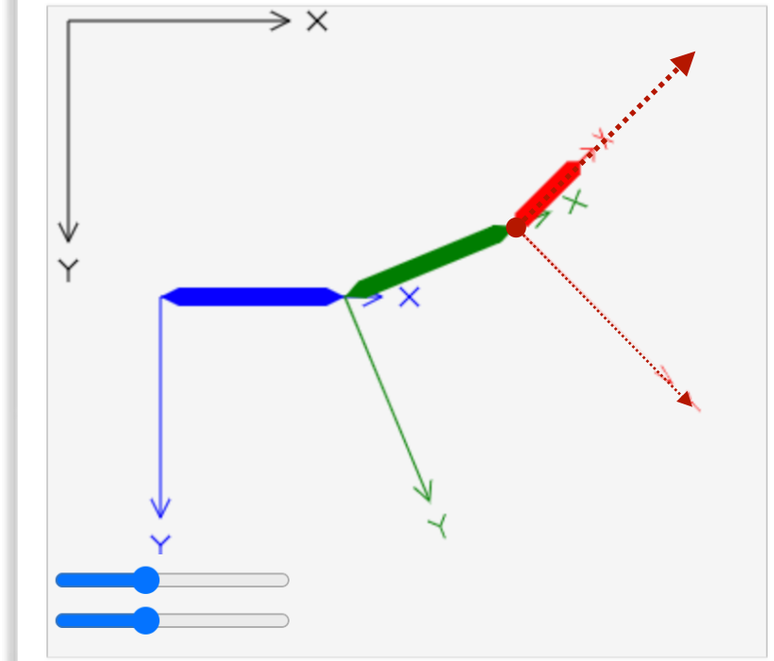
```
[...]

    // make sure you understand these
    axes("black");
    context.translate(50,150);
    linkage("blue");
    context.translate(100,0);
    context.rotate(theta1);
    linkage("green");
    context.translate(100,0);
    context.rotate(phi1);
    context.scale(0.5,1);
    linkage("red");

[...]
```

Translate (50,50) ----- Translate (100,0) ----- Rotate (θ_1) ----- Translate (100,0)

How are transforms combined?

jsbin.com/hozeyar

Week3/Demo0



JavaScript

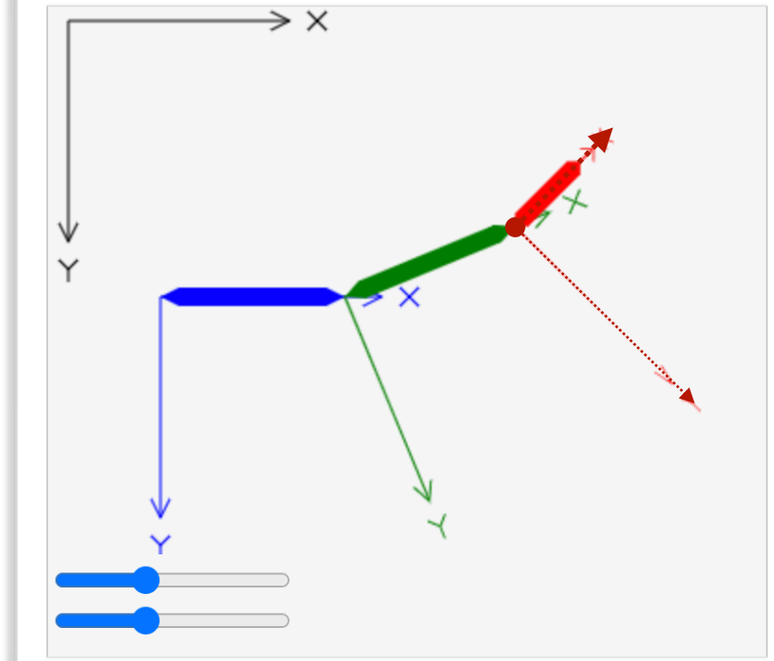
```
[...]

    // make sure you understand these
    axes("black");
    context.translate(50,150);
    linkage("blue");
    context.translate(100,0);
    context.rotate(theta1);
    linkage("green");
    context.translate(100,0);
    context.rotate(phi1);
    context.scale(0.5,1);
    linkage("red");

[...]
```

Translate (50,50)	Translate (100,0)	Rotate (θ_1)	Translate (100,0)	Rotate (φ_1)

How are transforms combined?

jsbin.com/hozeyar

Week3/Demo0



JavaScript

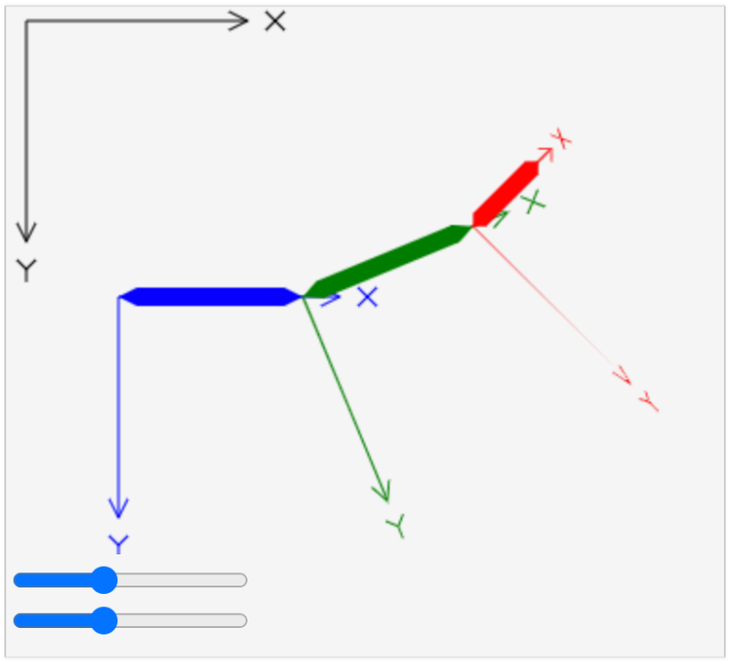
```
[...]

    // make sure you understand these
    axes("black");
    context.translate(50,150);
    linkage("blue");
    context.translate(100,0);
    context.rotate(theta1);
    linkage("green");
    context.translate(100,0);
    context.rotate(phi1);
    context.scale(0.5,1);
    linkage("red");

[...]
```

Translate (50,50)	Translate (100,0)	Rotate (θ_1)	Translate (100,0)	Rotate (φ_1)	Scale (0.5,1)

How are transforms combined?

jsbin.com/hozeyar

Week3/Demo0



JavaScript

```
[...]

    // make sure you understand these
    axes("black");
    context.translate(50,150);
    linkage("blue");
    context.translate(100,0);
    context.rotate(theta1);
    linkage("green");
    context.translate(100,0);
    context.rotate(phi1);
    context.scale(0.5,1);
    linkage("red");

[...]
```

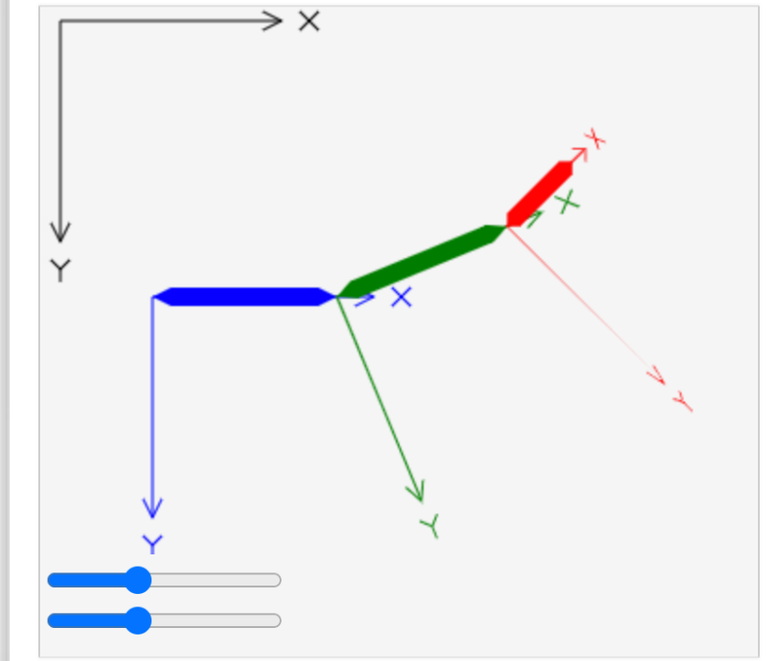
Transforms are introduced in this order
(current coordinate system is obtained by combining transforms left-to-right)

Points drawn in the current coordinate system are transformed back to canvas coordinates by applying the transforms in this order (right-to-left)

How are transforms combined?

jsbin.com/hozeyar

Week3/Demo0



JavaScript

```
[...]

    // make sure you understand these
    axes("black");
    context.translate(50,150);
    linkage("blue");
    context.translate(100,0);
    context.rotate(theta1);
    linkage("green");
    context.translate(100,0);
    context.rotate(phi1);
    context.scale(0.5,1);
    linkage("red");

[...]
```

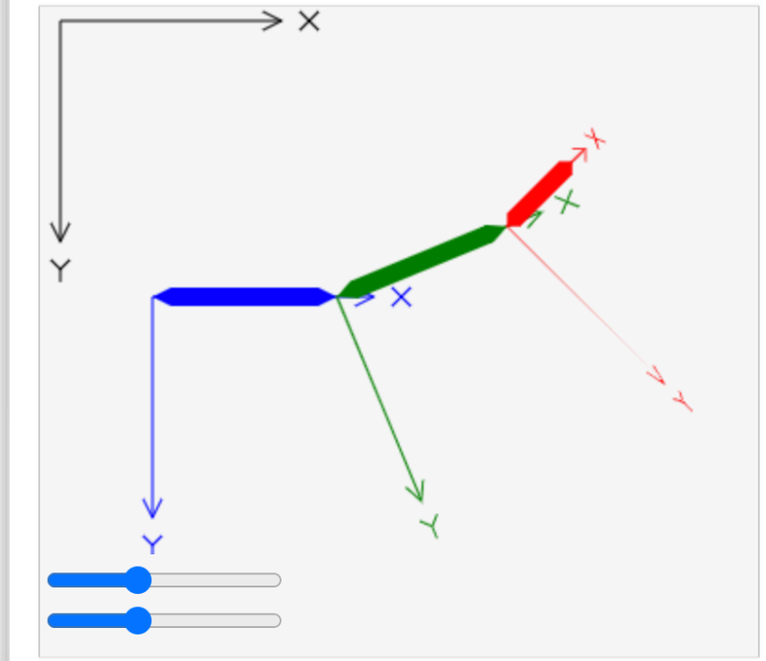
Intuitively, you can combine sub-sequences of this transform chain into "composite" transforms that do something more than elementary ones

Translate (50,50)	-----	Translate (100,0)	-----	Rotate (θ_1)	-----	Translate (100,0)	-----	Rotate (φ_1)	-----	Scale (0.5,1)

How are transforms combined?

jsbin.com/hozeyar

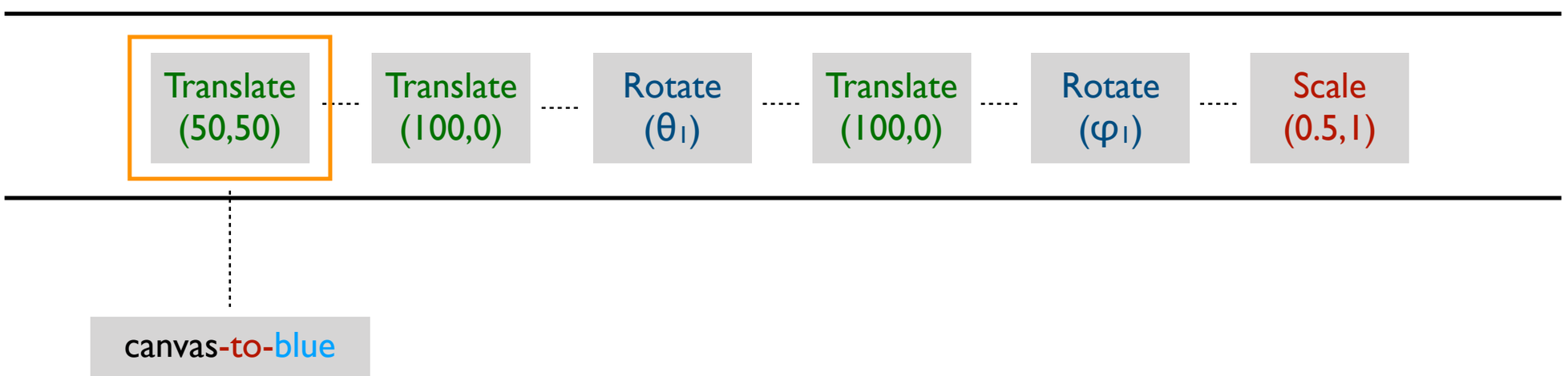
Week3/Demo0



JavaScript

```
[...]

    // make sure you understand these
    axes("black");
    context.translate(50,150);
    linkage("blue");
    context.translate(100,0);
    context.rotate(theta1);
    linkage("green");
    context.translate(100,0);
    context.rotate(phi1);
    context.scale(0.5,1);
    linkage("red");

[...]
```

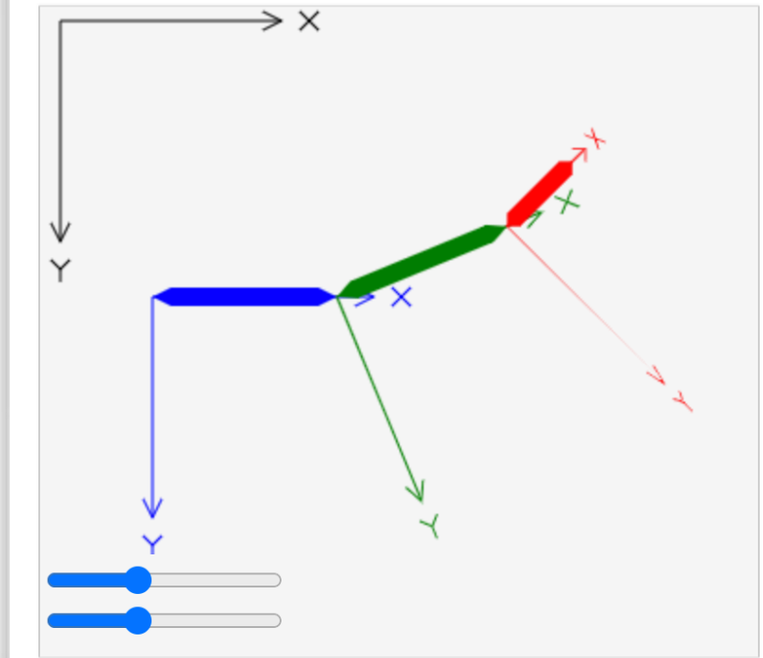
Intuitively, you can combine sub-sequences of this transform chain into "composite" transforms that do something more than elementary ones



How are transforms combined?

jsbin.com/hozeyar

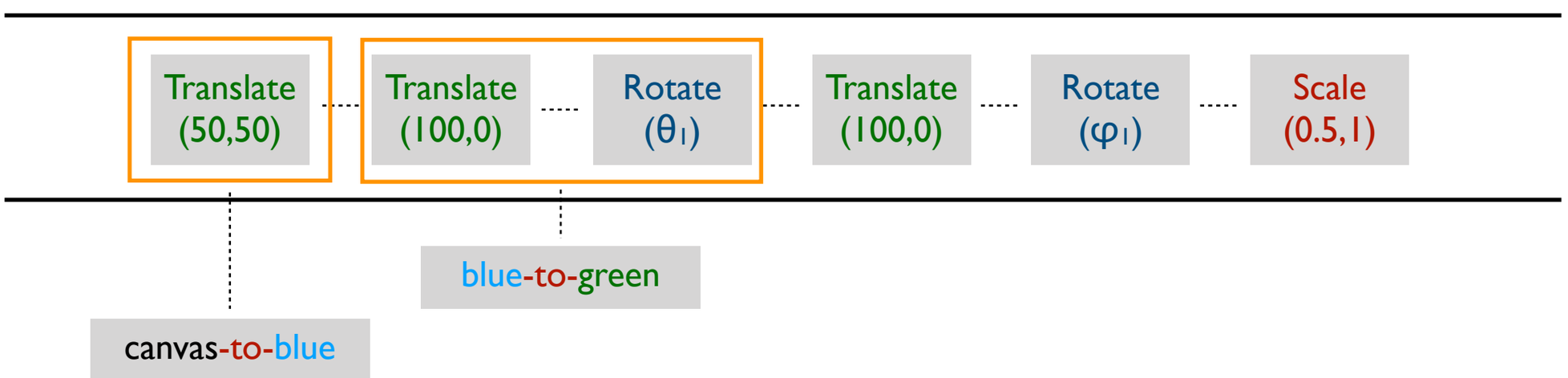
Week3/Demo0



JavaScript

```
[...]

    // make sure you understand these
    axes("black");
    context.translate(50,150);
    linkage("blue");
    context.translate(100,0);
    context.rotate(theta1);
    linkage("green");
    context.translate(100,0);
    context.rotate(phi1);
    context.scale(0.5,1);
    linkage("red");

[...]
```

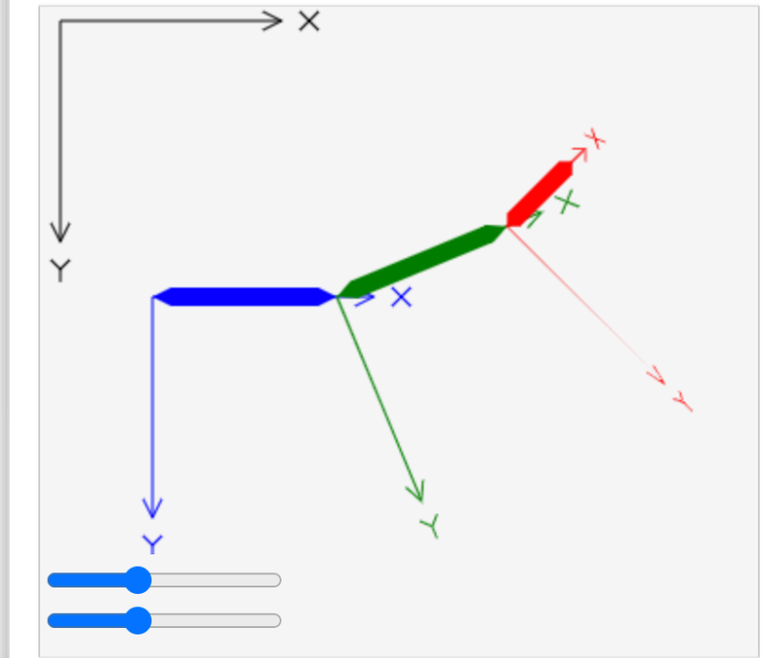
Intuitively, you can combine sub-sequences of this transform chain into "composite" transforms that do something more than elementary ones



How are transforms combined?

jsbin.com/hozeyar

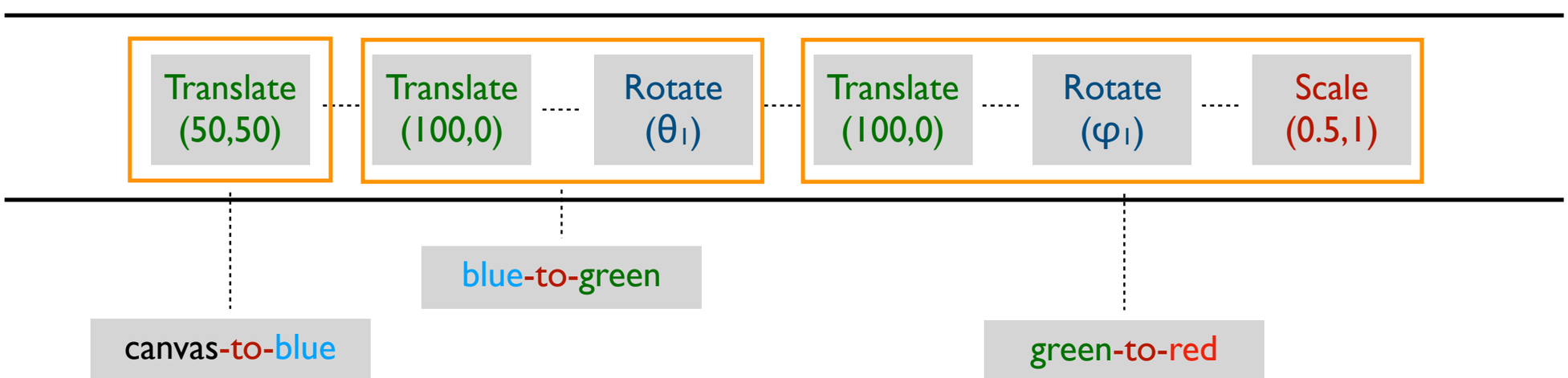
Week3/Demo0



JavaScript

```
[...]

    // make sure you understand these
    axes("black");
    context.translate(50,150);
    linkage("blue");
    context.translate(100,0);
    context.rotate(theta1);
    linkage("green");
    context.translate(100,0);
    context.rotate(phi1);
    context.scale(0.5,1);
    linkage("red");

[...]
```

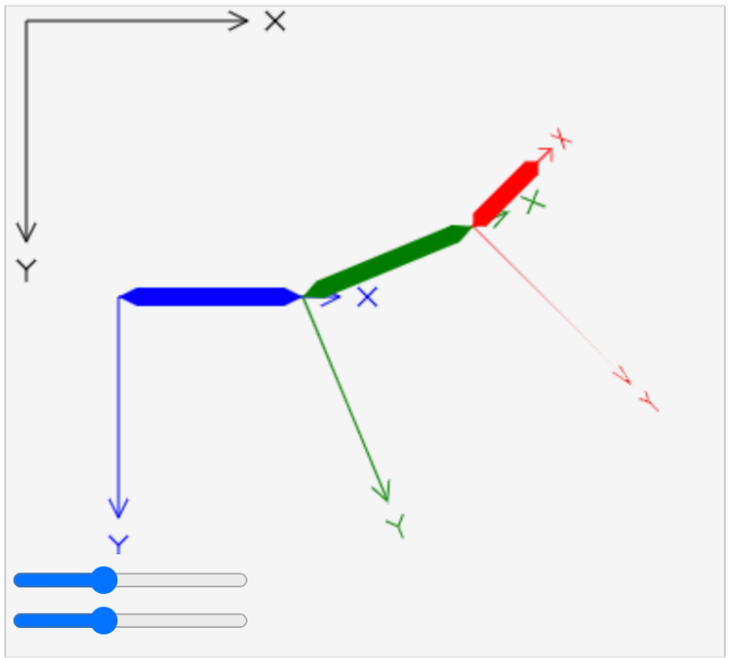
Intuitively, you can combine sub-sequences of this transform chain into "composite" transforms that do something more than elementary ones



How are transforms combined?

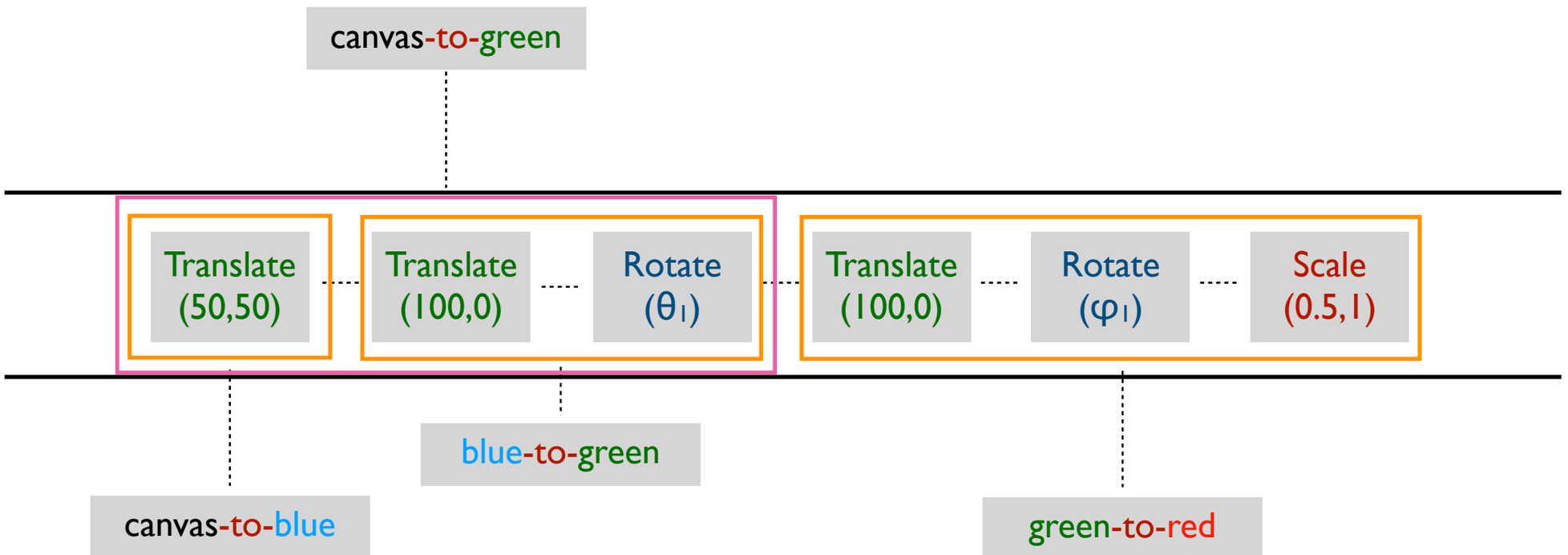
jsbin.com/hozeyar

Week3/Demo0



JavaScript

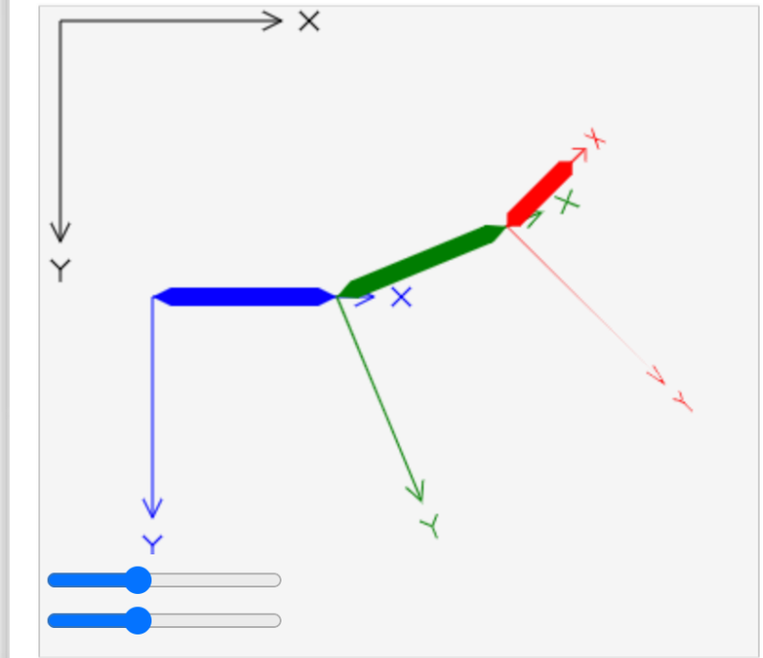
```
[...]

    // make sure you understand these
    axes("black");
    context.translate(50,150);
    linkage("blue");
    context.translate(100,0);
    context.rotate(theta1);
    linkage("green");
    context.translate(100,0);
    context.rotate(phi1);
    context.scale(0.5,1);
    linkage("red");

[...]
```



How are transforms combined?

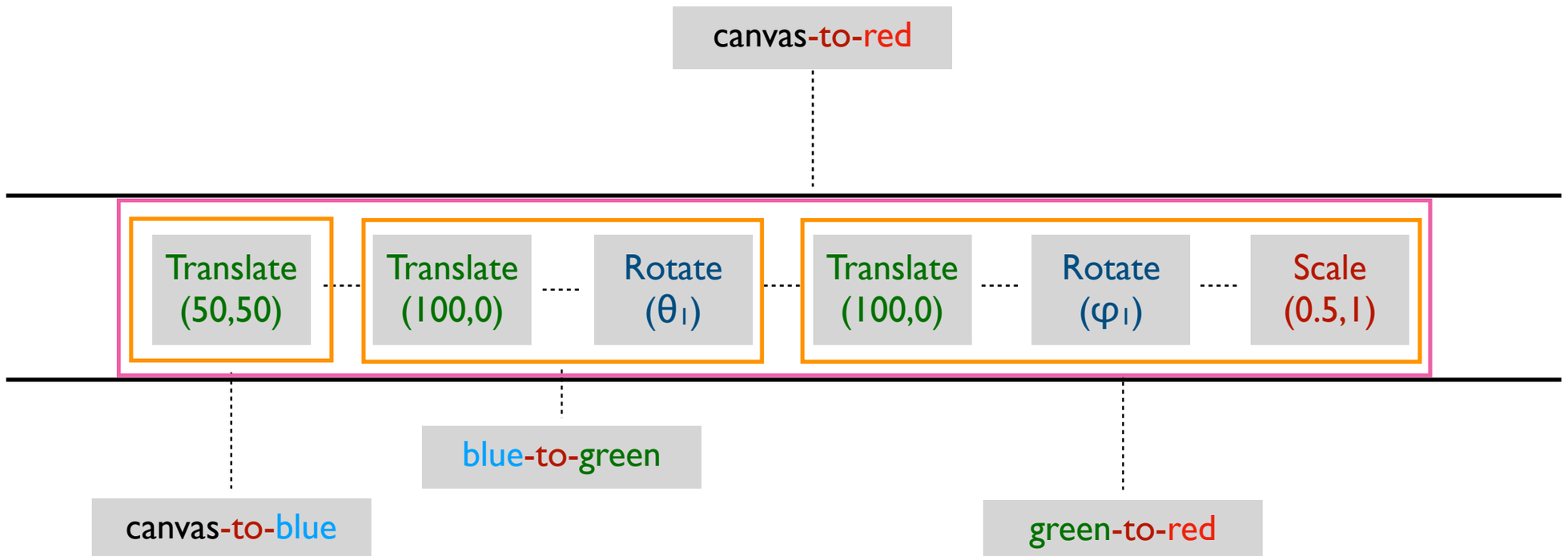
jsbin.com/hozeyar

Week3/Demo0



JavaScript

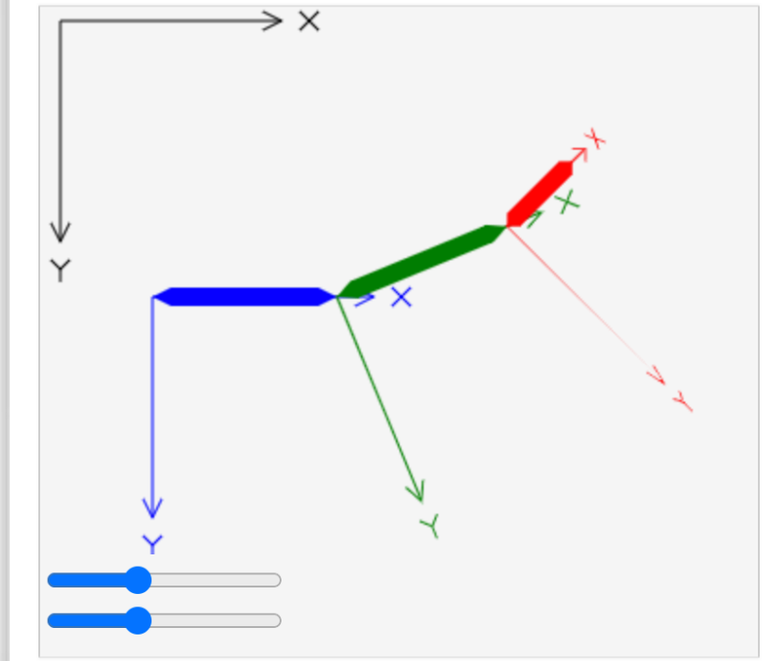
```
[...]

    // make sure you understand these
    axes("black");
    context.translate(50,150);
    linkage("blue");
    context.translate(100,0);
    context.rotate(theta1);
    linkage("green");
    context.translate(100,0);
    context.rotate(phi1);
    context.scale(0.5,1);
    linkage("red");

[...]
```



How are transforms combined?

jsbin.com/hozeyar

Week3/Demo0



JavaScript

```
[...]

    // make sure you understand these
    axes("black");
    context.translate(50,150);
    linkage("blue");
    context.translate(100,0);
    context.rotate(theta1);
    linkage("green");
    context.translate(100,0);
    context.rotate(phi1);
    context.scale(0.5,1);
    linkage("red");

[...]
```

canvas-to-blue ----- blue-to-green ----- green-to-red

For brevity, we may also show transform chains in this fashion, as sequences of "composite" transforms (with the understanding those were built from elementary ones)

Hierarchical modeling

jsbin.com/wovupusife

Week3/Demo1

- 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
  - In many cases, different instances do not rely on each other in just a linear/sequential fashion, but there is a "tree" of dependencies of their respective transforms

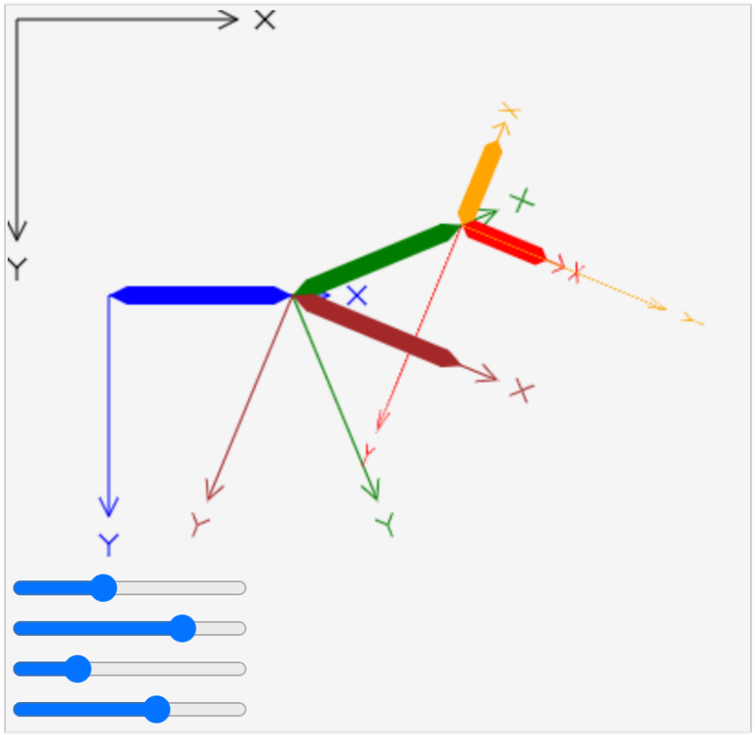
Hierarchical modeling

jsbin.com/wovupusife

Week3/Demo1

- Hierarchically dependent transforms are facilitated in Canvas by the save()/restore() functionality
  - Instead of a single chain of transforms (actually single combined transform representing a chain), Canvas maintains a ***stack*** of composite transforms.
    - Issuing transforms, via Canvas commands, appends to the transform chain at the ***top*** of the stack
    - The ***save()*** command duplicates the top of the stack, and pushes the duplicate copy down the stack.
    - The ***restore()*** command pops the transform (transform chain, or combined transform, in fact) from the stack top

Canvas transform stack



Canvas transform stack

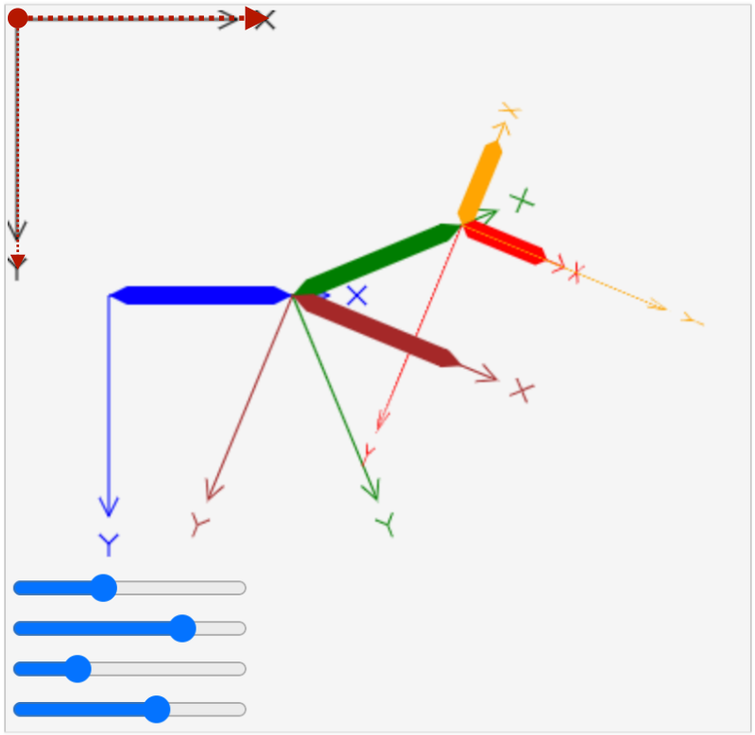
jsbin.com/wuyutirife

Week3/Demo2

JavaScript

```
[...]
  // still in Canvas coordinate system …

  context.translate(50,150);  // Transform from Canvas coordinate system ->
                              //   Blue coordinate system
                              // Stack is now : Blue -> Canvas (top)
  linkage("blue");
  context.save();             // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(theta1);     // Transform Green -> Blue is prefixed to top of stack
                              // Stack is now : Green -> Blue -> Canvas (top)
                              //                Blue -> Canvas

  linkage("green");
  context.save();             // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(phi1);
  context.scale(0.5,1);       // Transform Red -> Green is prefixed to top of stack
                              // Stack is now : Red -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("red");
    context.restore();        // We "pop" the Red transform (top of stack)
                              // Stack is now : Green -> Blue -> Canvas
                              //                Blue -> Canvas

    context.save();           // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(phi2);
    context.scale(0.5,1);     // Transform Orange -> Green is prefixed to top of stack
                              // Stack is now : Orange -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("orange");
    context.restore();        // Pop Stack twice -- essentially undo the Orange
                              //   -> Green -> Blue transforms
    context.restore();        // Stack is now : Blue -> Canvas (top)

    context.save();           // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(theta2);   // Transform Brown -> Blue is prefixed to top of stack
                              // Stack is now : Brown -> Blue -> Canvas (top)
                              //                Blue -> Canvas
    linkage("brown");
    context.restore();        // Stack is now : Blue -> Canvas (top)
[...]
```

Canvas transform stack



Canvas transform stack

(empty)

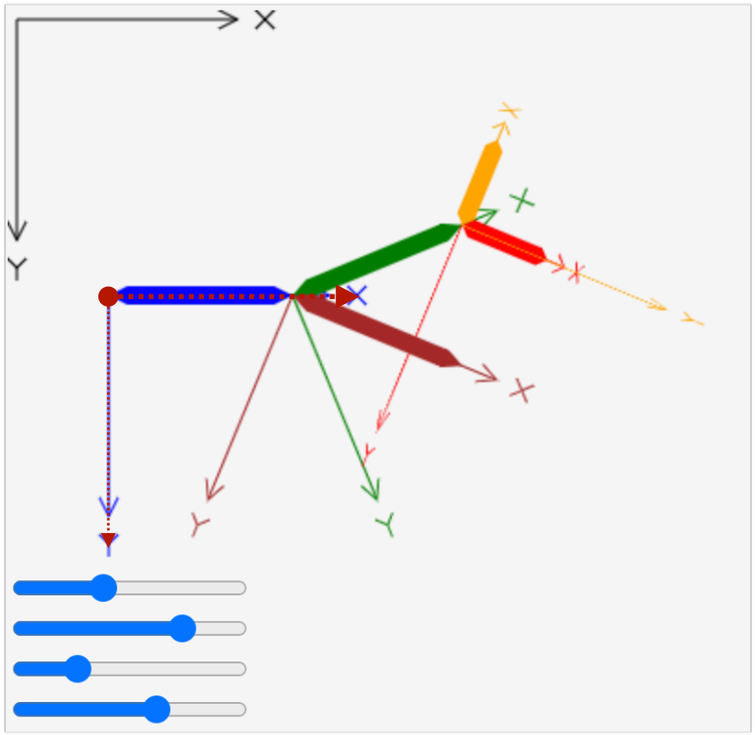
jsbin.com/wuyutirife

Week3/Demo2

JavaScript

```
[...]
  // still in Canvas coordinate system …

  context.translate(50,150);  // Transform from Canvas coordinate system ->
                              //   Blue coordinate system
                              // Stack is now : Blue -> Canvas (top)
  linkage("blue");
  context.save();             // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(theta1);     // Transform Green -> Blue is prefixed to top of stack
                              // Stack is now : Green -> Blue -> Canvas (top)
                              //                Blue -> Canvas

  linkage("green");
  context.save();             // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(phi1);
  context.scale(0.5,1);       // Transform Red -> Green is prefixed to top of stack
                              // Stack is now : Red -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("red");
    context.restore();        // We "pop" the Red transform (top of stack)
                              // Stack is now : Green -> Blue -> Canvas
                              //                Blue -> Canvas

    context.save();           // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(phi2);
    context.scale(0.5,1);     // Transform Orange -> Green is prefixed to top of stack
                              // Stack is now : Orange -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("orange");
    context.restore();        // Pop Stack twice -- essentially undo the Orange
                              //   -> Green -> Blue transforms
    context.restore();        // Stack is now : Blue -> Canvas (top)

    context.save();           // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(theta2);   // Transform Brown -> Blue is prefixed to top of stack
                              // Stack is now : Brown -> Blue -> Canvas (top)
                              //                Blue -> Canvas
    linkage("brown");
    context.restore();        // Stack is now : Blue -> Canvas (top)
[...]
```

Canvas transform stack



Canvas transform stack

canvas-to-blue

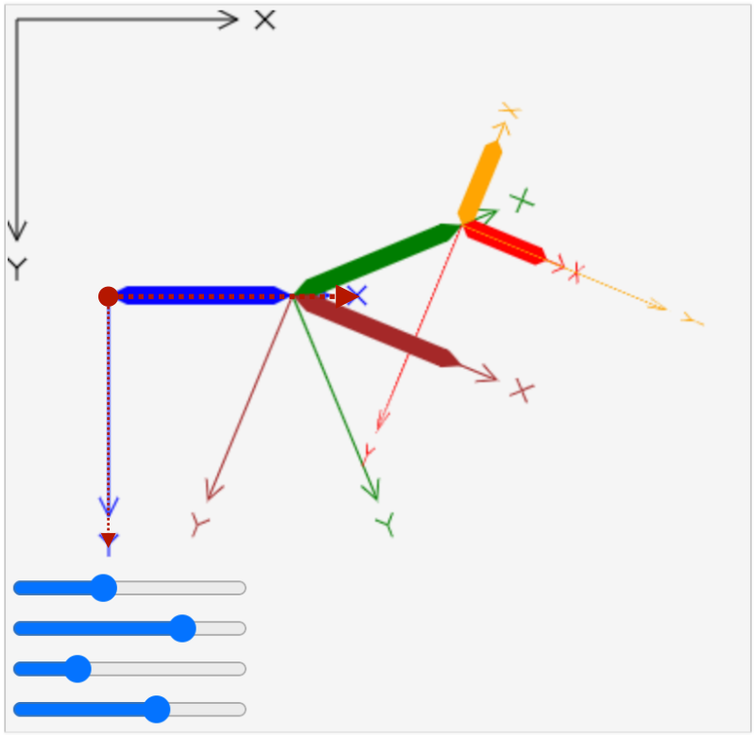
jsbin.com/wuyutirife

Week3/Demo2

JavaScript

```
[...]
  // still in Canvas coordinate system …

  context.translate(50,150);  // Transform from Canvas coordinate system ->
                              //   Blue coordinate system
                              // Stack is now : Blue -> Canvas (top)
  linkage("blue");
  context.save();             // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(theta1);     // Transform Green -> Blue is prefixed to top of stack
                              // Stack is now : Green -> Blue -> Canvas (top)
                              //                Blue -> Canvas

  linkage("green");
  context.save();             // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(phi1);
  context.scale(0.5,1);       // Transform Red -> Green is prefixed to top of stack
                              // Stack is now : Red -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("red");
    context.restore();        // We "pop" the Red transform (top of stack)
                              // Stack is now : Green -> Blue -> Canvas
                              //                Blue -> Canvas

    context.save();           // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(phi2);
    context.scale(0.5,1);     // Transform Orange -> Green is prefixed to top of stack
                              // Stack is now : Orange -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("orange");
    context.restore();        // Pop Stack twice -- essentially undo the Orange
                              //   -> Green -> Blue transforms
    context.restore();        // Stack is now : Blue -> Canvas (top)

    context.save();           // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(theta2);   // Transform Brown -> Blue is prefixed to top of stack
                              // Stack is now : Brown -> Blue -> Canvas (top)
                              //                Blue -> Canvas
    linkage("brown");
    context.restore();        // Stack is now : Blue -> Canvas (top)
[...]
```

Canvas transform stack



Canvas transform stack

canvas-to-blue

canvas-to-blue

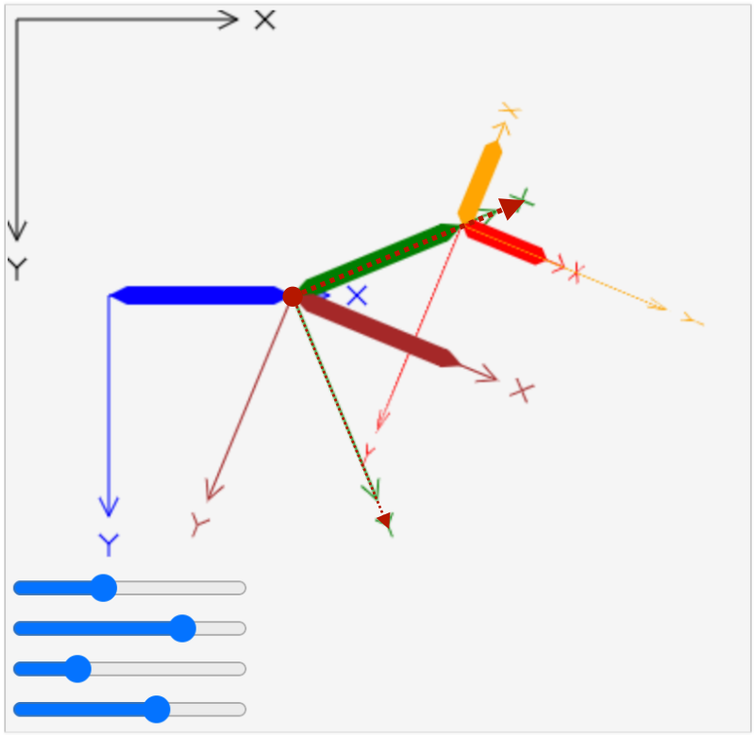
jsbin.com/wuyutirife

Week3/Demo2

JavaScript

```
[...]
  // still in Canvas coordinate system …

  context.translate(50,150);  // Transform from Canvas coordinate system ->
                              //   Blue coordinate system
                              // Stack is now : Blue -> Canvas (top)
  linkage("blue");
  context.save();             // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(theta1);     // Transform Green -> Blue is prefixed to top of stack
                              // Stack is now : Green -> Blue -> Canvas (top)
                              //                Blue -> Canvas

  linkage("green");
  context.save();             // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(phi1);
  context.scale(0.5,1);       // Transform Red -> Green is prefixed to top of stack
                              // Stack is now : Red -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("red");
    context.restore();        // We "pop" the Red transform (top of stack)
                              // Stack is now : Green -> Blue -> Canvas
                              //                Blue -> Canvas

    context.save();           // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(phi2);
    context.scale(0.5,1);     // Transform Orange -> Green is prefixed to top of stack
                              // Stack is now : Orange -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("orange");
    context.restore();        // Pop Stack twice -- essentially undo the Orange
                              //   -> Green -> Blue transforms
    context.restore();        // Stack is now : Blue -> Canvas (top)

    context.save();           // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(theta2);   // Transform Brown -> Blue is prefixed to top of stack
                              // Stack is now : Brown -> Blue -> Canvas (top)
                              //                Blue -> Canvas
    linkage("brown");
    context.restore();        // Stack is now : Blue -> Canvas (top)
[...]
```

Canvas transform stack



Canvas transform stack

canvas-to-blue

blue-to-green

canvas-to-blue

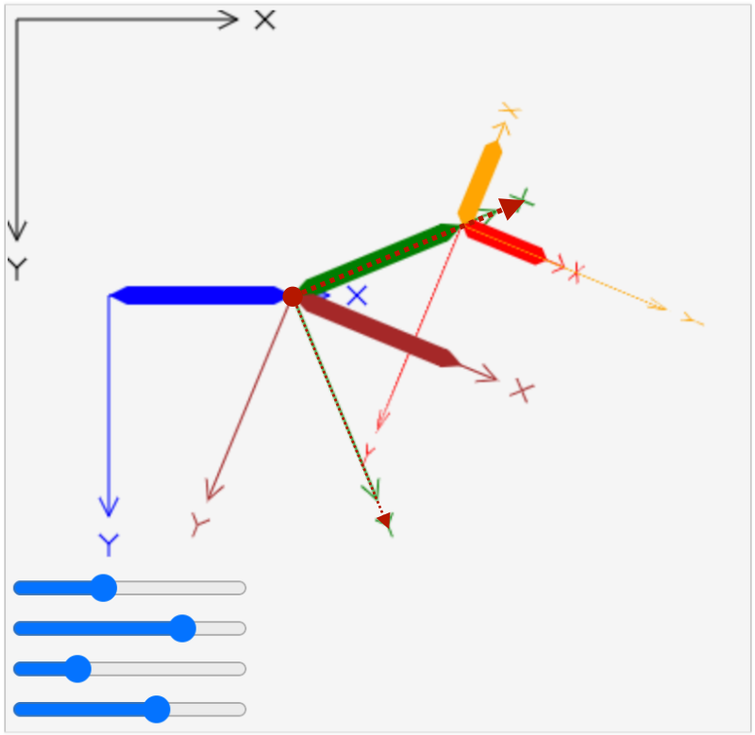
jsbin.com/wuyutirife

Week3/Demo2

JavaScript

```
[...]
  // still in Canvas coordinate system …

  context.translate(50,150);  // Transform from Canvas coordinate system ->
                              //   Blue coordinate system
                              // Stack is now : Blue -> Canvas (top)
  linkage("blue");
  context.save();             // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(theta1);     // Transform Green -> Blue is prefixed to top of stack
                              // Stack is now : Green -> Blue -> Canvas (top)
                              //                Blue -> Canvas

  linkage("green");
  context.save();             // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(phi1);
  context.scale(0.5,1);       // Transform Red -> Green is prefixed to top of stack
                              // Stack is now : Red -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("red");
    context.restore();        // We "pop" the Red transform (top of stack)
                              // Stack is now : Green -> Blue -> Canvas
                              //                Blue -> Canvas

    context.save();           // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(phi2);
    context.scale(0.5,1);     // Transform Orange -> Green is prefixed to top of stack
                              // Stack is now : Orange -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("orange");
    context.restore();        // Pop Stack twice -- essentially undo the Orange
                              //   -> Green -> Blue transforms
    context.restore();        // Stack is now : Blue -> Canvas (top)

    context.save();           // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(theta2);   // Transform Brown -> Blue is prefixed to top of stack
                              // Stack is now : Brown -> Blue -> Canvas (top)
                              //                Blue -> Canvas
    linkage("brown");
    context.restore();        // Stack is now : Blue -> Canvas (top)
[...]
```

Canvas transform stack



Canvas transform stack

canvas-to-blue | blue-to-green

canvas-to-blue | blue-to-green

canvas-to-blue

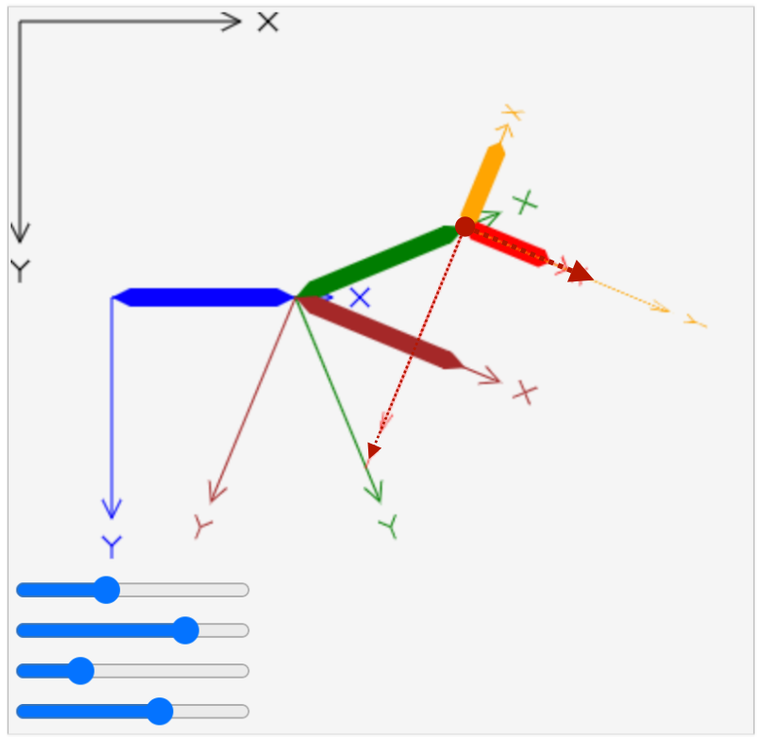
jsbin.com/wuyutirife

Week3/Demo2

JavaScript

```
[...]
  // still in Canvas coordinate system …

  context.translate(50,150);  // Transform from Canvas coordinate system ->
                              //   Blue coordinate system
                              // Stack is now : Blue -> Canvas (top)
  linkage("blue");
  context.save();             // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(theta1);     // Transform Green -> Blue is prefixed to top of stack
                              // Stack is now : Green -> Blue -> Canvas (top)
                              //                Blue -> Canvas

  linkage("green");
  context.save();             // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(phi1);
  context.scale(0.5,1);       // Transform Red -> Green is prefixed to top of stack
                              // Stack is now : Red -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("red");
    context.restore();        // We "pop" the Red transform (top of stack)
                              // Stack is now : Green -> Blue -> Canvas
                              //                Blue -> Canvas

    context.save();           // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(phi2);
    context.scale(0.5,1);     // Transform Orange -> Green is prefixed to top of stack
                              // Stack is now : Orange -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("orange");
    context.restore();        // Pop Stack twice -- essentially undo the Orange
                              //   -> Green -> Blue transforms
    context.restore();        // Stack is now : Blue -> Canvas (top)

    context.save();           // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(theta2);   // Transform Brown -> Blue is prefixed to top of stack
                              // Stack is now : Brown -> Blue -> Canvas (top)
                              //                Blue -> Canvas
    linkage("brown");
    context.restore();        // Stack is now : Blue -> Canvas (top)
[...]
```

Canvas transform stack



Canvas transform stack

canvas-to-blue | blue-to-green | green-to-red

canvas-to-blue | blue-to-green

canvas-to-blue

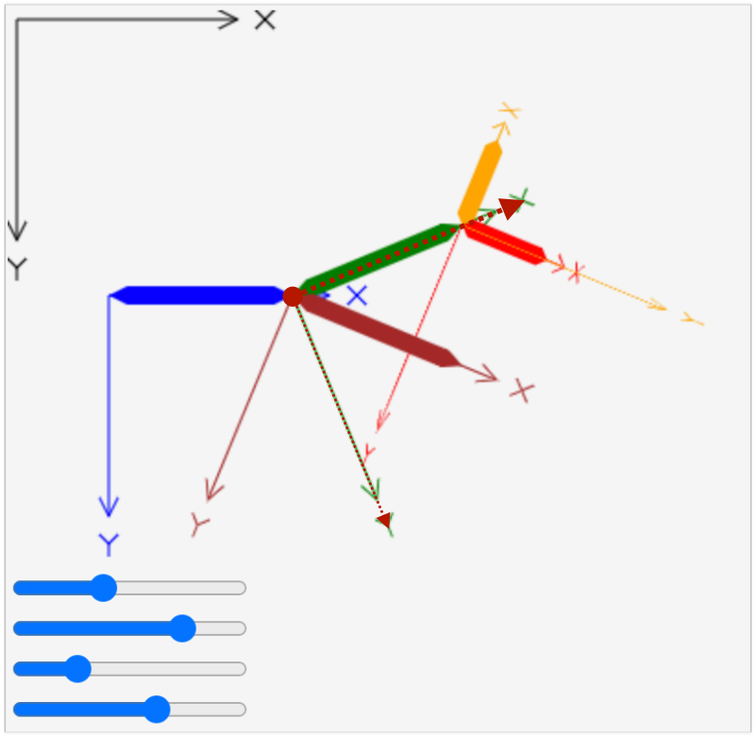
jsbin.com/wuyutirife

Week3/Demo2

JavaScript

```
[...]
  // still in Canvas coordinate system …

  context.translate(50,150);  // Transform from Canvas coordinate system ->
                              //   Blue coordinate system
                              // Stack is now : Blue -> Canvas (top)
  linkage("blue");
  context.save();             // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(theta1);     // Transform Green -> Blue is prefixed to top of stack
                              // Stack is now : Green -> Blue -> Canvas (top)
                              //                Blue -> Canvas

  linkage("green");
  context.save();             // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(phi1);
  context.scale(0.5,1);       // Transform Red -> Green is prefixed to top of stack
                              // Stack is now : Red -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
  linkage("red");
    context.restore();        // We "pop" the Red transform (top of stack)
                              // Stack is now : Green -> Blue -> Canvas
                              //                Blue -> Canvas

    context.save();           // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(phi2);
    context.scale(0.5,1);     // Transform Orange -> Green is prefixed to top of stack
                              // Stack is now : Orange -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("orange");
    context.restore();        // Pop Stack twice -- essentially undo the Orange
                              //   -> Green -> Blue transforms
    context.restore();        // Stack is now : Blue -> Canvas (top)

    context.save();           // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(theta2);   // Transform Brown -> Blue is prefixed to top of stack
                              // Stack is now : Brown -> Blue -> Canvas (top)
                              //                Blue -> Canvas
    linkage("brown");
    context.restore();        // Stack is now : Blue -> Canvas (top)
[...]
```

Canvas transform stack



Canvas transform stack

canvas-to-blue

blue-to-green

canvas-to-blue

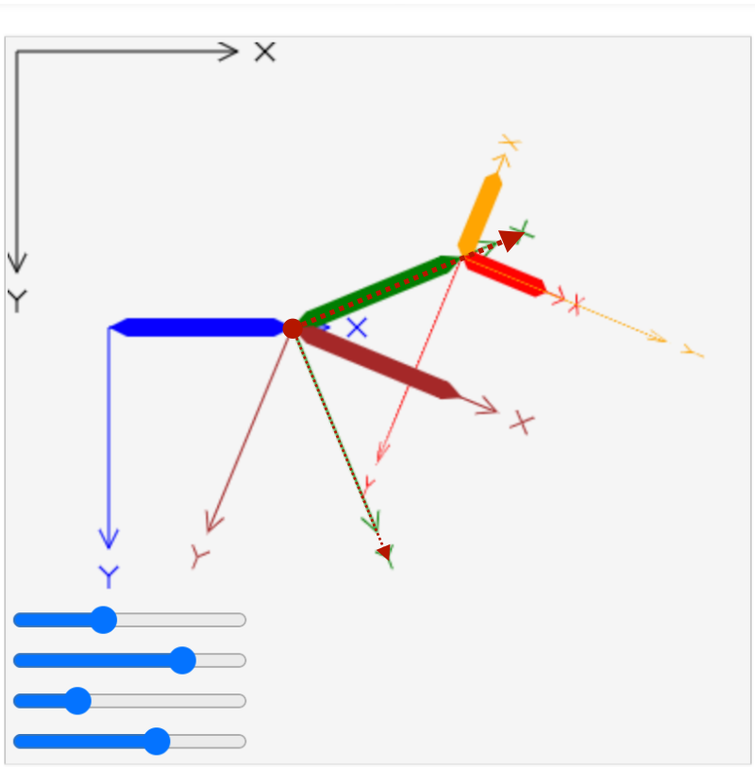
jsbin.com/wuyutirife

Week3/Demo2

JavaScript

```
[...]
  // still in Canvas coordinate system …

  context.translate(50,150);  // Transform from Canvas coordinate system ->
                              //   Blue coordinate system
                              // Stack is now : Blue -> Canvas (top)
  linkage("blue");
  context.save();             // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(theta1);     // Transform Green -> Blue is prefixed to top of stack
                              // Stack is now : Green -> Blue -> Canvas (top)
                              //                Blue -> Canvas

  linkage("green");
  context.save();             // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(phi1);
  context.scale(0.5,1);       // Transform Red -> Green is prefixed to top of stack
                              // Stack is now : Red -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("red");
    context.restore();        // We "pop" the Red transform (top of stack)
                              // Stack is now : Green -> Blue -> Canvas
                              //                Blue -> Canvas

    context.save();           // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(phi2);
    context.scale(0.5,1);     // Transform Orange -> Green is prefixed to top of stack
                              // Stack is now : Orange -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("orange");
    context.restore();        // Pop Stack twice -- essentially undo the Orange
                              //   -> Green -> Blue transforms
    context.restore();        // Stack is now : Blue -> Canvas (top)

    context.save();           // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(theta2);   // Transform Brown -> Blue is prefixed to top of stack
                              // Stack is now : Brown -> Blue -> Canvas (top)
                              //                Blue -> Canvas
    linkage("brown");
    context.restore();        // Stack is now : Blue -> Canvas (top)
[...]
```

Canvas transform stack



Canvas transform stack

canvas-to-blue | blue-to-green

canvas-to-blue | blue-to-green

canvas-to-blue

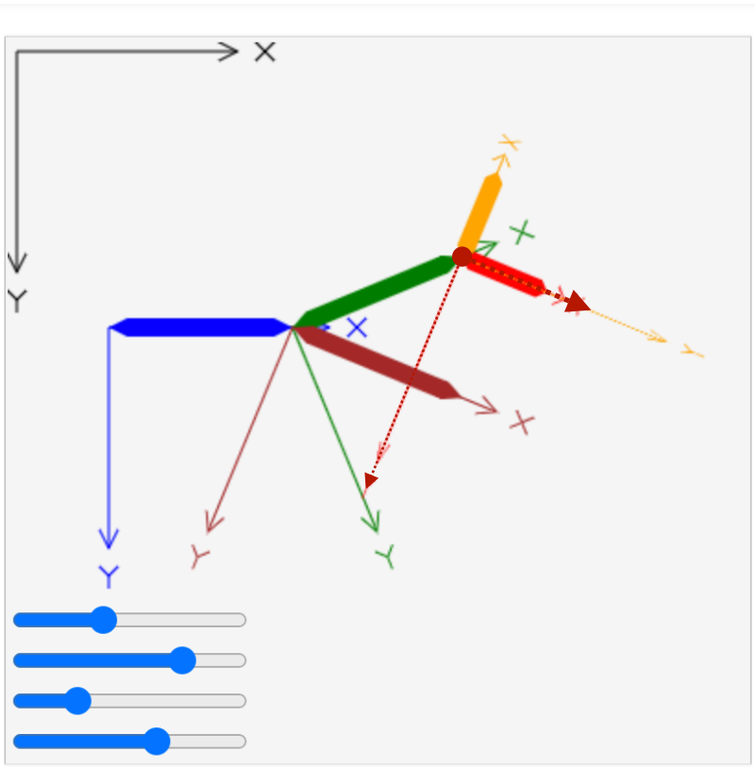
jsbin.com/wuyutirife

Week3/Demo2

JavaScript

```
[...]
  // still in Canvas coordinate system …

  context.translate(50,150);  // Transform from Canvas coordinate system ->
                              //   Blue coordinate system
                              // Stack is now : Blue -> Canvas (top)
  linkage("blue");
  context.save();             // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(theta1);     // Transform Green -> Blue is prefixed to top of stack
                              // Stack is now : Green -> Blue -> Canvas (top)
                              //                Blue -> Canvas

  linkage("green");
  context.save();             // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(phi1);
  context.scale(0.5,1);       // Transform Red -> Green is prefixed to top of stack
                              // Stack is now : Red -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("red");
    context.restore();        // We "pop" the Red transform (top of stack)
                              // Stack is now : Green -> Blue -> Canvas
                              //                Blue -> Canvas

    context.save();           // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(phi2);
    context.scale(0.5,1);     // Transform Orange -> Green is prefixed to top of stack
                              // Stack is now : Orange -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("orange");
    context.restore();        // Pop Stack twice -- essentially undo the Orange
                              //   -> Green -> Blue transforms
    context.restore();        // Stack is now : Blue -> Canvas (top)

    context.save();           // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(theta2);   // Transform Brown -> Blue is prefixed to top of stack
                              // Stack is now : Brown -> Blue -> Canvas (top)
                              //                Blue -> Canvas
    linkage("brown");
    context.restore();        // Stack is now : Blue -> Canvas (top)
[...]
```

Canvas transform stack



Canvas transform stack

canvas-to-blue | blue-to-green | green-to-orange

canvas-to-blue | blue-to-green

canvas-to-blue

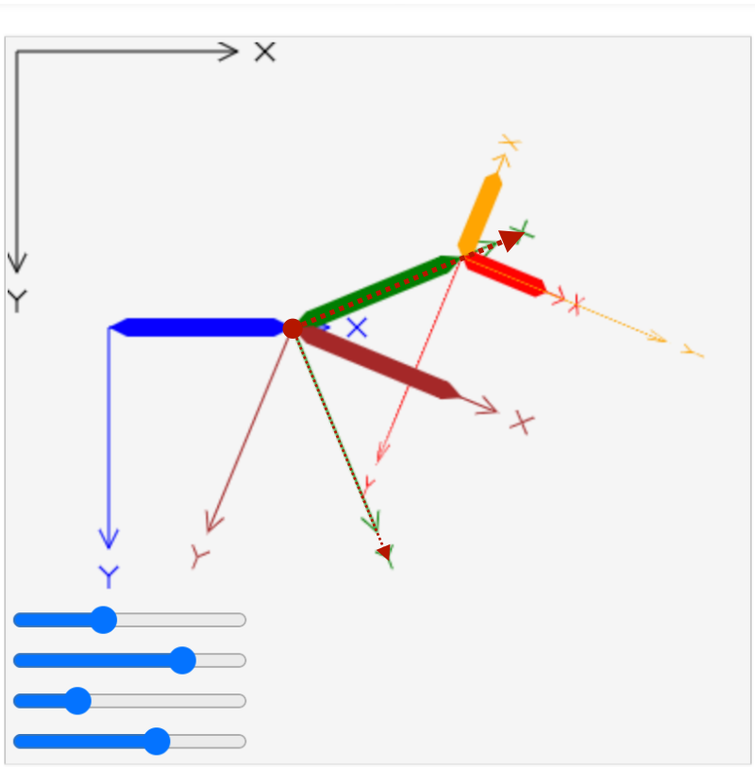
jsbin.com/wuyutirife

Week3/Demo2

JavaScript

```
[...]
  // still in Canvas coordinate system …

  context.translate(50,150);  // Transform from Canvas coordinate system ->
                              //   Blue coordinate system
                              // Stack is now : Blue -> Canvas (top)
  linkage("blue");
  context.save();             // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(theta1);     // Transform Green -> Blue is prefixed to top of stack
                              // Stack is now : Green -> Blue -> Canvas (top)
                              //                Blue -> Canvas

  linkage("green");
  context.save();             // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(phi1);
  context.scale(0.5,1);       // Transform Red -> Green is prefixed to top of stack
                              // Stack is now : Red -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("red");
    context.restore();        // We "pop" the Red transform (top of stack)
                              // Stack is now : Green -> Blue -> Canvas
                              //                Blue -> Canvas

    context.save();           // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(phi2);
    context.scale(0.5,1);     // Transform Orange -> Green is prefixed to top of stack
                              // Stack is now : Orange -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("orange");
    context.restore();        // Pop Stack twice -- essentially undo the Orange
                              //   -> Green -> Blue transforms
    context.restore();        // Stack is now : Blue -> Canvas (top)

    context.save();           // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(theta2);   // Transform Brown -> Blue is prefixed to top of stack
                              // Stack is now : Brown -> Blue -> Canvas (top)
                              //                Blue -> Canvas
    linkage("brown");
    context.restore();        // Stack is now : Blue -> Canvas (top)
[...]
```

Canvas transform stack



Canvas transform stack

canvas-to-blue

blue-to-green

canvas-to-blue

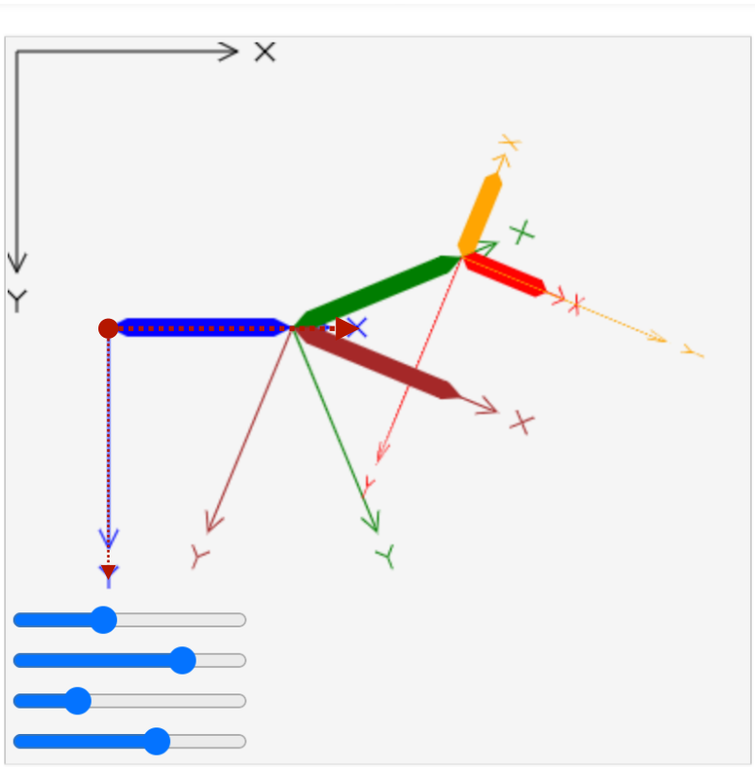
jsbin.com/wuyutirife

Week3/Demo2

JavaScript

```
[...]
  // still in Canvas coordinate system …

  context.translate(50,150);  // Transform from Canvas coordinate system ->
                              //   Blue coordinate system
                              // Stack is now : Blue -> Canvas (top)
  linkage("blue");
  context.save();             // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(theta1);     // Transform Green -> Blue is prefixed to top of stack
                              // Stack is now : Green -> Blue -> Canvas (top)
                              //                Blue -> Canvas

  linkage("green");
  context.save();             // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(phi1);
  context.scale(0.5,1);       // Transform Red -> Green is prefixed to top of stack
                              // Stack is now : Red -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("red");
    context.restore();        // We "pop" the Red transform (top of stack)
                              // Stack is now : Green -> Blue -> Canvas
                              //                Blue -> Canvas

    context.save();           // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(phi2);
    context.scale(0.5,1);     // Transform Orange -> Green is prefixed to top of stack
                              // Stack is now : Orange -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("orange");
    context.restore();        // Pop Stack twice -- essentially undo the Orange
                              //   -> Green -> Blue transforms
    context.restore();        // Stack is now : Blue -> Canvas (top)

    context.save();           // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(theta2);   // Transform Brown -> Blue is prefixed to top of stack
                              // Stack is now : Brown -> Blue -> Canvas (top)
                              //                Blue -> Canvas
    linkage("brown");
    context.restore();        // Stack is now : Blue -> Canvas (top)
[...]
```

Canvas transform stack



Canvas transform stack

canvas-to-blue

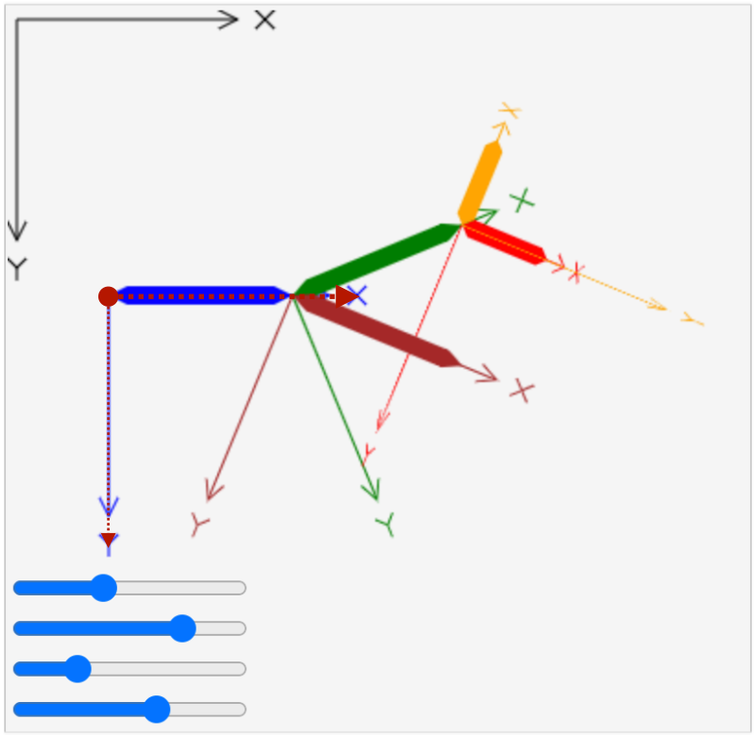
jsbin.com/wuyutirife

Week3/Demo2

JavaScript

```
[...]
  // still in Canvas coordinate system …

  context.translate(50,150);  // Transform from Canvas coordinate system ->
                              //   Blue coordinate system
                              // Stack is now : Blue -> Canvas (top)
  linkage("blue");
  context.save();             // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(theta1);     // Transform Green -> Blue is prefixed to top of stack
                              // Stack is now : Green -> Blue -> Canvas (top)
                              //                Blue -> Canvas

  linkage("green");
  context.save();             // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(phi1);
  context.scale(0.5,1);       // Transform Red -> Green is prefixed to top of stack
                              // Stack is now : Red -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("red");
    context.restore();        // We "pop" the Red transform (top of stack)
                              // Stack is now : Green -> Blue -> Canvas
                              //                Blue -> Canvas

    context.save();           // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(phi2);
    context.scale(0.5,1);     // Transform Orange -> Green is prefixed to top of stack
                              // Stack is now : Orange -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("orange");
    context.restore();        // Pop Stack twice -- essentially undo the Orange
                              //   -> Green -> Blue transforms
    context.restore();        // Stack is now : Blue -> Canvas (top)

    context.save();           // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(theta2);   // Transform Brown -> Blue is prefixed to top of stack
                              // Stack is now : Brown -> Blue -> Canvas (top)
                              //                Blue -> Canvas
    linkage("brown");
    context.restore();        // Stack is now : Blue -> Canvas (top)
[...]
```

Canvas transform stack



Canvas transform stack

canvas-to-blue

canvas-to-blue

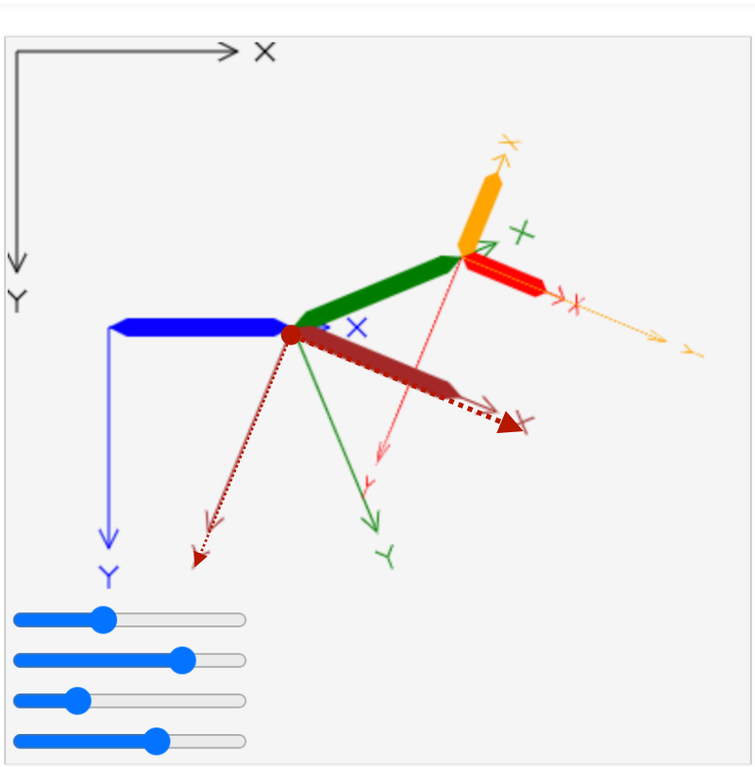
jsbin.com/wuyutirife

Week3/Demo2

JavaScript

```
[...]
  // still in Canvas coordinate system …

  context.translate(50,150);  // Transform from Canvas coordinate system ->
                              //   Blue coordinate system
                              // Stack is now : Blue -> Canvas (top)
  linkage("blue");
  context.save();             // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(theta1);     // Transform Green -> Blue is prefixed to top of stack
                              // Stack is now : Green -> Blue -> Canvas (top)
                              //                Blue -> Canvas

  linkage("green");
  context.save();             // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(phi1);
  context.scale(0.5,1);       // Transform Red -> Green is prefixed to top of stack
                              // Stack is now : Red -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("red");
    context.restore();        // We "pop" the Red transform (top of stack)
                              // Stack is now : Green -> Blue -> Canvas
                              //                Blue -> Canvas

    context.save();           // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(phi2);
    context.scale(0.5,1);     // Transform Orange -> Green is prefixed to top of stack
                              // Stack is now : Orange -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("orange");
    context.restore();        // Pop Stack twice -- essentially undo the Orange
                              //   -> Green -> Blue transforms
    context.restore();        // Stack is now : Blue -> Canvas (top)

    context.save();           // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(theta2);   // Transform Brown -> Blue is prefixed to top of stack
                              // Stack is now : Brown -> Blue -> Canvas (top)
                              //                Blue -> Canvas
    linkage("brown");
    context.restore();        // Stack is now : Blue -> Canvas (top)
[...]
```

Canvas transform stack



Canvas transform stack

canvas-to-blue

Blue-to-brown

canvas-to-blue

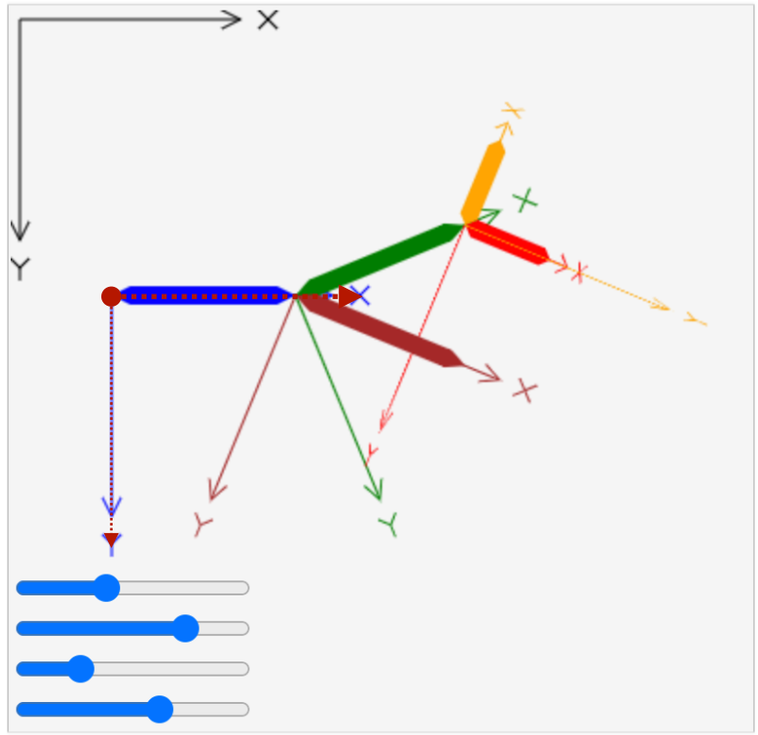
jsbin.com/wuyutirife

Week3/Demo2

JavaScript

```
[...]
  // still in Canvas coordinate system …

  context.translate(50,150);  // Transform from Canvas coordinate system ->
                              //   Blue coordinate system
                              // Stack is now : Blue -> Canvas (top)
  linkage("blue");
  context.save();             // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(theta1);     // Transform Green -> Blue is prefixed to top of stack
                              // Stack is now : Green -> Blue -> Canvas (top)
                              //                Blue -> Canvas

  linkage("green");
  context.save();             // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(phi1);
  context.scale(0.5,1);       // Transform Red -> Green is prefixed to top of stack
                              // Stack is now : Red -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("red");
    context.restore();        // We "pop" the Red transform (top of stack)
                              // Stack is now : Green -> Blue -> Canvas
                              //                Blue -> Canvas

    context.save();           // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(phi2);
    context.scale(0.5,1);     // Transform Orange -> Green is prefixed to top of stack
                              // Stack is now : Orange -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("orange");
    context.restore();        // Pop Stack twice -- essentially undo the Orange
                              //   -> Green -> Blue transforms
    context.restore();        // Stack is now : Blue -> Canvas (top)

    context.save();           // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(theta2);   // Transform Brown -> Blue is prefixed to top of stack
                              // Stack is now : Brown -> Blue -> Canvas (top)
                              //                Blue -> Canvas
    linkage("brown");
    context.restore();        // Stack is now : Blue -> Canvas (top)
[...]
```

Canvas transform stack



Canvas transform stack

canvas-to-blue

jsbin.com/wuyutirife

Week3/Demo2

JavaScript

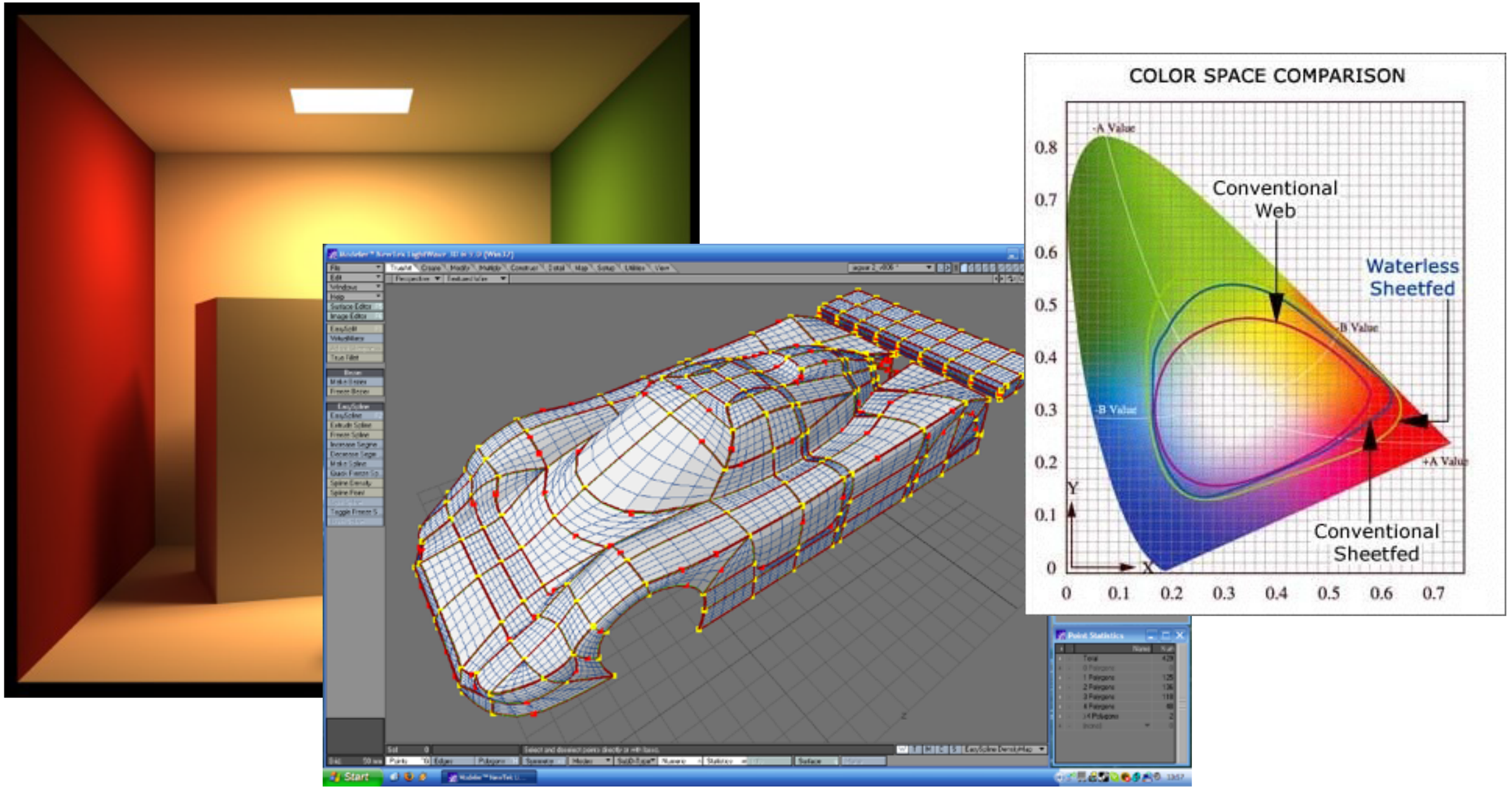
```
[...]
  // still in Canvas coordinate system …

  context.translate(50,150);  // Transform from Canvas coordinate system ->
                              //   Blue coordinate system
                              // Stack is now : Blue -> Canvas (top)
  linkage("blue");
  context.save();             // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(theta1);     // Transform Green -> Blue is prefixed to top of stack
                              // Stack is now : Green -> Blue -> Canvas (top)
                              //                Blue -> Canvas

  linkage("green");
  context.save();             // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
  context.translate(100,0);
  context.rotate(phi1);
  context.scale(0.5,1);       // Transform Red -> Green is prefixed to top of stack
                              // Stack is now : Red -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("red");
    context.restore();        // We "pop" the Red transform (top of stack)
                              // Stack is now : Green -> Blue -> Canvas
                              //                Blue -> Canvas

    context.save();           // Stack is now : Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(phi2);
    context.scale(0.5,1);     // Transform Orange -> Green is prefixed to top of stack
                              // Stack is now : Orange -> Green -> Blue -> Canvas (top)
                              //                Green -> Blue -> Canvas
                              //                Blue -> Canvas
    linkage("orange");
    context.restore();        // Pop Stack twice -- essentially undo the Orange
                              //   -> Green -> Blue transforms
    context.restore();        // Stack is now : Blue -> Canvas (top)

    context.save();           // Stack is now : Blue -> Canvas (top)
                              //                Blue -> Canvas
    context.translate(100,0);
    context.rotate(theta2);   // Transform Brown -> Blue is prefixed to top of stack
                              // Stack is now : Brown -> Blue -> Canvas (top)
                              //                Blue -> Canvas
    linkage("brown");
    context.restore();        // Stack is now : Blue -> Canvas (top)
[...]
```

What's coming next?

- Mathematical representation of elementary transforms
- Algebra of transform compositions and point/vector transformations
- An introductory look at homogeneous coordinates
- Implementation of hierarchical modeling and transform compositions via linear algebra libraries (as opposed to Canvas' internal pipeline and stack)

About your *next* homework …

- Your *next p*rogramming assignment will be released this weekend (about a after the deadline; we'll announce it on Piazza once it goes live)
- It will ask you to implement a moving or interactive picture that leverages hierarchical modeling (and the Canvas stack).
- The description will include useful readings
- Programming assignment #2 will be due Wed Oct 6th



Lecture 5 : Transforms and Hierarchical Modeling in 2D (and overview of the Canvas transform stack)

Thursday September 23rd 2021