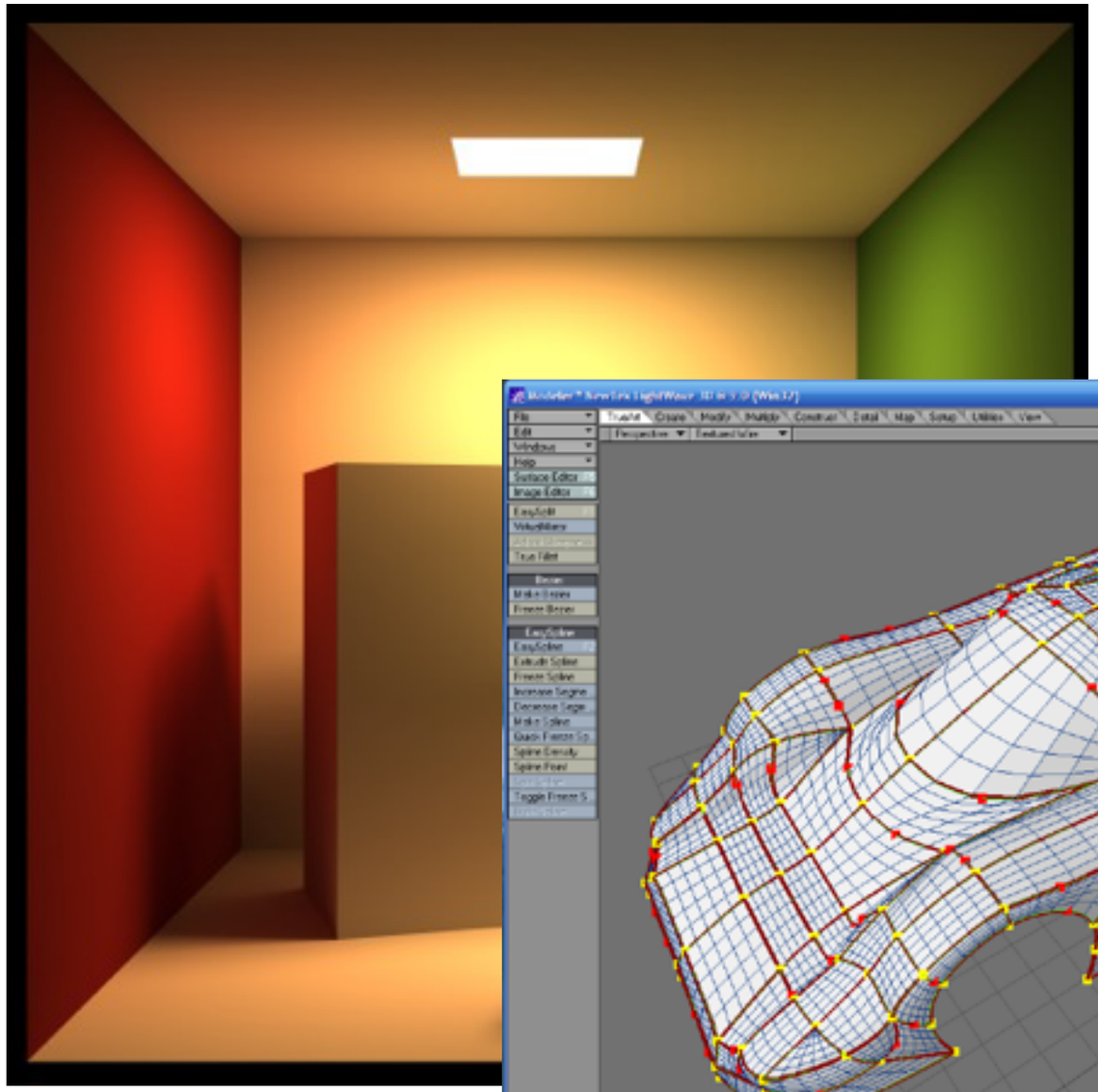
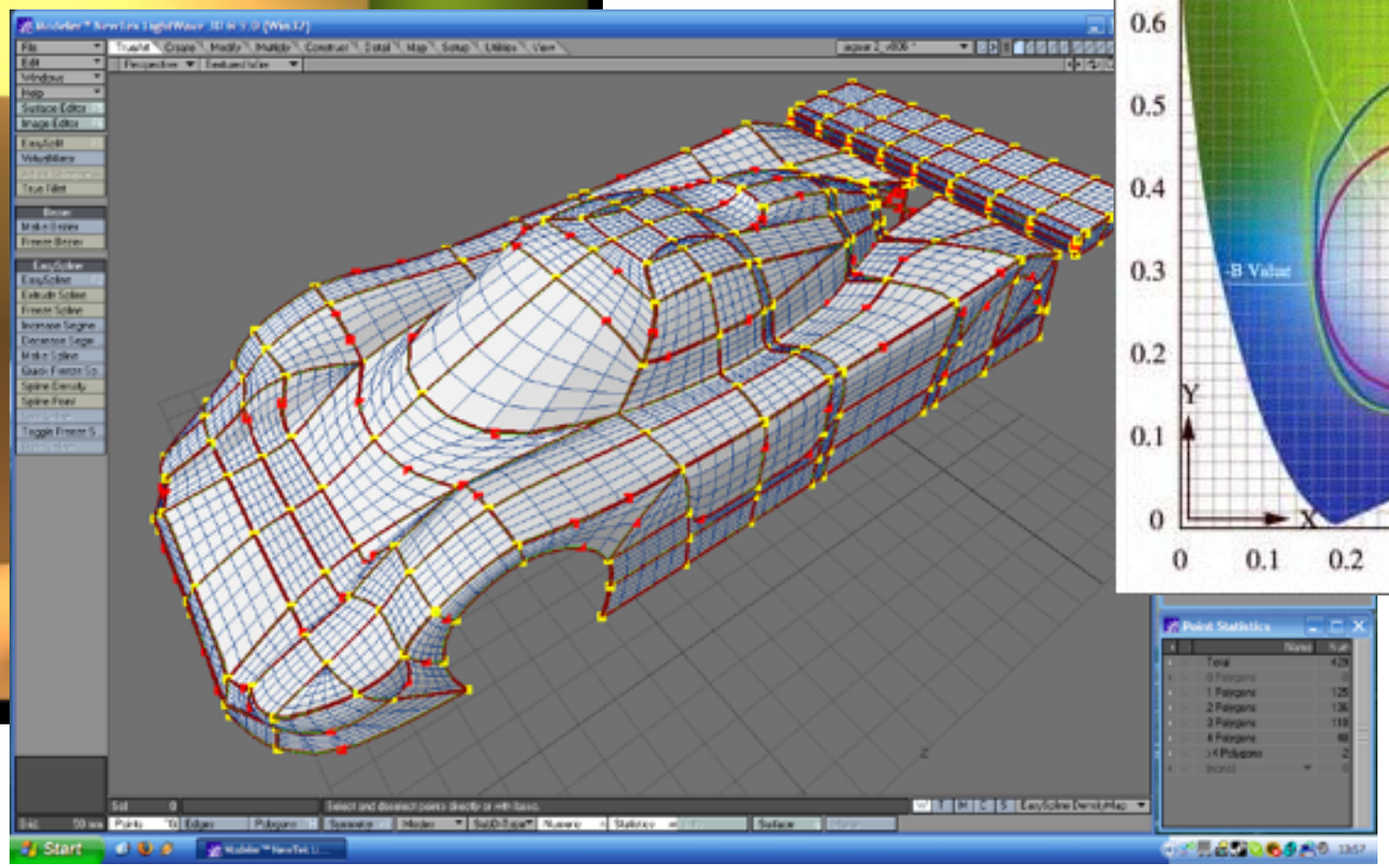
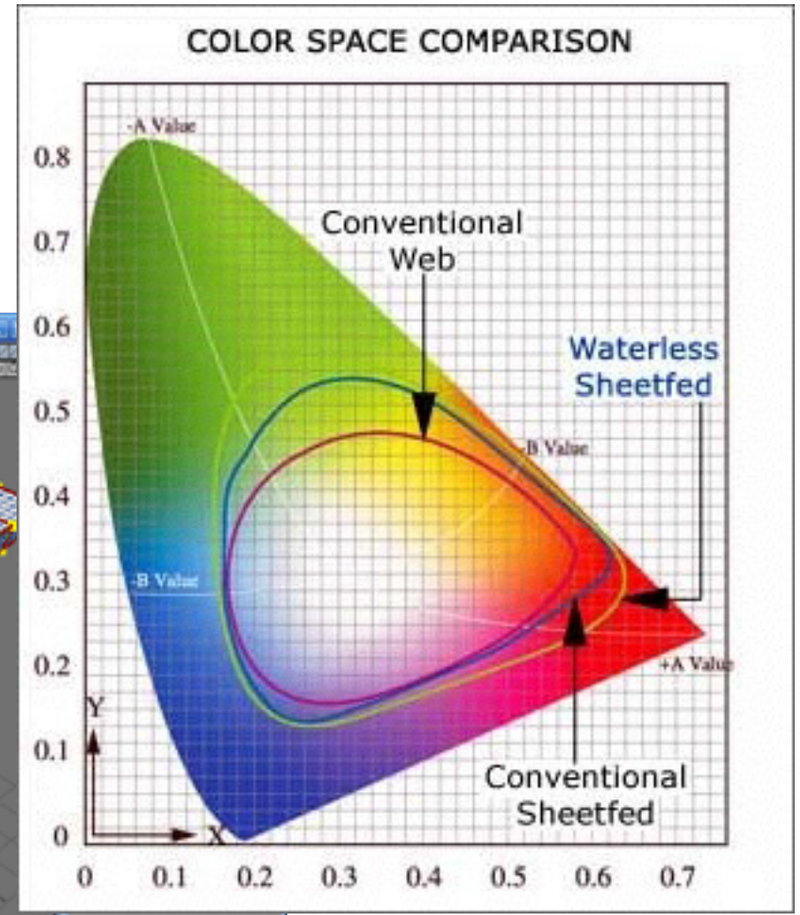






Lecture 6 : Transforms and Hierarchical Modeling in 2D (with an intro to the theory and algebra of transforms)

Tuesday September 26th 2023

Administrative stuff

- Your 2nd Programming Assignment has been posted
  - Due *Saturday Oct 1st (late submission till Oct 4th)*
- The description on canvas also includes some readings
  - Pointers to textbook chapters, can be downloaded via Canvas
  - Consider these as supplements to the lectures (you *should* be able to do fine with information given in lectures)
  - Can be useful for exam review
- Grades for Assignment #1 also posted!

Today's lecture

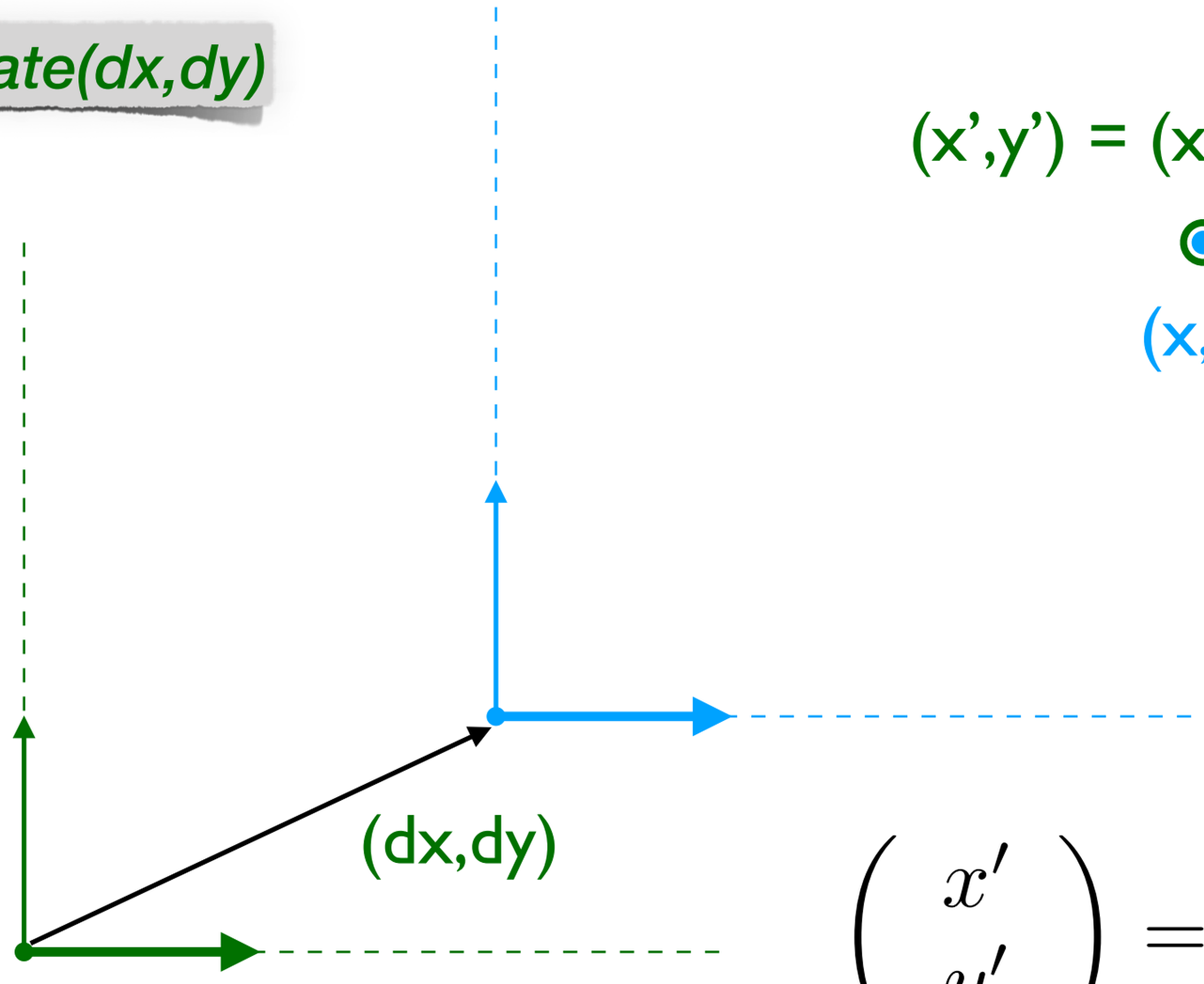
- Mathematical representation of elementary transforms
- Algebra of transform compositions and point/vector transformations
- An introductory look at homogeneous coordinates
- Implementation of hierarchical modeling and transform compositions via linear algebra libraries

Transforms and their semantics

jsbin.com/hozeyar

Week3/Demo0

Translate(dx,dy)



$$(x',y') = (x+dx,y+dy)$$

$$(x,y)$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} dx \\ dy \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} dx \\ dy \end{pmatrix}$$

Transforms and their semantics

jsbin.com/hozeyar

Week3/Demo0

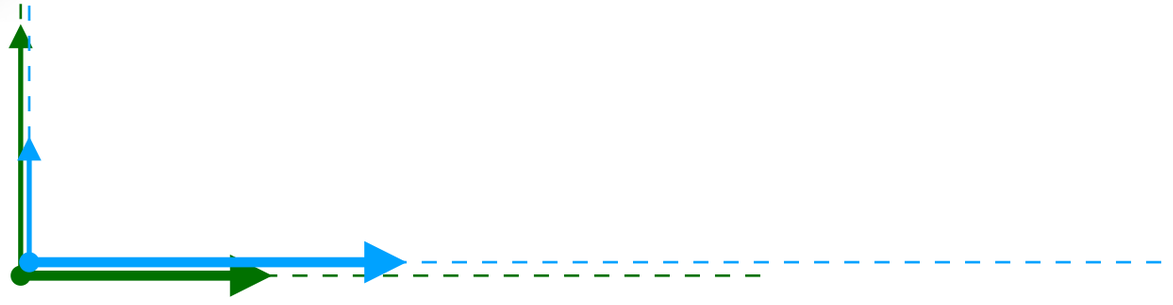
Scale(a,b)

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

(x',y') = (ax,by)

⊙

(x,y)



$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

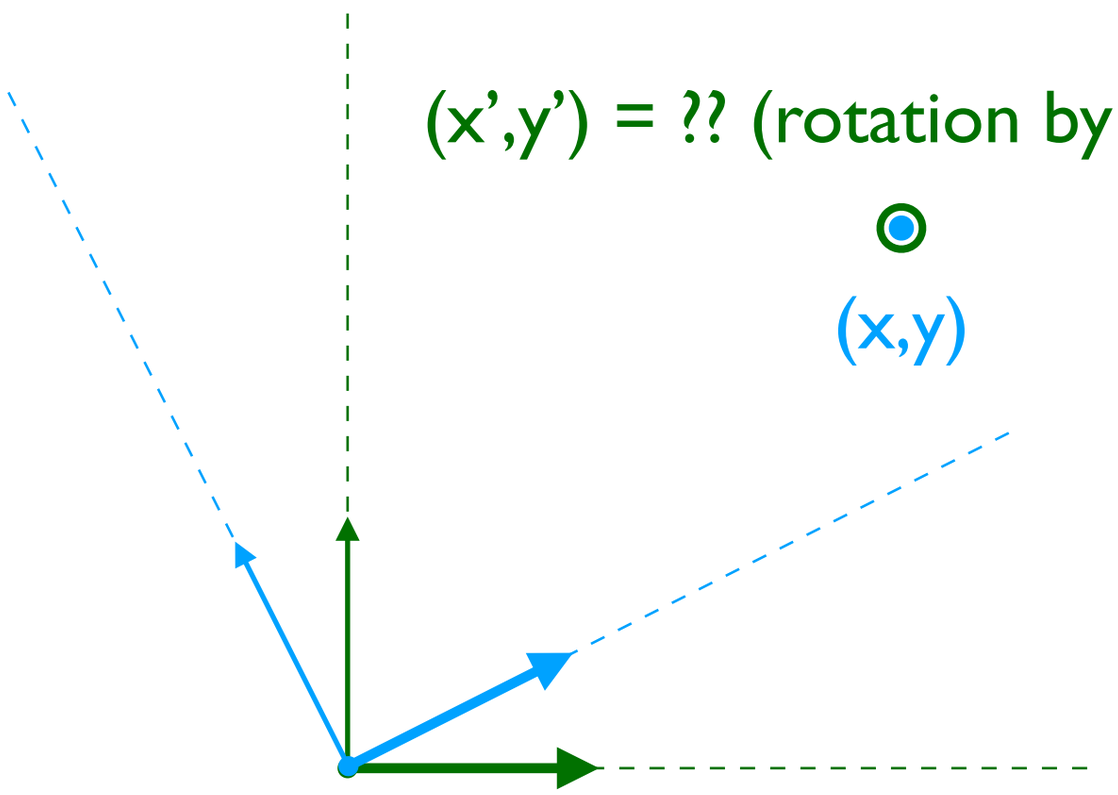
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Transforms and their semantics

jsbin.com/hozeyar

Week3/Demo0

Rotate(θ)



$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Algebra of transforms

Translate(dx,dy)

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} dx \\ dy \end{pmatrix}$$

Scale(a,b)

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Rotate(θ)

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

All three are special cases of linear (actually, affine) mappings

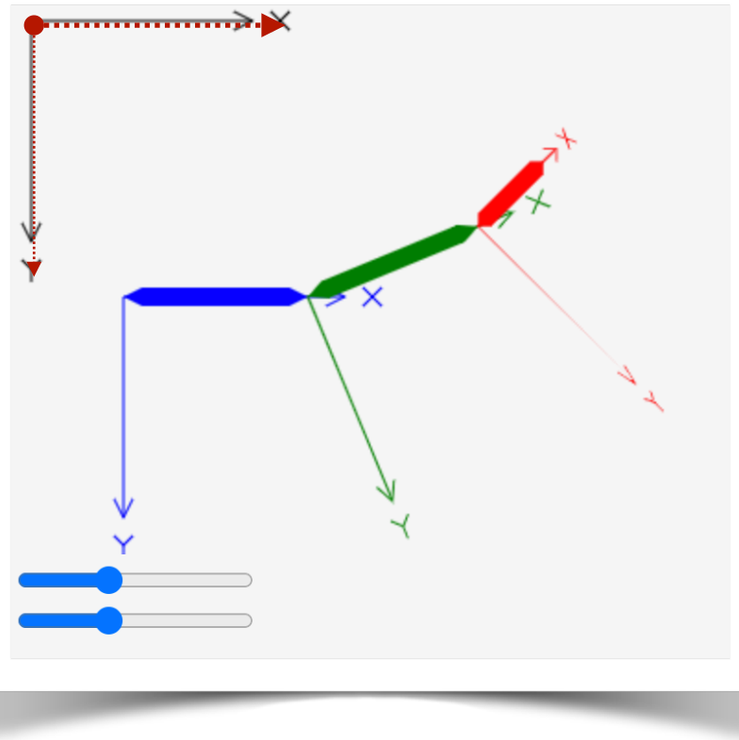
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \mathbf{A} \begin{pmatrix} x \\ y \end{pmatrix} + \mathbf{b}$$

How are transforms combined?

jsbin.com/hozeyar

Week3/Demo0



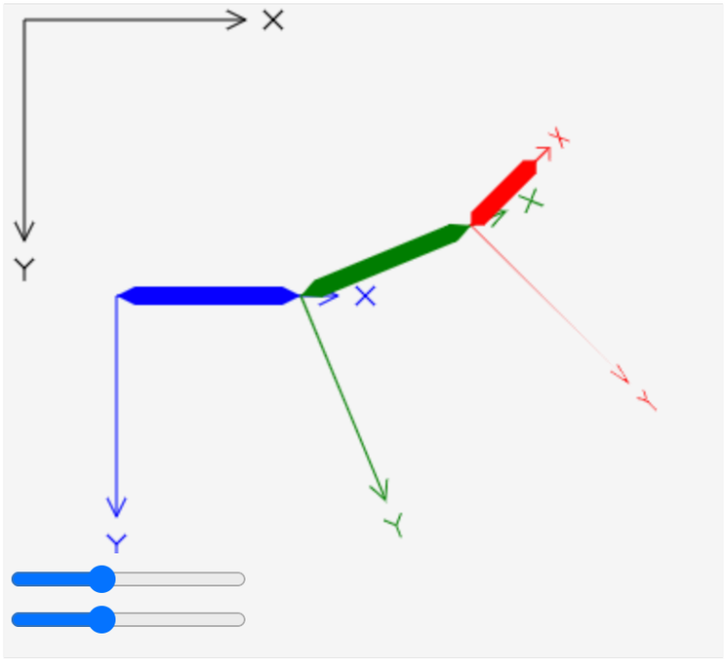
JavaScript

```
[...]

    // 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?

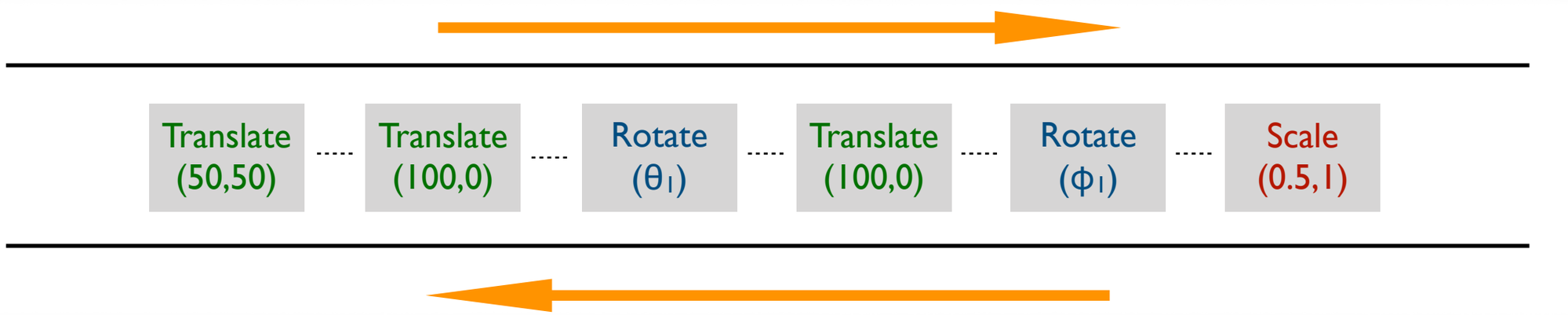
jsbin.com/hozeyar



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");

[...]
```

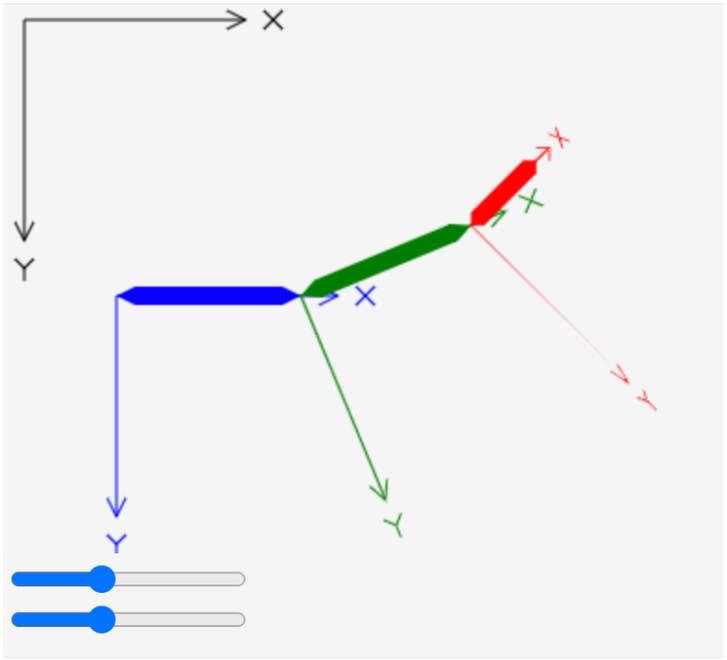
Transforms are introduced in this order
(current coordinate system is obtained by combining transforms left-to-right)



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



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)	$\vec{x}'''$	Translate (100,0)	$\vec{x}''$	Rotate (ϕ_1)	$\vec{x}'$	Scale (0.5,1)	$\vec{x}$

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

Algebra of transforms

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \mathbf{A} \begin{pmatrix} x \\ y \end{pmatrix} + \mathbf{b}$$

$$\vec{x}' = \mathbf{A}_1 \vec{x} + \vec{b}_1$$

$$\vec{x}'' = \mathbf{A}_2 \vec{x}' + \vec{b}_2$$

Two successive transforms

$$\vec{x}'' = \mathbf{A}_2(\mathbf{A}_1 \vec{x} + \vec{b}_1) + \vec{b}_2$$

$$\vec{x}'' = \underbrace{\mathbf{A}_2 \mathbf{A}_1}_{\mathbf{A}_{12}} \vec{x} + \underbrace{\mathbf{A}_2 \vec{b}_1 + \vec{b}_2}_{\vec{b}_{12}}$$

Combined transform is also affine …

$$\vec{x}'' = \mathbf{A}_{12} \vec{x} + \vec{b}_{12}$$

… and admits the same representation!

Algebra of transforms

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

Take this operation …

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

… and rewrite like this!

Algebra of transforms

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

Take this operation …

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

… and rewrite like this!

This is the “homogeneous representation” of the vector (before and after the transformation).
The 3rd coordinate is hard-wired to be equal to 1 (for now)

Algebra of transforms

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

Take this operation …

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

… and rewrite like this!

And this is the homogeneous representation of the transformation (you will see this being referred to as the "transformation matrix"). Applying the transform becomes simply a matrix-vector multiplication with this 3x3 matrix (as opposed to multiplication by 2x2 matrix, and addition of vector)

Algebra of transforms

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \mathbf{T}_1 \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} x'' \\ y'' \\ 1 \end{pmatrix} = \mathbf{T}_2 \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} x'' \\ y'' \\ 1 \end{pmatrix} = \underbrace{\mathbf{T}_2\mathbf{T}_1}_{\mathbf{T}_{12}} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Combining two sequentially applied transforms is a simple matrix multiplication

Algebra of transforms

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \mathbf{T}_1 \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} x'' \\ y'' \\ 1 \end{pmatrix} = \mathbf{T}_2 \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} x'' \\ y'' \\ 1 \end{pmatrix} = \underbrace{\mathbf{T}_2\mathbf{T}_1}_{\mathbf{T}_{12}} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

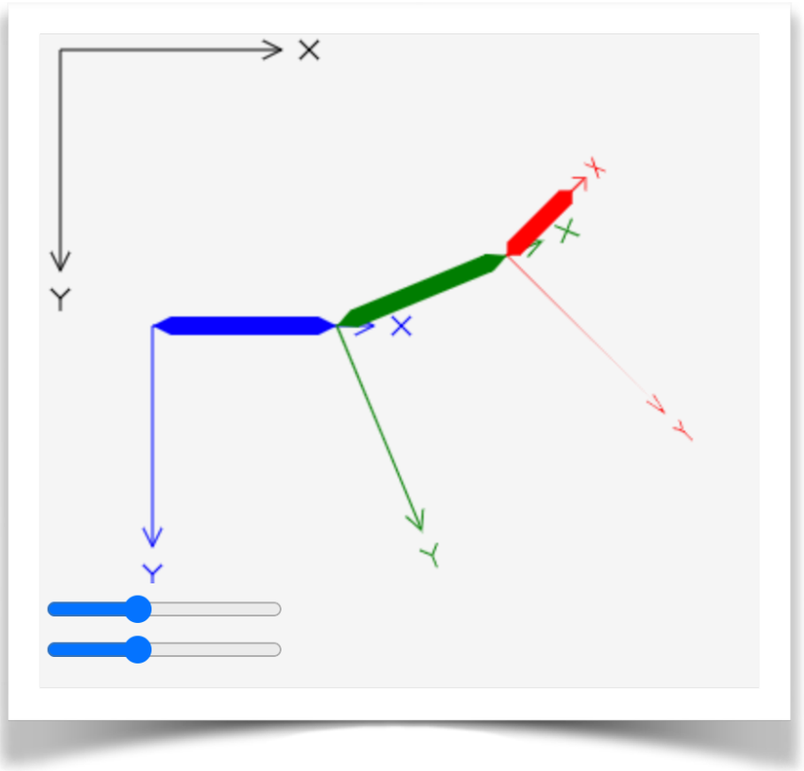
(contrasted with this …)

$$\vec{x}'' = \underbrace{\mathbf{A}_2\mathbf{A}_1}_{\mathbf{A}_{12}}\vec{x} + \underbrace{\mathbf{A}_2\vec{b}_1 + \vec{b}_2}_{\vec{b}_{12}}$$

How are transforms combined?

jsbin.com/hozeyar

Week3/Demo0



And this works with longer transform chains!

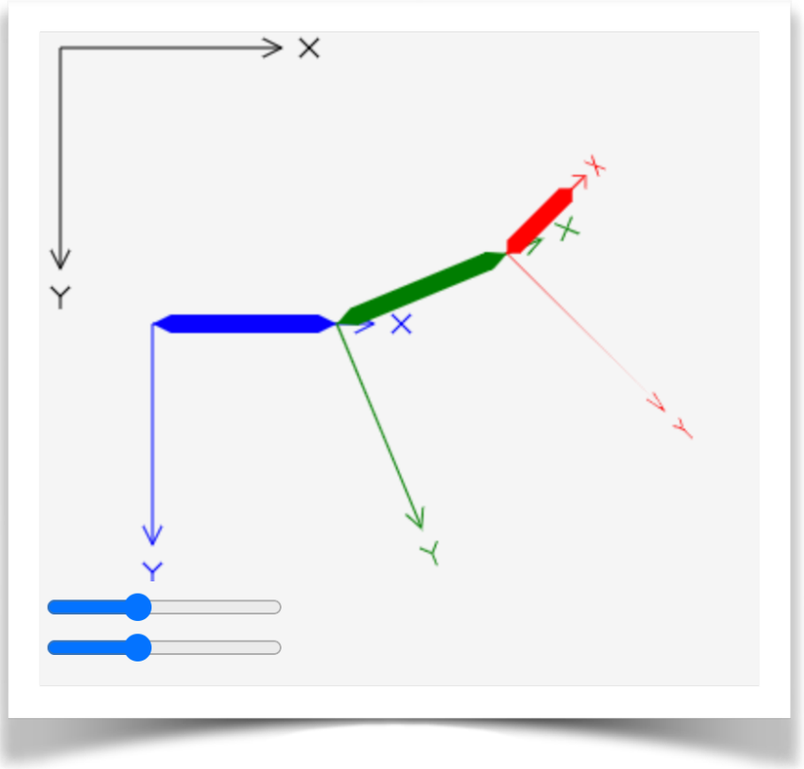
$$\mathbf{T}_6 \quad \cdots \quad \mathbf{T}_2 \quad \mathbf{T}_1$$

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



And this works with longer transform chains!

$$\mathbf{T}_6 \quad \cdots \quad \mathbf{T}_2 \quad \mathbf{T}_1$$

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

$$\mathbf{T}_{\text{combined}} = \mathbf{T}_6 \mathbf{T}_5 \cdots \mathbf{T}_2 \mathbf{T}_1$$

(beware of the order of multiplication!)

Practical implementation

jsbin.com/yifahoq

Week4/Demo0

- We will need to use a linear algebra library
- Ultimately you might need a library with comprehensive functionality, but for now we'll use a simple variant that will get us up-and-running quickly
- Today: we'll see an example using glmatrix
- Thursday: we'll contrast with TWGL (and discuss some diverging notational conventions) and Canvas
- Short term objective: take a deep dive, see this concept working in action; don't worry, we will revisit

Practical implementation

jsbin.com/yifahoq

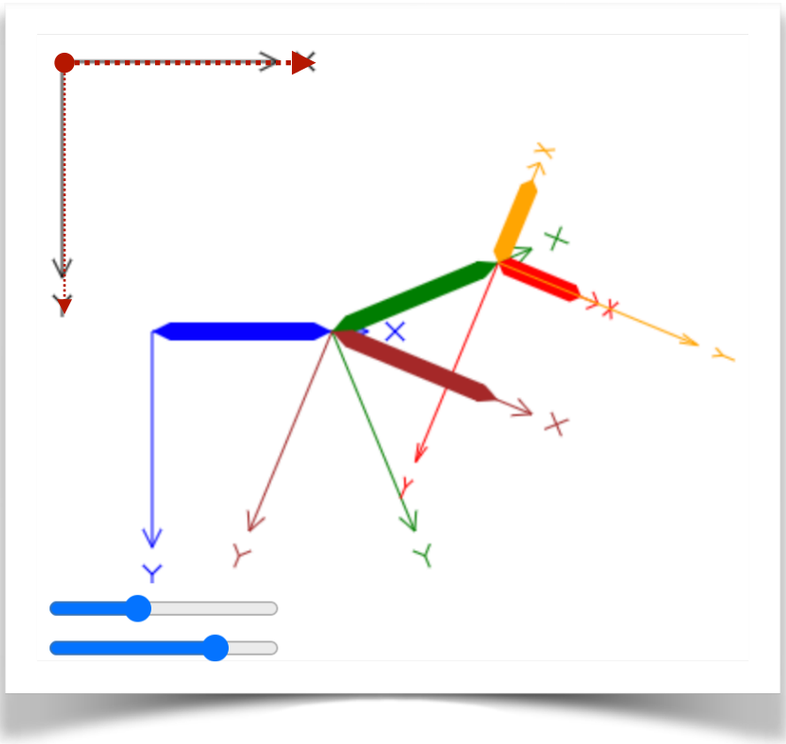
Week4/Demo0

- glMatrix URL : https://glmatrix.net/
- Documentation : https://glmatrix.net/docs/
- Relevant modules (for today) :
  - vec2 : https://glmatrix.net/docs/module-vec2.html
  - mat3 : https://glmatrix.net/docs/module-mat3.html

Transforms in algebraic form (via *glmatrix*)

jsbin.com/yifahoq

Week4/Demo0



This demo illustrates a new paradigm for 2D drawing … instead of relying on Canvas to implement transforms, we take care of the implementation via glMatrix

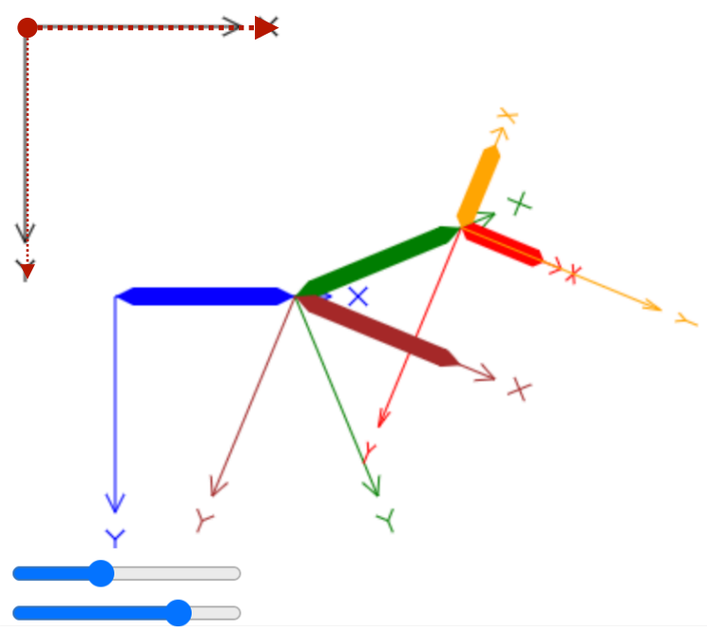
demo.js

```
[...]

    function moveToTx(x,y,Tx)
    {var res=vec2.create(); vec2.transformMat3(res,[x,y],Tx);
     context.moveTo(res[0],res[1]);}

    function lineToTx(x,y,Tx)
    {var res=vec2.create(); vec2.transformMat3(res,[x,y],Tx);
     context.lineTo(res[0],res[1]);}

    function linkage(color,Tx) {
        context.beginPath();
        context.fillStyle = color;
        moveToTx(0,0,Tx);
        lineToTx(10,5,Tx);
        lineToTx(90,5,Tx);
        lineToTx(100,0,Tx);
        lineToTx(90,-5,Tx);
        lineToTx(10,-5,Tx);
        context.closePath();
        context.fill();
        axes(color,Tx);
    }

    function axes(color,Tx) {
        context.strokeStyle=color;
        context.beginPath();
        // Axes
        moveToTx(120,0,Tx);lineToTx(0,0,Tx);lineToTx(0,120,Tx);
        // Arrowheads
        moveToTx(110,5,Tx);lineToTx(120,0,Tx);lineToTx(110,-5,Tx);
        moveToTx(5,110,Tx);lineToTx(0,120,Tx);lineToTx(-5,110,Tx);
        // X-label
        moveToTx(130,-5,Tx);lineToTx(140,5,Tx);
        moveToTx(130,5,Tx);lineToTx(140,-5,Tx);
        // Y-label
        moveToTx(-5,130,Tx);lineToTx(0,135,Tx);lineToTx(5,130,Tx);
        moveToTx(0,135,Tx);lineToTx(0,142,Tx);
        context.stroke();
    }

[...]
```

Transforms in algebraic form (via *glmatrix*)

jsbin.com/yifahoq

Week4/Demo0



moveToTx/lineToTx functions are "wrappers" around the moveto/lineto methods in the Canvas drawing context that are provided an explicit transform as an input parameter (this parameter is in a format that glMatrix creates and knows how to apply)

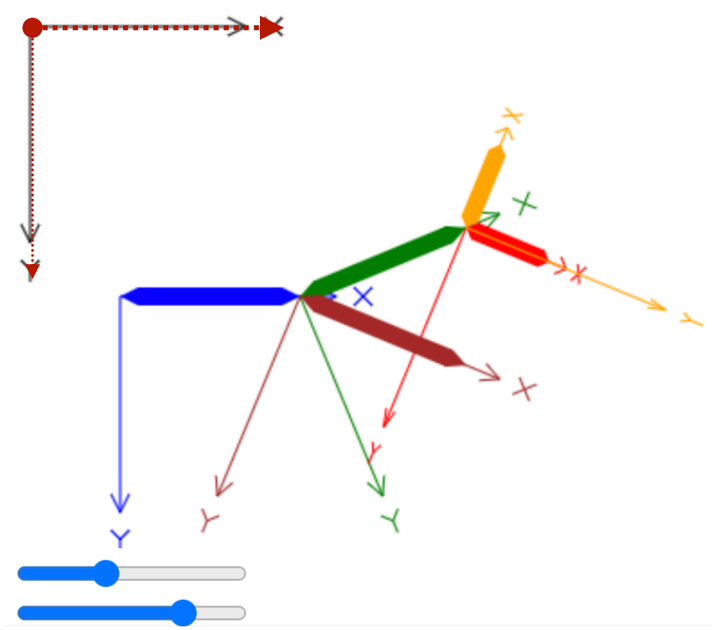
demo.js

```
[...]

   function moveToTx(x,y,Tx)
   {var res=vec2.create(); vec2.transformMat3(res,[x,y],Tx);
    context.moveTo(res[0],res[1]);}

   function lineToTx(x,y,Tx)
   {var res=vec2.create(); vec2.transformMat3(res,[x,y],Tx);
    context.lineTo(res[0],res[1]);}

   function linkage(color,Tx) {
       context.beginPath();
       context.fillStyle = color;
       moveToTx(0,0,Tx);
       lineToTx(10,5,Tx);
       lineToTx(90,5,Tx);
       lineToTx(100,0,Tx);
       lineToTx(90,-5,Tx);
       lineToTx(10,-5,Tx);
       context.closePath();
       context.fill();
       axes(color,Tx);
   }

   ion axes(color,Tx) {
   ontext.strokeStyle=color;
   ontext.beginPath();
   / Axes
   oveToTx(120,0,Tx);lineToTx(0,0,Tx);lineToTx(0,120,Tx);
   / Arrowheads
   oveToTx(110,5,Tx);lineToTx(120,0,Tx);lineToTx(110,-5,Tx);
   oveToTx(5,110,Tx);lineToTx(0,120,Tx);lineToTx(-5,110,Tx);
   / X-label
   oveToTx(130,-5,Tx);lineToTx(140,5,Tx);
   oveToTx(130,5,Tx);lineToTx(140,-5,Tx);
   / Y-label
   oveToTx(-5,130,Tx);lineToTx(0,135,Tx);lineToTx(5,130,Tx);
   oveToTx(0,135,Tx);lineToTx(0,142,Tx);
   context.stroke();
   }

[...]
```

Transforms in algebraic form (via *glmatrix*)

jsbin.com/yifahoq

Week4/Demo0



We apply the input transformation explicitly via a call to the glMatrix transformMat3() method within the vec2 module (vec2 documentation)

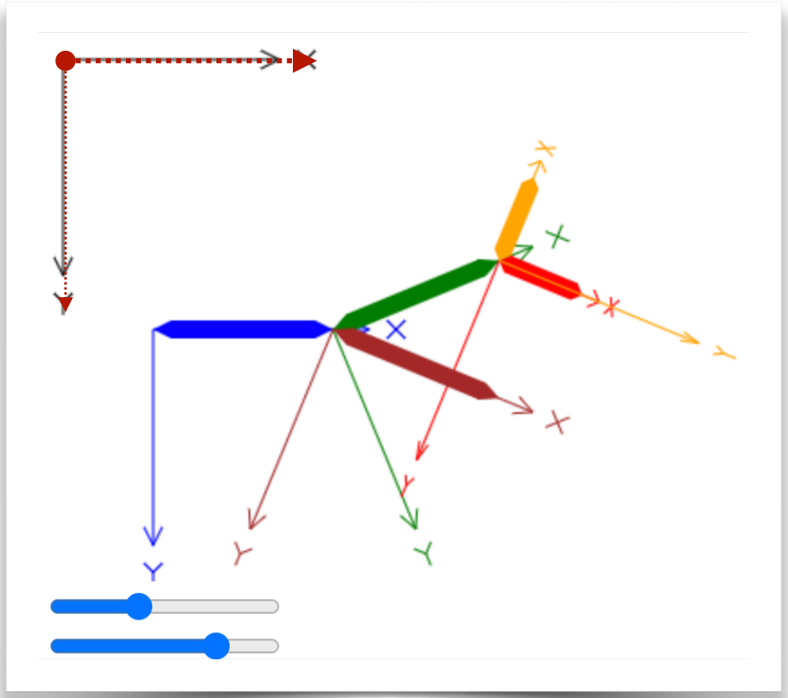
demo.js

```
[...]

    function moveToTx(x,y,Tx)
    {var res=vec2.create(); vec2.transformMat3(res,[x,y],Tx);
     context.moveTo(res[0],res[1]);}

    function lineToTx(x,y,Tx)
    {var res=vec2.create(); vec2.transformMat3(res,[x,y],Tx);
     context.lineTo(res[0],res[1]);}

    function linkage(color,Tx) {
        context.beginPath();
        context.fillStyle = color;
        moveToTx(0,0,Tx);
        lineToTx(10,5,Tx);
        lineToTx(90,5,Tx);
        lineToTx(100,0,Tx);
        lineToTx(90,-5,Tx);
        lineToTx(10,-5,Tx);
        context.closePath();
        context.fill();
        axes(color,Tx);
    }

    function axes(color,Tx) {
        context.strokeStyle=color;
        context.beginPath();
        // Axes
        moveToTx(120,0,Tx);lineToTx(0,0,Tx);lineToTx(0,120,Tx);
        // Arrowheads
        moveToTx(110,5,Tx);lineToTx(120,0,Tx);lineToTx(110,-5,Tx);
        moveToTx(5,110,Tx);lineToTx(0,120,Tx);lineToTx(-5,110,Tx);
        // X-label
        moveToTx(130,-5,Tx);lineToTx(140,5,Tx);
        moveToTx(130,5,Tx);lineToTx(140,-5,Tx);
        // Y-label
        moveToTx(-5,130,Tx);lineToTx(0,135,Tx);lineToTx(5,130,Tx);
        moveToTx(0,135,Tx);lineToTx(0,142,Tx);
        context.stroke();
    }

[...]
```

Transforms in algebraic form (via *glmatrix*)

jsbin.com/yifahoq

Week4/Demo0



Functions linkage() and axes() take the input transform ("Tx") as input, and call the transform-specific drawing calls to perform their task.

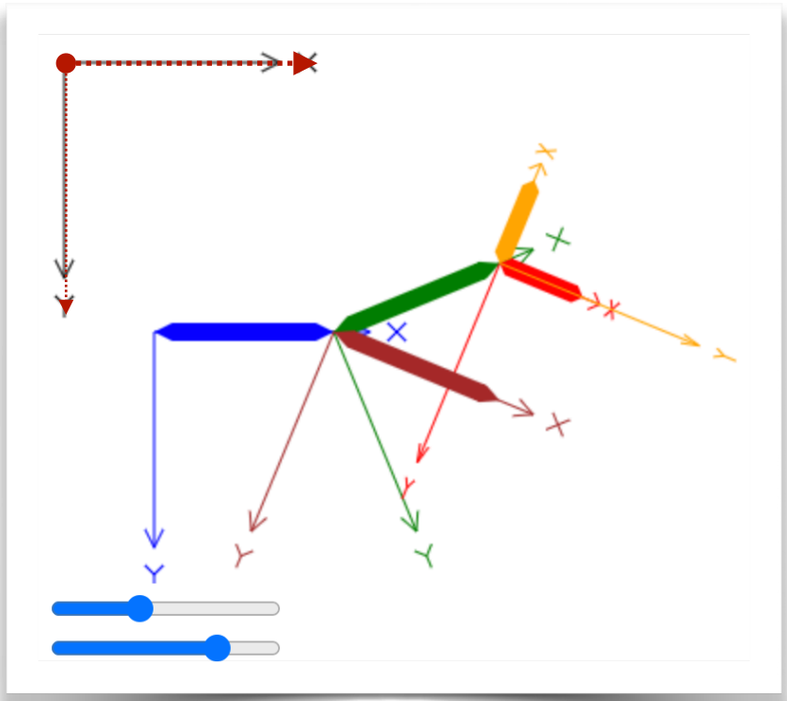
demo.js

```
[...]

    function moveToTx(x,y,Tx)
    {var res=vec2.create(); vec2.transformMat3(res,[x,y],Tx);
     context.moveTo(res[0],res[1]);}

    function lineToTx(x,y,Tx)
    {var res=vec2.create(); vec2.transformMat3(res,[x,y],Tx);
     context.lineTo(res[0],res[1]);}

    function linkage(color,Tx) {
        context.beginPath();
        context.fillStyle = color;
        moveToTx(0,0,Tx);
        lineToTx(10,5,Tx);
        lineToTx(90,5,Tx);
        lineToTx(100,0,Tx);
        lineToTx(90,-5,Tx);
        lineToTx(10,-5,Tx);
        context.closePath();
        context.fill();
        axes(color,Tx);
    }

    function axes(color,Tx) {
        context.strokeStyle=color;
        context.beginPath();
        // Axes
        moveToTx(120,0,Tx);lineToTx(0,0,Tx);lineToTx(0,120,Tx);
        // Arrowheads
        moveToTx(110,5,Tx);lineToTx(120,0,Tx);lineToTx(110,-5,Tx);
        moveToTx(5,110,Tx);lineToTx(0,120,Tx);lineToTx(-5,110,Tx);
        // X-label
        moveToTx(130,-5,Tx);lineToTx(140,5,Tx);
        moveToTx(130,5,Tx);lineToTx(140,-5,Tx);
        // Y-label
        moveToTx(-5,130,Tx);lineToTx(0,135,Tx);lineToTx(5,130,Tx);
        moveToTx(0,135,Tx);lineToTx(0,142,Tx);
        context.stroke();
    }

[...]
```

Transforms in algebraic form (via *glmatrix*)

jsbin.com/yifahoq

Week4/Demo0



demo.js

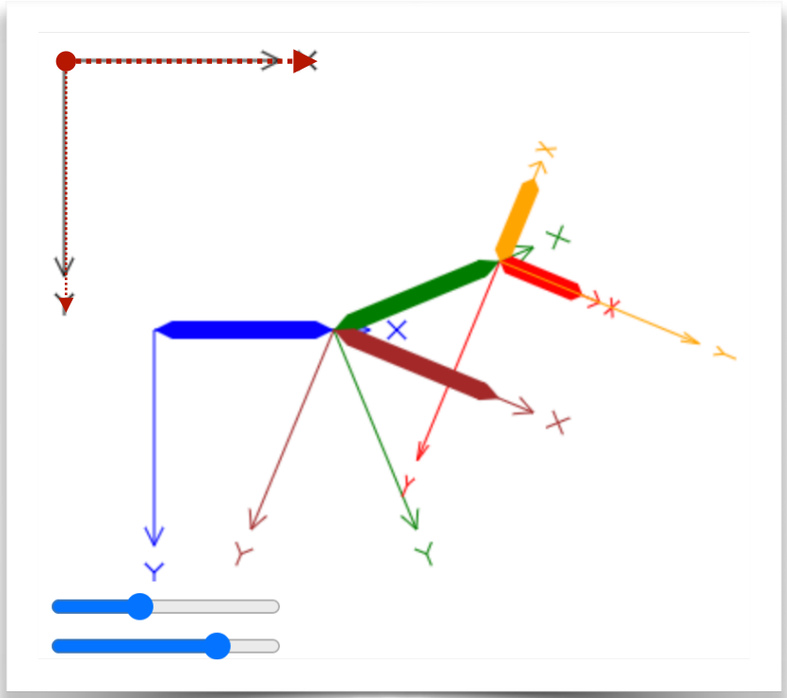
```
[...]

    var Tblue_to_canvas = mat3.create();
    mat3.fromTranslation(Tblue_to_canvas,[50,150]);
    linkage("blue",Tblue_to_canvas);

    var Tgreen_to_blue = mat3.create();
    mat3.fromTranslation(Tgreen_to_blue,[100,0]);
    mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
    var Tgreen_to_canvas = mat3.create();
    mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
    linkage("green",Tgreen_to_canvas);

    var Tred_to_green = mat3.create();
    mat3.fromTranslation(Tred_to_green,[100,0]);
    mat3.rotate(Tred_to_green,Tred_to_green,phi1);
    mat3.scale(Tred_to_green,Tred_to_green,[0.5,1]);
    var Tred_to_canvas = mat3.create();
    mat3.multiply(Tred_to_canvas,Tgreen_to_canvas,Tred_to_green);
    linkage("red",Tred_to_canvas);

[...]
```

(no transforms yet!)

Transforms in algebraic form (via *glmatrix*)

jsbin.com/yifahoq

Week4/Demo0



demo.js

```
[...]

  var Tblue_to_canvas = mat3.create();
  mat3.fromTranslation(Tblue_to_canvas,[50,150]);
  linkage("blue",Tblue_to_canvas);

  var Tgreen_to_blue = mat3.create();
  mat3.fromTranslation(Tgreen_to_blue,[100,0]);
  mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
  var Tgreen_to_canvas = mat3.create();
  mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
  linkage("green",Tgreen_to_canvas);

  var Tred_to_green = mat3.create();
  mat3.fromTranslation(Tred_to_green,[100,0]);
  mat3.rotate(Tred_to_green,Tred_to_green,phi1);
  mat3.scale(Tred_to_green,Tred_to_green,[0.5,1]);
  var Tred_to_canvas = mat3.create();
  mat3.multiply(Tred_to_canvas,Tgreen_to_canvas,Tred_to_green);
  linkage("red",Tred_to_canvas);
```

$$\mathbf{T}_{\text{blue2canvas}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

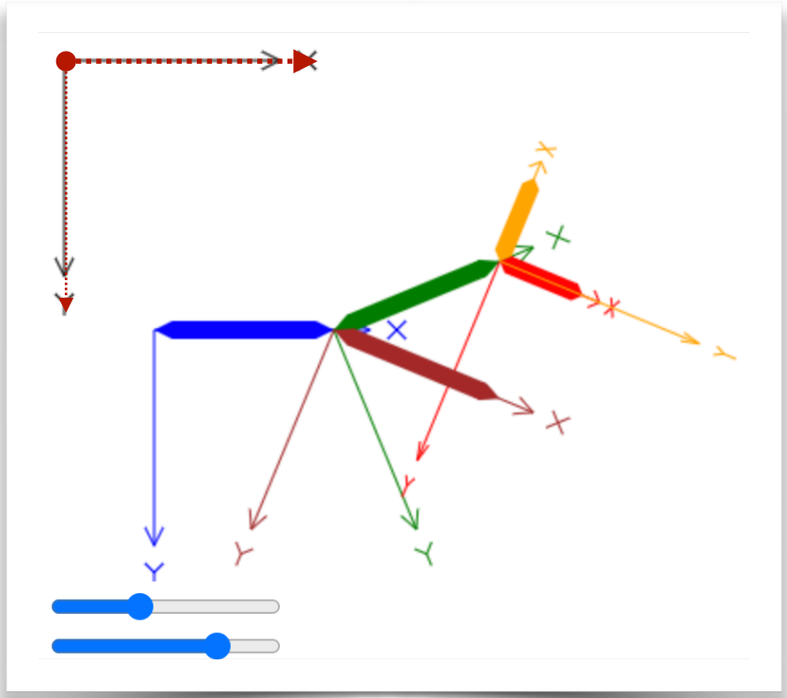
Translate (50,150)

blue-to-canvas

Transforms in algebraic form (via *glmatrix*)

jsbin.com/yifahoq

Week4/Demo0



demo.js

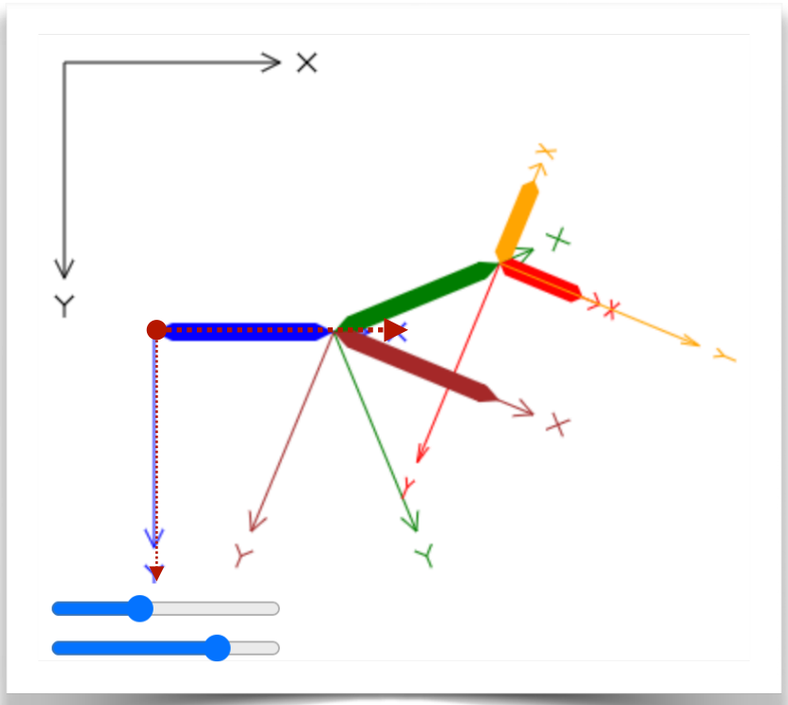
```
[...]

    var Tblue_to_canvas = mat3.create();
    mat3.fromTranslation(Tblue_to_canvas,[50,150]);
    linkage("blue",Tblue_to_canvas);

    var Tgreen_to_blue = mat3.create();
    mat3.fromTranslation(Tgreen_to_blue,[100,0]);
    mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
    var Tgreen_to_canvas = mat3.create();
    mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
    linkage("green",Tgreen_to_canvas);

    var Tred_to_green = mat3.create();
    mat3.fromTranslation(Tred_to_green,[100,0]);
    mat3.rotate(Tred_to_green,Tred_to_green,phi1);
    mat3.scale(Tred_to_green,Tred_to_green,[0.5,1]);
    var Tred_to_canvas = mat3.create();
    mat3.multiply(Tred_to_canvas,Tgreen_to_canvas,Tred_to_green);
    linkage("red",Tred_to_canvas);
```

$$\mathbf{T}_{\text{blue2canvas}} = \begin{pmatrix} 1 & 0 & 50 \\ 0 & 1 & 150 \\ 0 & 0 & 1 \end{pmatrix}$$

Transforms in algebraic form (via *glmatrix*)

jsbin.com/yifahoq

Week4/Demo0



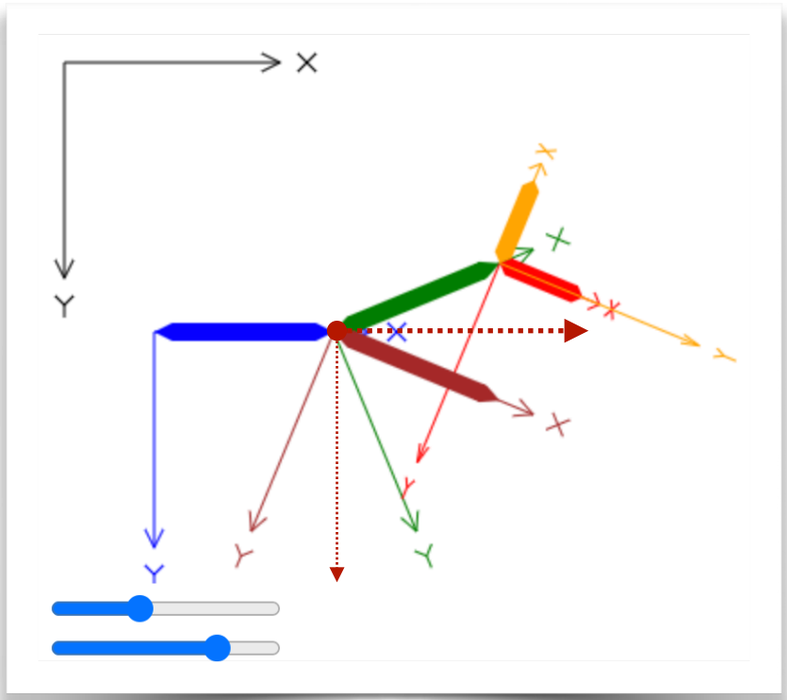
demo.js

```
[...]

  var Tblue_to_canvas = mat3.create();
  mat3.fromTranslation(Tblue_to_canvas,[50,150]);
  linkage("blue",Tblue_to_canvas);

  var Tgreen_to_blue = mat3.create();
  mat3.fromTranslation(Tgreen_to_blue,[100,0]);
  mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
  var Tgreen_to_canvas = mat3.create();
  mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
  linkage("green",Tgreen_to_canvas);

  var Tred_to_green = mat3.create();
  mat3.fromTranslation(Tred_to_green,[100,0]);
  mat3.rotate(Tred_to_green,Tred_to_green,phi1);
  mat3.scale(Tred_to_green,Tred_to_green,[0.5,1]);
  var Tred_to_canvas = mat3.create();
  mat3.multiply(Tred_to_canvas,Tgreen_to_canvas,Tred_to_green);
  linkage("red",Tred_to_canvas);

[...]
```

Transforms in algebraic form (via *glmatrix*)

jsbin.com/yifahoq

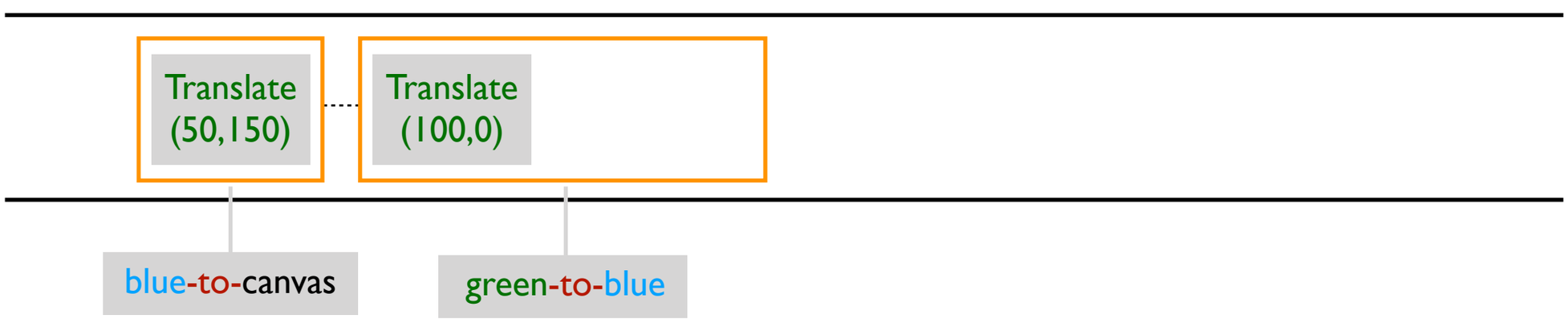
Week4/Demo0



$$\mathbf{T}_{\text{green2blue}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

demo.js

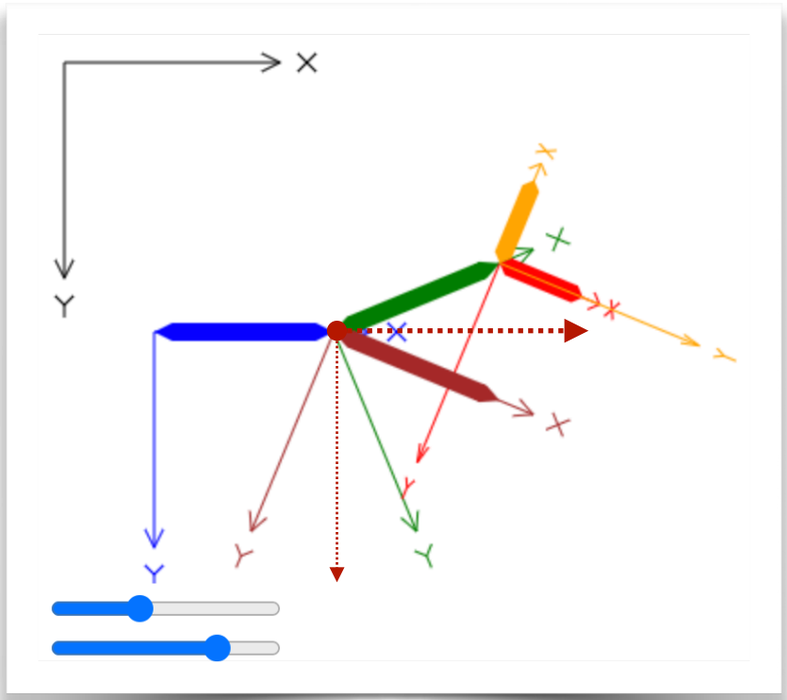
```
[...]

    var Tblue_to_canvas = mat3.create();
    mat3.fromTranslation(Tblue_to_canvas,[50,150]);
    linkage("blue",Tblue_to_canvas);

    var Tgreen_to_blue = mat3.create();
    mat3.fromTranslation(Tgreen_to_blue,[100,0]);
    mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
    var Tgreen_to_canvas = mat3.create();
    mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
    linkage("green",Tgreen_to_canvas);

    var Tred_to_green = mat3.create();
    mat3.fromTranslation(Tred_to_green,[100,0]);
    (Tred_to_green,Tred_to_green,phi1);
    Tred_to_green,Tred_to_green,[0.5,1]);
    _canvas = mat3.create();
    ly(Tred_to_canvas,Tgreen_to_canvas,Tred_to_green);
    d",Tred_to_canvas);
```



Transforms in algebraic form (via *glmatrix*)

jsbin.com/yifahoq

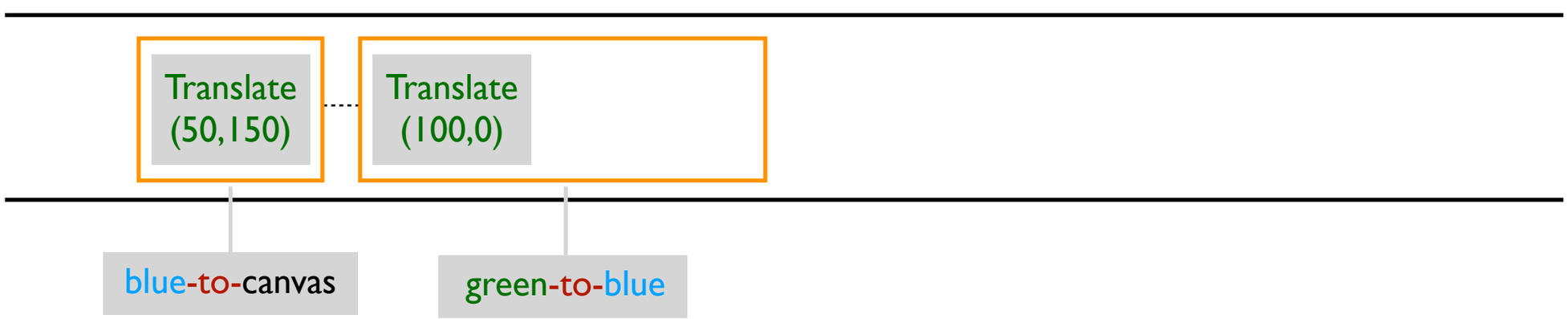
Week4/Demo0



$$\mathbf{T}_{\text{green2blue}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 100 \\ 0 & 0 & 1 \end{pmatrix}$$

demo.js

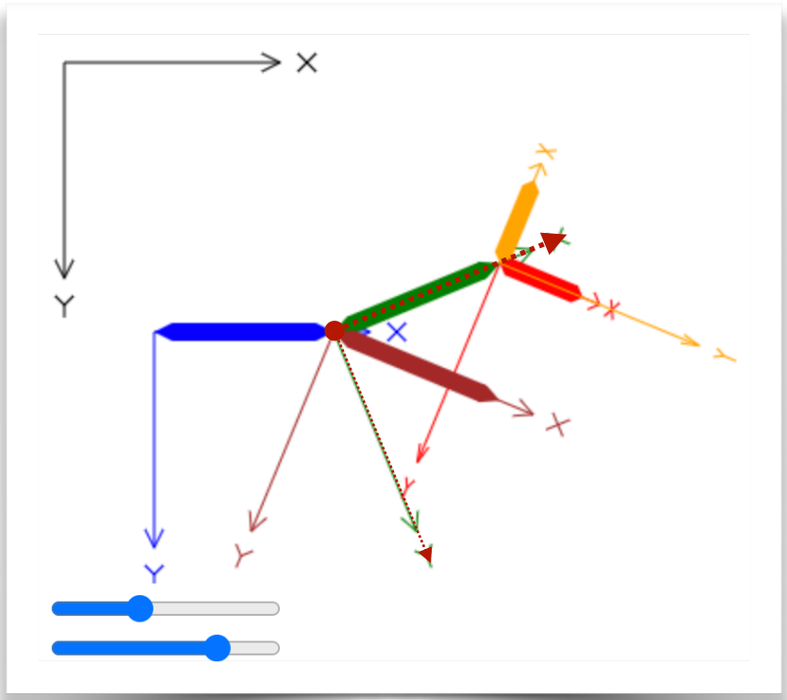
```
[...]

    var Tblue_to_canvas = mat3.create();
    mat3.fromTranslation(Tblue_to_canvas,[50,150]);
    linkage("blue",Tblue_to_canvas);

    var Tgreen_to_blue = mat3.create();
    mat3.fromTranslation(Tgreen_to_blue,[100,0]);
    mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
    var Tgreen_to_canvas = mat3.create();
    mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
    linkage("green",Tgreen_to_canvas);

    var Tred_to_green = mat3.create();
    mat3.fromTranslation(Tred_to_green,[100,0]);
    mat3.rotate(Tred_to_green,Tred_to_green,phi1);
    mat3.fromTranslation(Tred_to_green,Tred_to_green,[0.5,1]);
    var Tred_to_canvas = mat3.create();
    mat3.multiply(Tred_to_canvas,Tgreen_to_canvas,Tred_to_green);
    linkage("red",Tred_to_canvas);
```



Transforms in algebraic form (via *glmatrix*)

jsbin.com/yifahoq

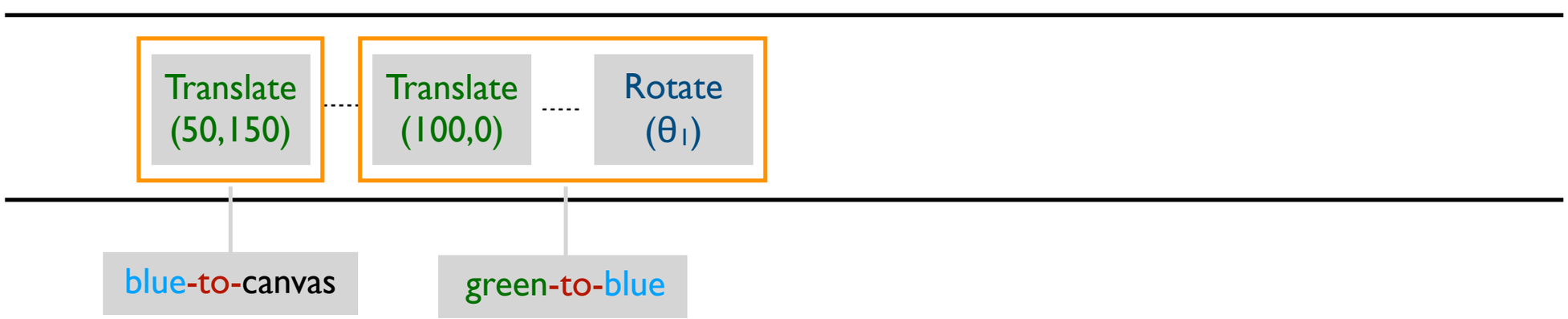
Week4/Demo0



$$\mathbf{T}_{\text{green2blue}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 100 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

demo.js

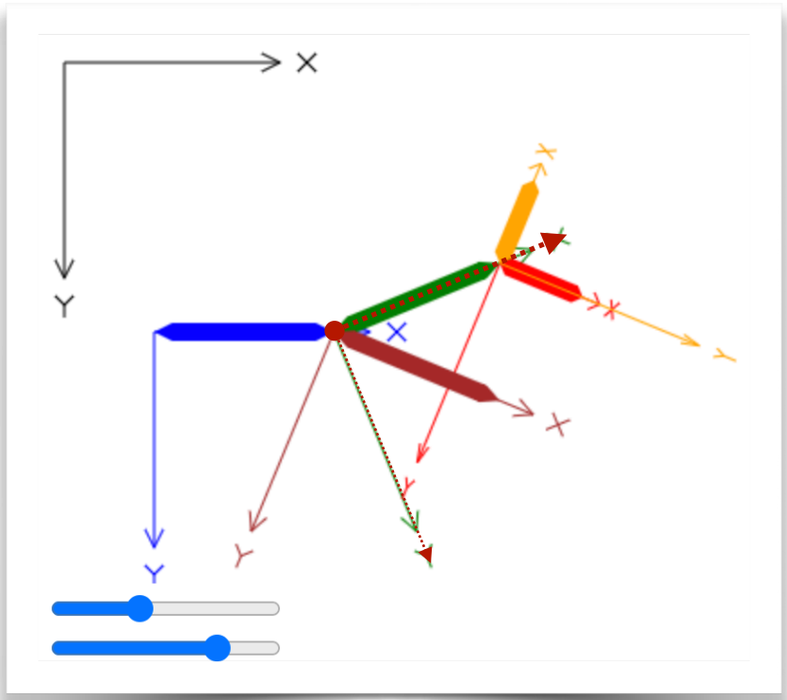
```
[...]

    var Tblue_to_canvas = mat3.create();
    mat3.fromTranslation(Tblue_to_canvas,[50,150]);
    linkage("blue",Tblue_to_canvas);

    var Tgreen_to_blue = mat3.create();
    mat3.fromTranslation(Tgreen_to_blue,[100,0]);
    mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
    var Tgreen_to_canvas = mat3.create();
    mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
    linkage("green",Tgreen_to_canvas);

    var Tred_to_green = mat3.create();
    mat3.fromTranslation(Tred_to_green,[100,0]);
    (Tred_to_green,Tred_to_green,phi1);
    Tred_to_green,Tred_to_green,[0.5,1]);
    _canvas = mat3.create();
    ly(Tred_to_canvas,Tgreen_to_canvas,Tred_to_green);
    d",Tred_to_canvas);
```



Transforms in algebraic form (via *glmatrix*)

jsbin.com/yifahoq

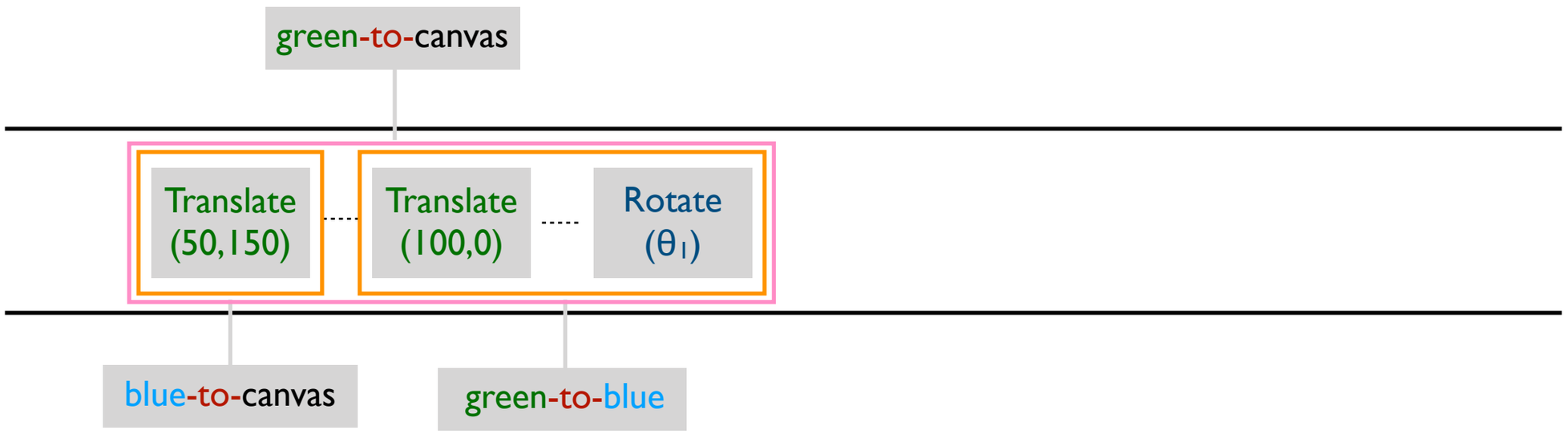
Week4/Demo0



$$\mathbf{T}_{\text{green2canvas}} = \mathbf{T}_{\text{blue2canvas}}\mathbf{T}_{\text{green2blue}}$$

demo.js

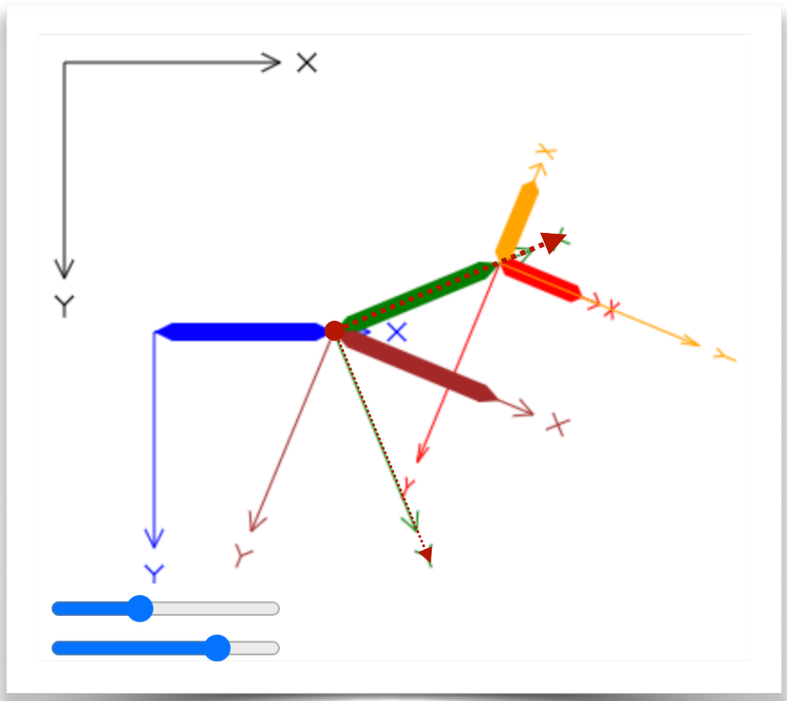
```
[...]

    var Tblue_to_canvas = mat3.create();
    mat3.fromTranslation(Tblue_to_canvas,[50,150]);
    linkage("blue",Tblue_to_canvas);

    var Tgreen_to_blue = mat3.create();
    mat3.fromTranslation(Tgreen_to_blue,[100,0]);
    mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
    var Tgreen_to_canvas = mat3.create();
    mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
    linkage("green",Tgreen_to_canvas);

    var Tred_to_green = mat3.create();
    mat3.fromTranslation(Tred_to_green,[100,0]);
(Tred_to_green,Tred_to_green,phi1);
Tred_to_green,Tred_to_green,[0.5,1]);
_canvas = mat3.create();
ly(Tred_to_canvas,Tgreen_to_canvas,Tred_to_green);
ed",Tred_to_canvas);
```



Transforms in algebraic form (via *glmatrix*)

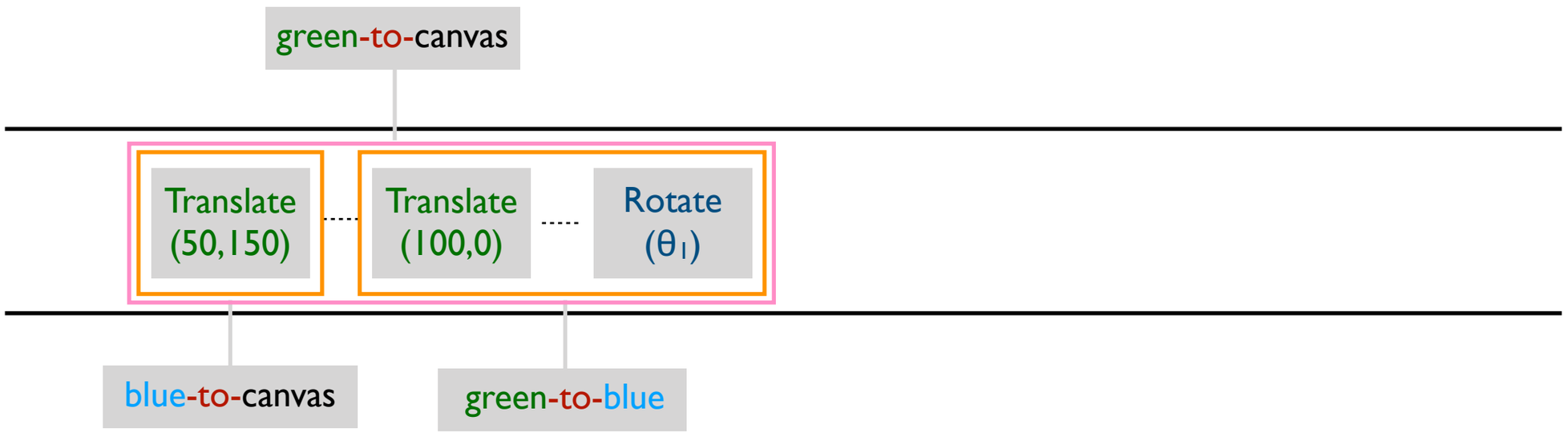
jsbin.com/yifahoq

Week4/Demo0



demo.js

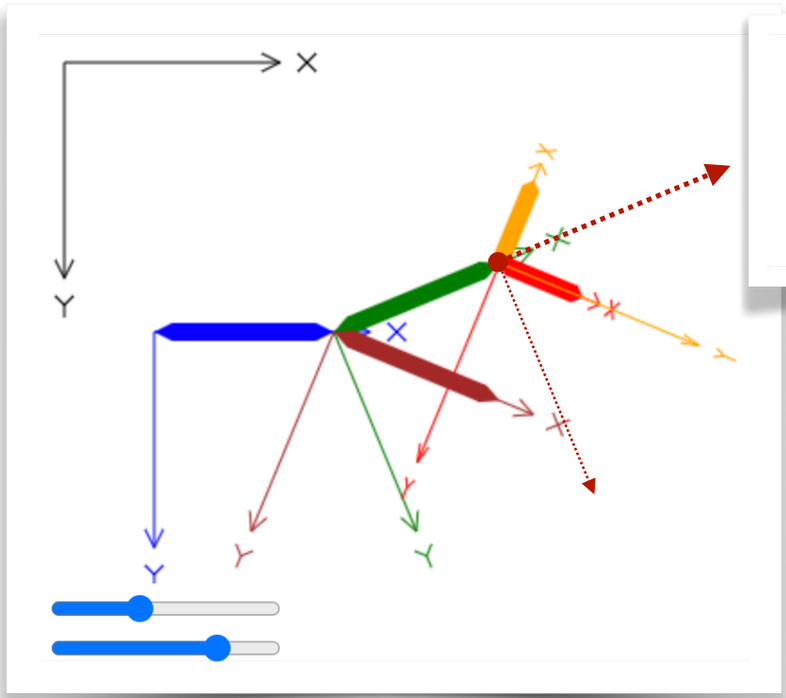
```
[...]

    var Tblue_to_canvas = mat3.create();
    mat3.fromTranslation(Tblue_to_canvas,[50,150]);
    linkage("blue",Tblue_to_canvas);

    var Tgreen_to_blue = mat3.create();
    mat3.fromTranslation(Tgreen_to_blue,[100,0]);
    mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
    var Tgreen_to_canvas = mat3.create();
    mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
    linkage("green",Tgreen_to_canvas);

    var Tred_to_green = mat3.create();
    mat3.fromTranslation(Tred_to_green,[100,0]);
    mat3.rotate(Tred_to_green,Tred_to_green,phi1);
    mat3.scale(Tred_to_green,Tred_to_green,[0.5,1]);
    var Tred_to_canvas = mat3.create();
    mat3.multiply(Tred_to_canvas,Tgreen_to_canvas,Tred_to_green);
    linkage("red",Tred_to_canvas);

[...]
```



Transforms in algebraic form (via *glmatrix*)

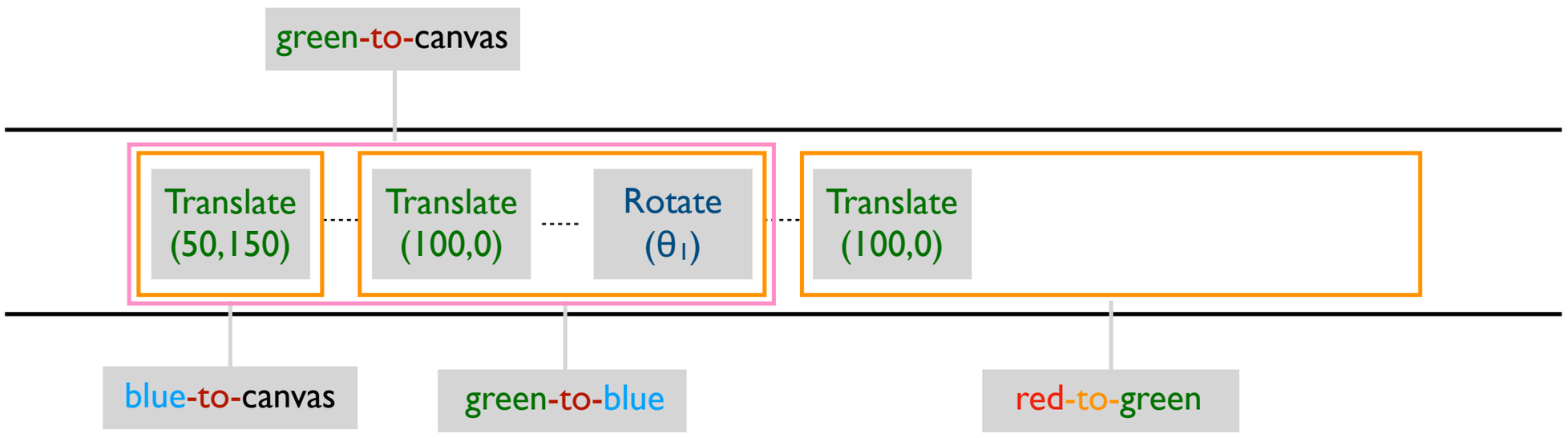
jsbin.com/yifahoq

Week4/Demo0



$$\mathbf{T}_{\text{red2green}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

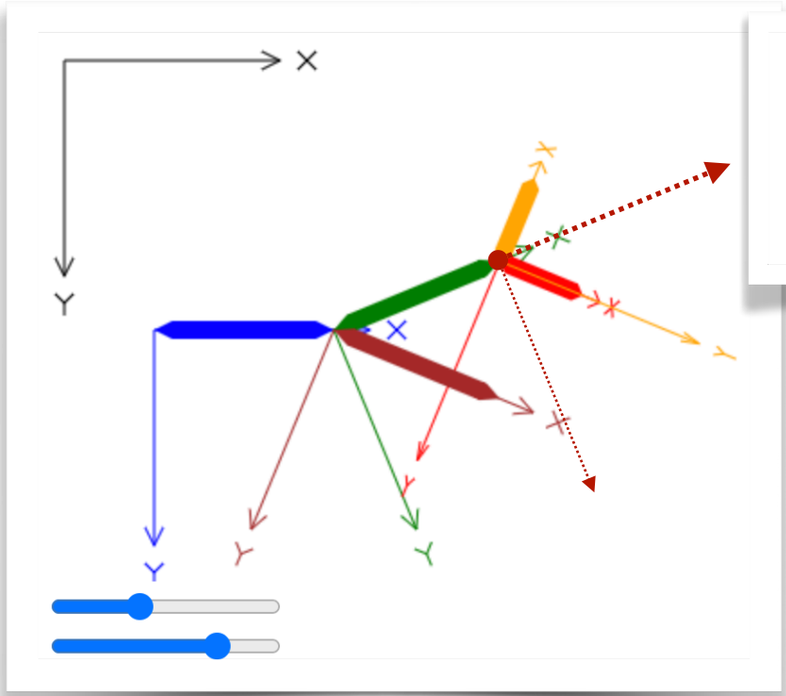
```
mat3.fromTranslation(Tblue_to_canvas,[50,150]);
linkage("blue",Tblue_to_canvas);

var Tgreen_to_blue = mat3.create();
mat3.fromTranslation(Tgreen_to_blue,[100,0]);
mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
var Tgreen_to_canvas = mat3.create();
mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
linkage("green",Tgreen_to_canvas);

var Tred_to_green = mat3.create();
mat3.fromTranslation(Tred_to_green,[100,0]);
mat3.rotate(Tred_to_green,Tred_to_green,phi1);
mat3.scale(Tred_to_green,Tred_to_green,[0.5,1]);
var Tred_to_canvas = mat3.create();
mat3.multiply(Tred_to_canvas,Tgreen_to_canvas,Tred_to_green);
linkage("red",Tred_to_canvas);

[...]
```



Transforms in algebraic form (via *glmatrix*)

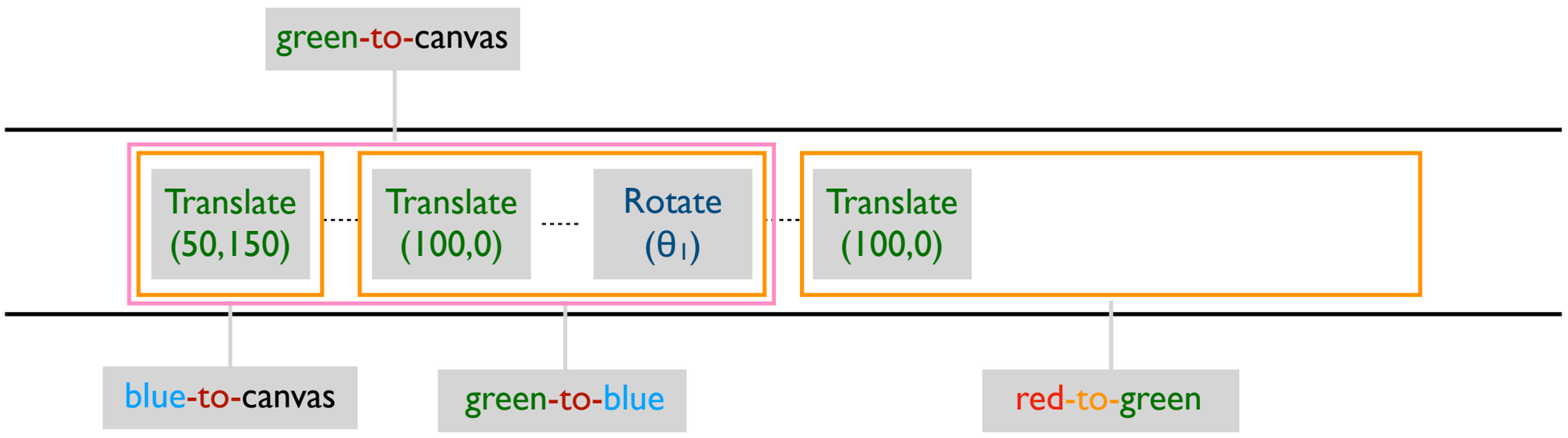
jsbin.com/yifahoq

Week4/Demo0



$$\mathbf{T}_{\text{red2green}} = \begin{pmatrix} 1 & 0 & 100 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

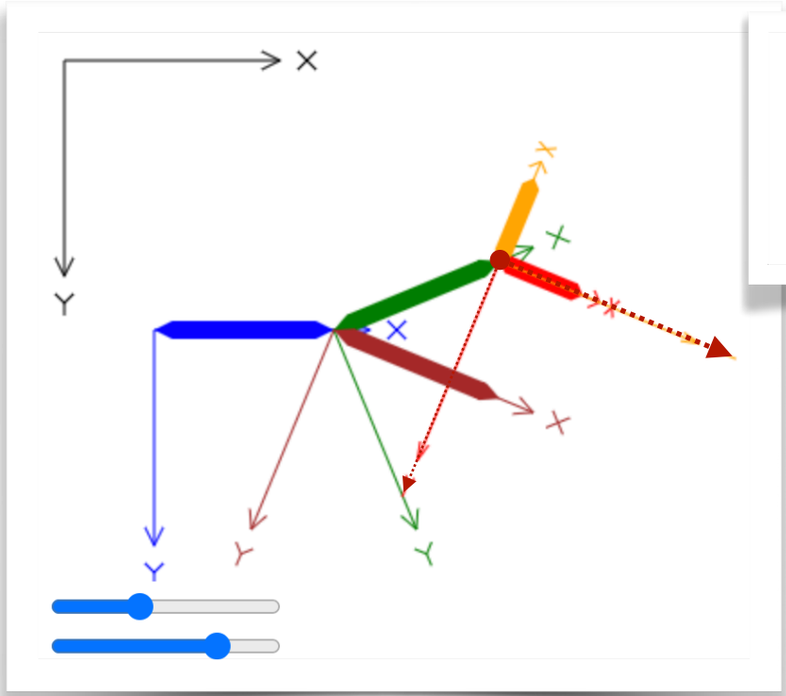
```
mat3.fromTranslation(Tblue_to_canvas,[50,150]);
linkage("blue",Tblue_to_canvas);

var Tgreen_to_blue = mat3.create();
mat3.fromTranslation(Tgreen_to_blue,[100,0]);
mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
var Tgreen_to_canvas = mat3.create();
mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
linkage("green",Tgreen_to_canvas);

var Tred_to_green = mat3.create();
mat3.fromTranslation(Tred_to_green,[100,0]);
mat3.rotate(Tred_to_green,Tred_to_green,phi1);
mat3.scale(Tred_to_green,Tred_to_green,[0.5,1]);
var Tred_to_canvas = mat3.create();
mat3.multiply(Tred_to_canvas,Tgreen_to_canvas,Tred_to_green);
linkage("red",Tred_to_canvas);

[...]
```



Transforms in algebraic form (via *glmatrix*)

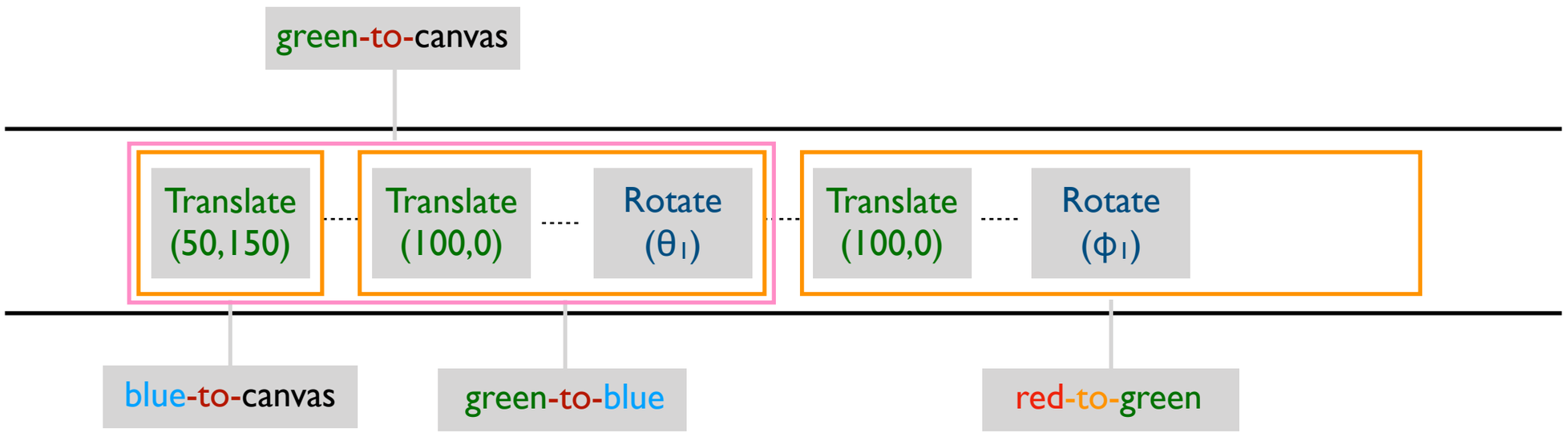
jsbin.com/yifahoq

Week4/Demo0



$$\mathbf{T}_{\text{red2green}} = \begin{pmatrix} 1 & 0 & 100 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\phi_1 & -\sin\phi_1 & 0 \\ \sin\phi_1 & \cos\phi_1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

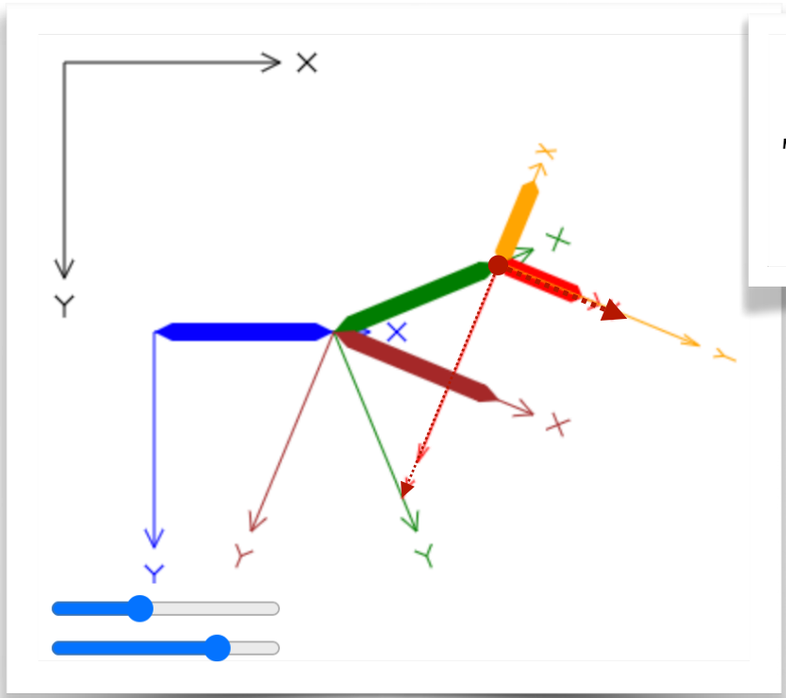
```
mat3.fromTranslation(Tblue_to_canvas,[50,150]);
linkage("blue",Tblue_to_canvas);

var Tgreen_to_blue = mat3.create();
mat3.fromTranslation(Tgreen_to_blue,[100,0]);
mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
var Tgreen_to_canvas = mat3.create();
mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
linkage("green",Tgreen_to_canvas);

var Tred_to_green = mat3.create();
mat3.fromTranslation(Tred_to_green,[100,0]);
mat3.rotate(Tred_to_green,Tred_to_green,phi1);
mat3.scale(Tred_to_green,Tred_to_green,[0.5,1]);
var Tred_to_canvas = mat3.create();
mat3.multiply(Tred_to_canvas,Tgreen_to_canvas,Tred_to_green);
linkage("red",Tred_to_canvas);

[...]
```



Transforms in algebraic form (via *glmatrix*)

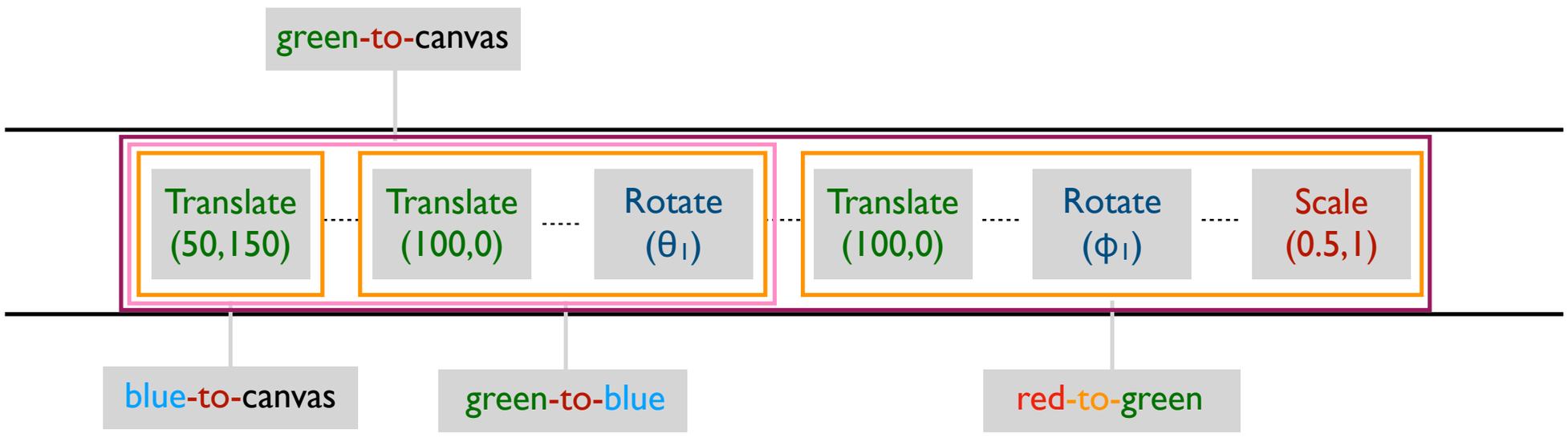
jsbin.com/yifahoq

Week4/Demo0



$$\mathbf{T}_{\text{red2green}} = \begin{pmatrix} 1 & 0 & 100 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\phi_1 & -\sin\phi_1 & 0 \\ \sin\phi_1 & \cos\phi_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0.5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

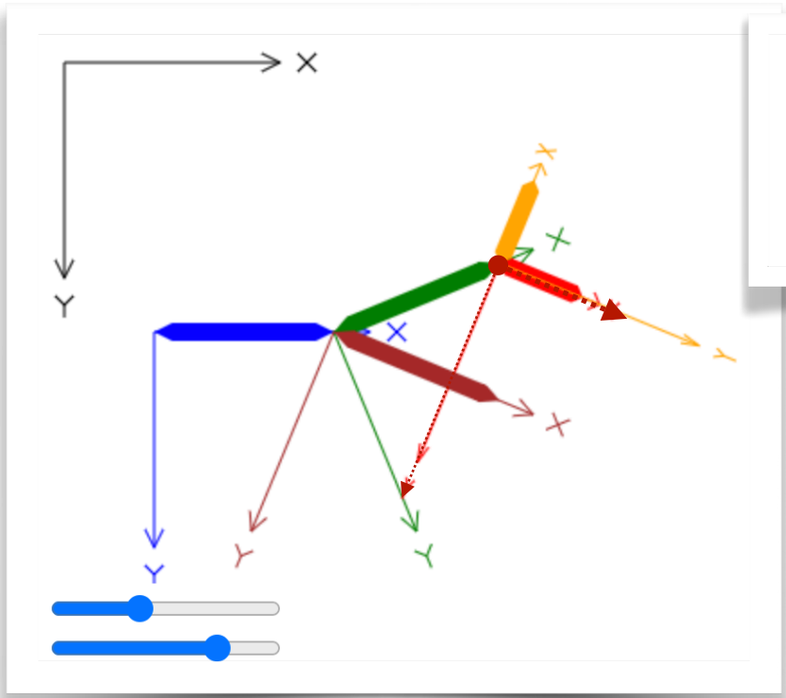
```
mat3.fromTranslation(Tblue_to_canvas,[50,150]);
linkage("blue",Tblue_to_canvas);

var Tgreen_to_blue = mat3.create();
mat3.fromTranslation(Tgreen_to_blue,[100,0]);
mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
var Tgreen_to_canvas = mat3.create();
mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
linkage("green",Tgreen_to_canvas);

var Tred_to_green = mat3.create();
mat3.fromTranslation(Tred_to_green,[100,0]);
mat3.rotate(Tred_to_green,Tred_to_green,phi1);
mat3.scale(Tred_to_green,Tred_to_green,[0.5,1]);
var Tred_to_canvas = mat3.create();
mat3.multiply(Tred_to_canvas,Tgreen_to_canvas,Tred_to_green);
linkage("red",Tred_to_canvas);

[...]
```



Transforms in algebraic form (via *glmatrix*)

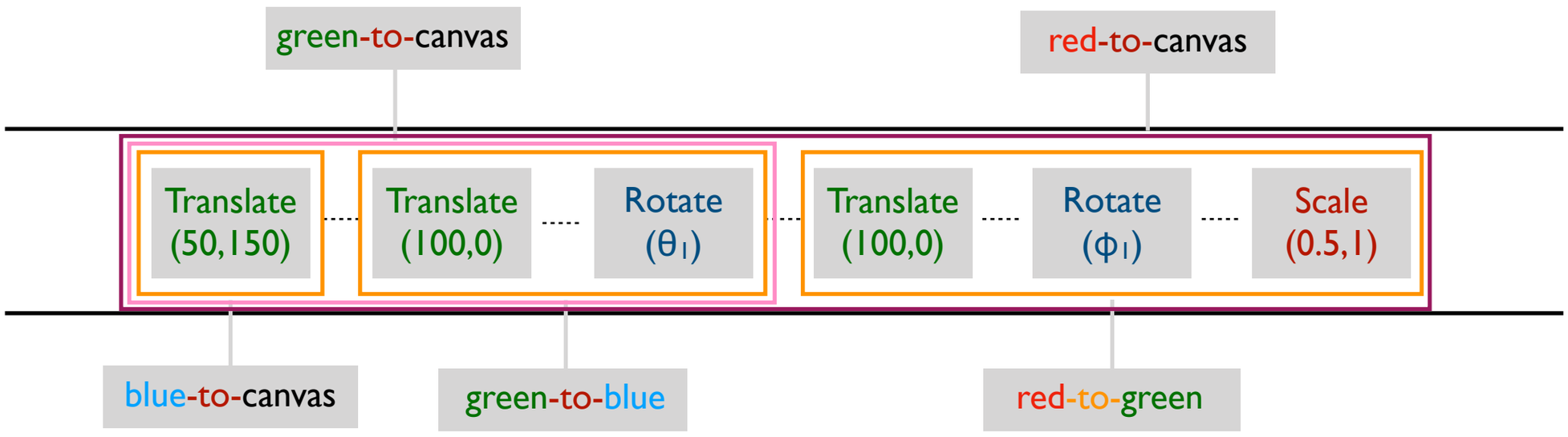
jsbin.com/yifahoq

Week4/Demo0



$$\mathbf{T}_{\text{red2canvas}} = \mathbf{T}_{\text{green2canvas}}\mathbf{T}_{\text{red2green}}$$

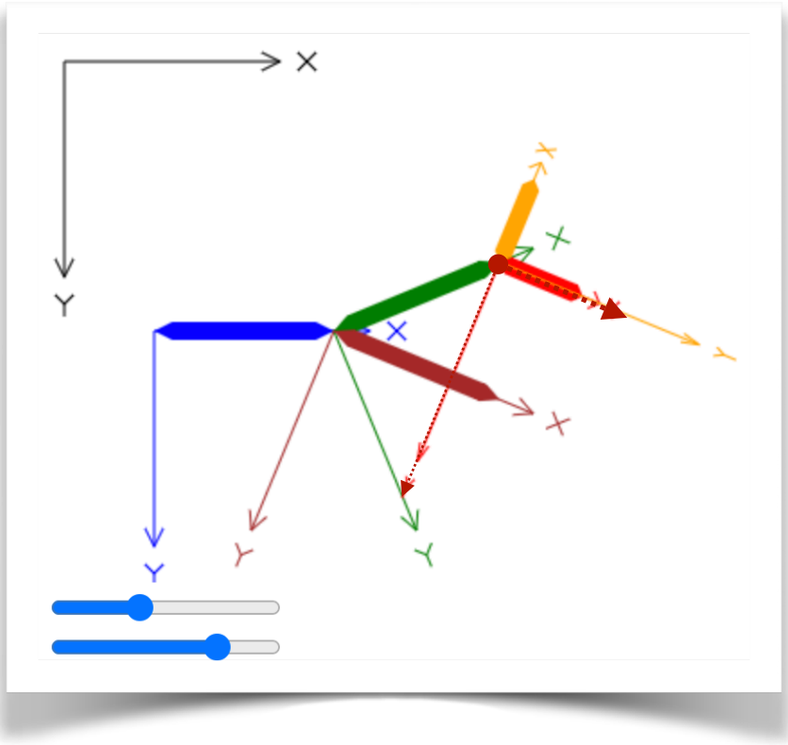
```
mat3.fromTranslation(Tblue_to_canvas,[50,150]);
linkage("blue",Tblue_to_canvas);

var Tgreen_to_blue = mat3.create();
mat3.fromTranslation(Tgreen_to_blue,[100,0]);
mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
var Tgreen_to_canvas = mat3.create();
mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
linkage("green",Tgreen_to_canvas);

var Tred_to_green = mat3.create();
mat3.fromTranslation(Tred_to_green,[100,0]);
mat3.rotate(Tred_to_green,Tred_to_green,phi1);
mat3.scale(Tred_to_green,Tred_to_green,[0.5,1]);
var Tred_to_canvas = mat3.create();
mat3.multiply(Tred_to_canvas,Tgreen_to_canvas,Tred_to_green);
linkage("red",Tred_to_canvas);

[...]
```



Transforms in algebraic form (via *glmatrix*)

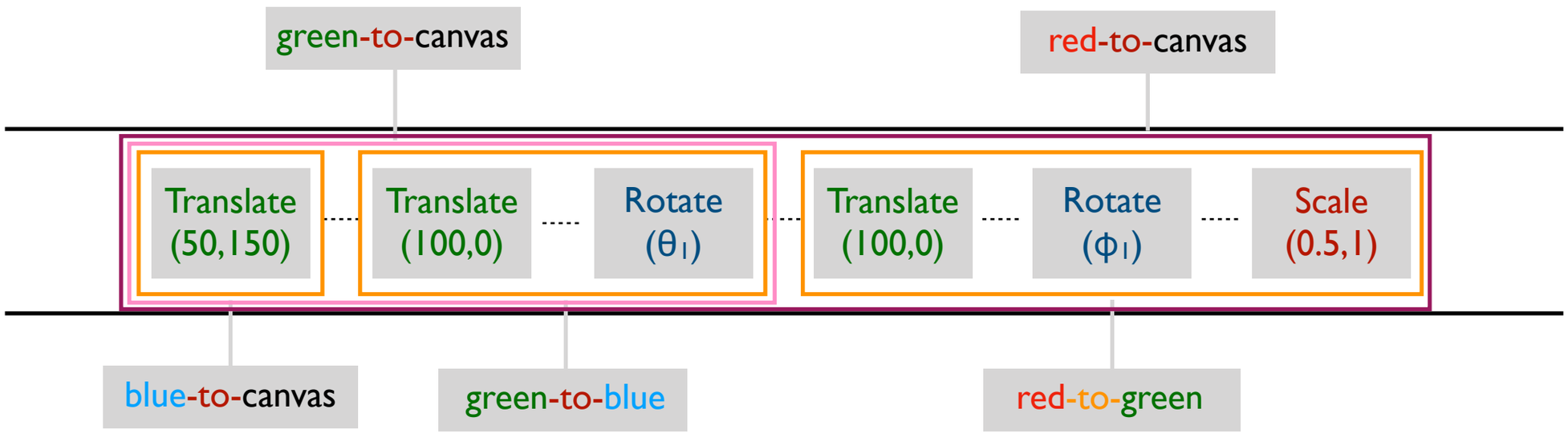
jsbin.com/yifahoq

Week4/Demo0



demo.js

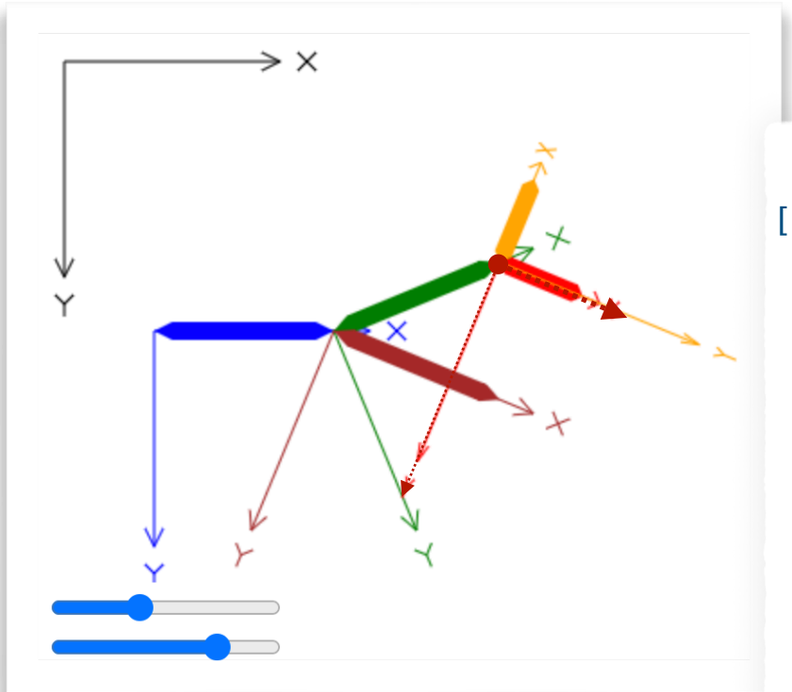
```
[...]

    var Tblue_to_canvas = mat3.create();
    mat3.fromTranslation(Tblue_to_canvas,[50,150]);
    linkage("blue",Tblue_to_canvas);

    var Tgreen_to_blue = mat3.create();
    mat3.fromTranslation(Tgreen_to_blue,[100,0]);
    mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
    var Tgreen_to_canvas = mat3.create();
    mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
    linkage("green",Tgreen_to_canvas);

    var Tred_to_green = mat3.create();
    mat3.fromTranslation(Tred_to_green,[100,0]);
    mat3.rotate(Tred_to_green,Tred_to_green,phi1);
    mat3.scale(Tred_to_green,Tred_to_green,[0.5,1]);
    var Tred_to_canvas = mat3.create();
    mat3.multiply(Tred_to_canvas,Tgreen_to_canvas,Tred_to_green);
    linkage("red",Tred_to_canvas);

[...]
```



Transforms in algebraic form (via *glmatrix*)

jsbin.com/yifahoq

Week4/Demo0



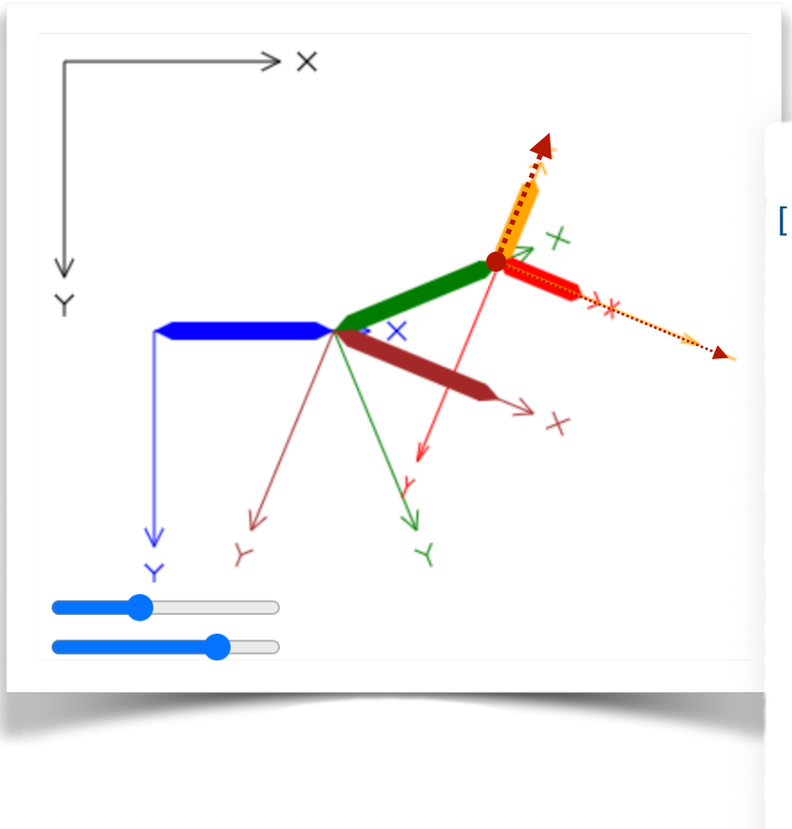
demo.js

```
[...]

    var Tblue_to_canvas = mat3.create();
    mat3.fromTranslation(Tblue_to_canvas,[50,150]);
    linkage("blue",Tblue_to_canvas);

    var Tgreen_to_blue = mat3.create();
    mat3.fromTranslation(Tgreen_to_blue,[100,0]);
    mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
    var Tgreen_to_canvas = mat3.create();
    mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
    linkage("green",Tgreen_to_canvas);

    var Tred_to_green = mat3.create();
    mat3.fromTranslation(Tred_to_green,[100,0]);
    mat3.rotate(Tred_to_green,Tred_to_green,phi1);
    mat3.scale(Tred_to_green,Tred_to_green,[0.5,1]);
    var Tred_to_canvas = mat3.create();
    mat3.multiply(Tred_to_canvas,Tgreen_to_canvas,Tred_to_green);
    linkage("red",Tred_to_canvas);

    var Torange_to_green = mat3.create();
    mat3.fromTranslation(Torange_to_green,[100,0]);
    mat3.rotate(Torange_to_green,Torange_to_green,phi2);
    mat3.scale(Torange_to_green,Torange_to_green,[0.5,1]);
    var Torange_to_canvas = mat3.create();
    mat3.multiply(Torange_to_canvas,Tgreen_to_canvas,Torange_to_green);
    linkage("orange",Torange_to_canvas);

    var Tbrown_to_blue = mat3.create();
    mat3.fromTranslation(Tbrown_to_blue,[100,0]);
    mat3.rotate(Tbrown_to_blue,Tbrown_to_blue,theta2);
    var Tbrown_to_canvas = mat3.create();
    mat3.multiply(Tbrown_to_canvas, Tblue_to_canvas, Tbrown_to_blue);
    linkage("brown",Tbrown_to_canvas);

[...]
```

Transforms in algebraic form (via *glmatrix*)

jsbin.com/yifahoq

Week4/Demo0



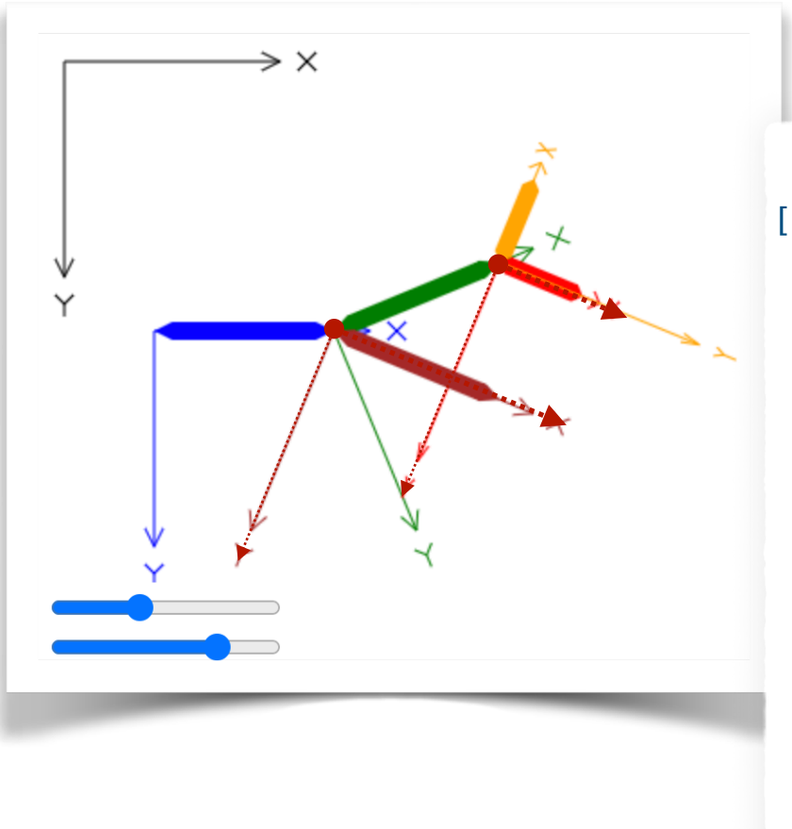
demo.js

```
[...]

    var Tblue_to_canvas = mat3.create();
    mat3.fromTranslation(Tblue_to_canvas,[50,150]);
    linkage("blue",Tblue_to_canvas);

    var Tgreen_to_blue = mat3.create();
    mat3.fromTranslation(Tgreen_to_blue,[100,0]);
    mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
    var Tgreen_to_canvas = mat3.create();
    mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
    linkage("green",Tgreen_to_canvas);

    var Tred_to_green = mat3.create();
    mat3.fromTranslation(Tred_to_green,[100,0]);
    mat3.rotate(Tred_to_green,Tred_to_green,phi1);
    mat3.scale(Tred_to_green,Tred_to_green,[0.5,1]);
    var Tred_to_canvas = mat3.create();
    mat3.multiply(Tred_to_canvas,Tgreen_to_canvas,Tred_to_green);
    linkage("red",Tred_to_canvas);

    var Torange_to_green = mat3.create();
    mat3.fromTranslation(Torange_to_green,[100,0]);
    mat3.rotate(Torange_to_green,Torange_to_green,phi2);
    mat3.scale(Torange_to_green,Torange_to_green,[0.5,1]);
    var Torange_to_canvas = mat3.create();
    mat3.multiply(Torange_to_canvas,Tgreen_to_canvas,Torange_to_green);
    linkage("orange",Torange_to_canvas);

    var Tbrown_to_blue = mat3.create();
    mat3.fromTranslation(Tbrown_to_blue,[100,0]);
    mat3.rotate(Tbrown_to_blue,Tbrown_to_blue,theta2);
    var Tbrown_to_canvas = mat3.create();
    mat3.multiply(Tbrown_to_canvas, Tblue_to_canvas, Tbrown_to_blue);
    linkage("brown",Tbrown_to_canvas);

[...]
```

Transforms in algebraic form (via *glmatrix*)

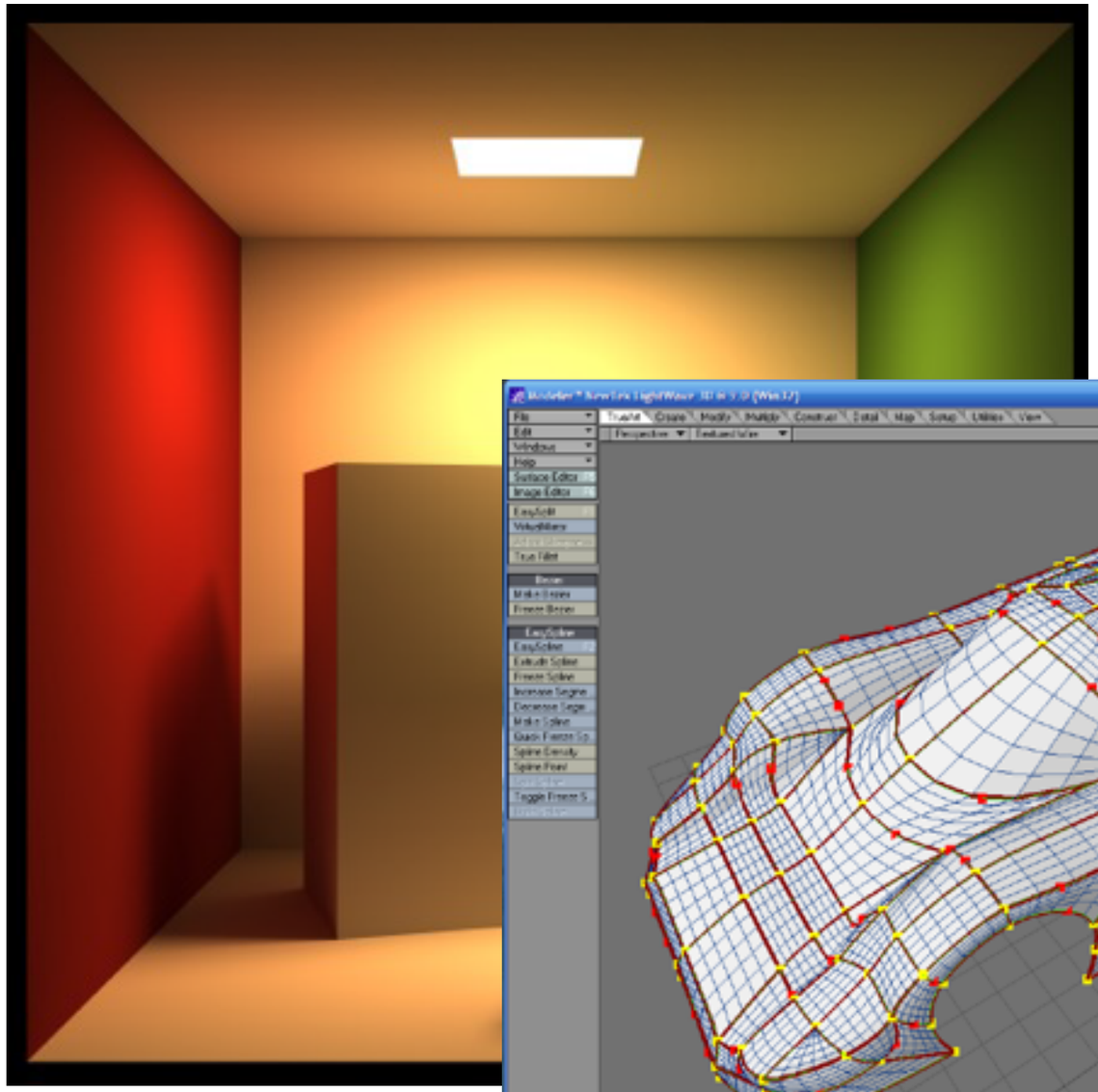
jsbin.com/yifahoq

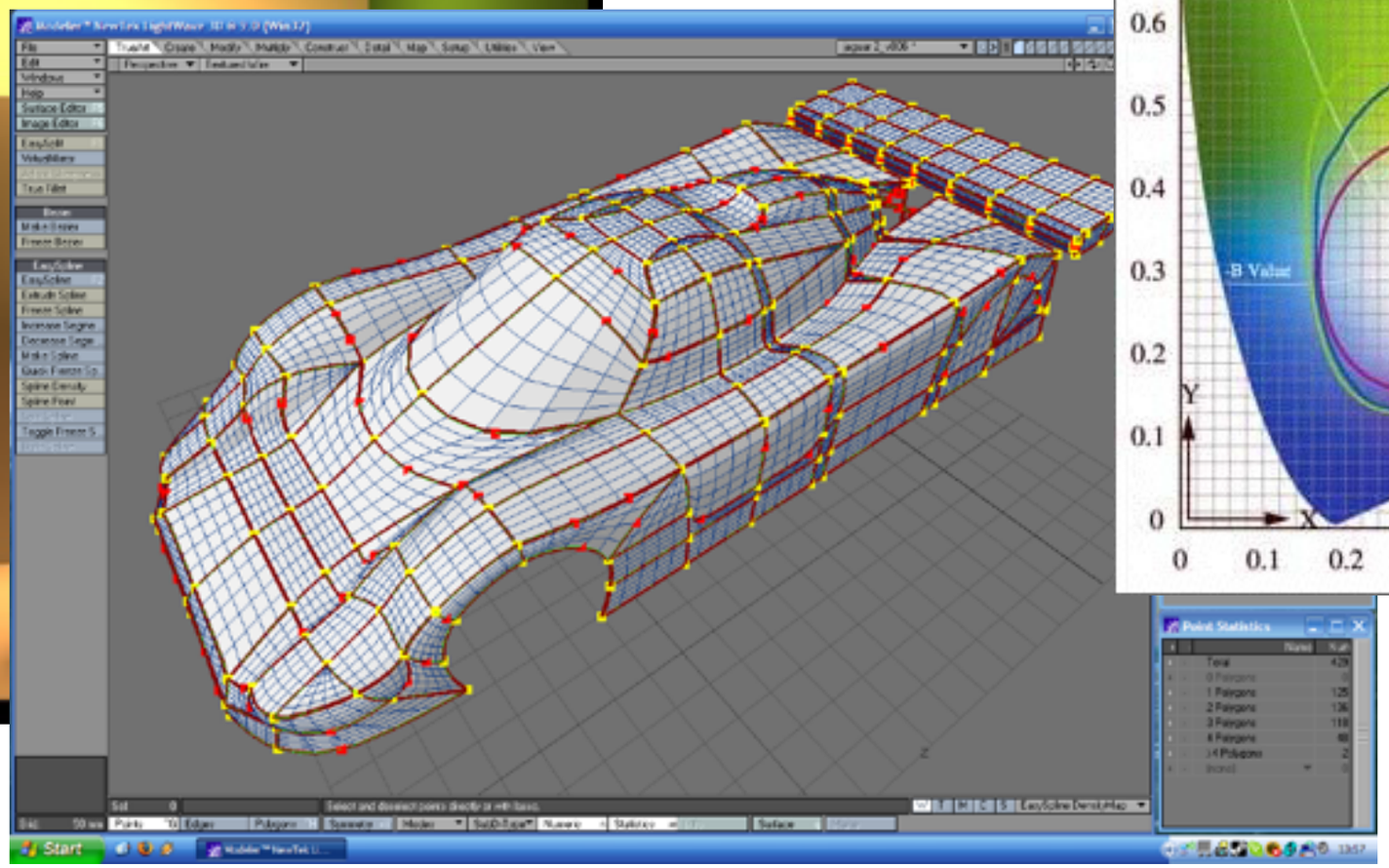
Week4/Demo0

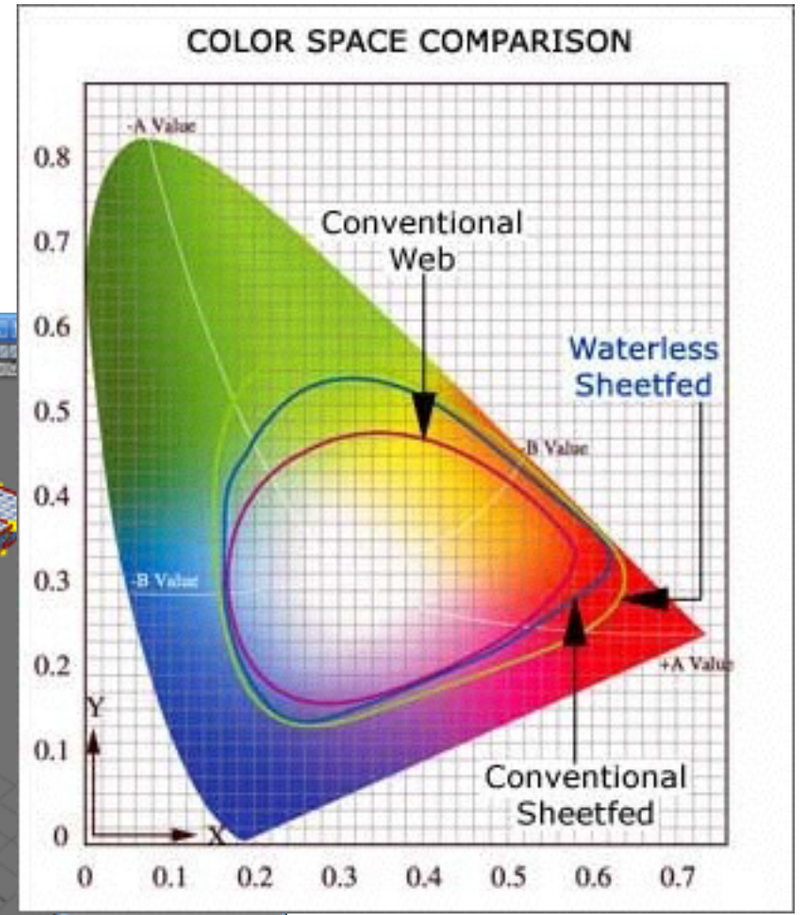


demo.js

```
[...]

    var Tblue_to_canvas = mat3.create();
    mat3.fromTranslation(Tblue_to_canvas,[50,150]);
    linkage("blue",Tblue_to_canvas);

    var Tgreen_to_blue = mat3.create();
    mat3.fromTranslation(Tgreen_to_blue,[100,0]);
    mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
    var Tgreen_to_canvas = mat3.create();
    mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
    linkage("green",Tgreen_to_canvas);

    var Tred_to_green = mat3.create();
    mat3.fromTranslation(Tred_to_green,[100,0]);
    mat3.rotate(Tred_to_green,Tred_to_green,phi1);
    mat3.scale(Tred_to_green,Tred_to_green,[0.5,1]);
    var Tred_to_canvas = mat3.create();
    mat3.multiply(Tred_to_canvas,Tgreen_to_canvas,Tred_to_green);
    linkage("red",Tred_to_canvas);

    var Torange_to_green = mat3.create();
    mat3.fromTranslation(Torange_to_green,[100,0]);
    mat3.rotate(Torange_to_green,Torange_to_green,phi2);
    mat3.scale(Torange_to_green,Torange_to_green,[0.5,1]);
    var Torange_to_canvas = mat3.create();
    mat3.multiply(Torange_to_canvas,Tgreen_to_canvas,Torange_to_green);
    linkage("orange",Torange_to_canvas);

    var Tbrown_to_blue = mat3.create();
    mat3.fromTranslation(Tbrown_to_blue,[100,0]);
    mat3.rotate(Tbrown_to_blue,Tbrown_to_blue,theta2);
    var Tbrown_to_canvas = mat3.create();
    mat3.multiply(Tbrown_to_canvas, Tblue_to_canvas, Tbrown_to_blue);
    linkage("brown",Tbrown_to_canvas);

[...]
```







Lecture 6 : Transforms and Hierarchical Modeling in 2D (with an intro to the theory and algebra of transforms)

Tuesday September 26th 2023

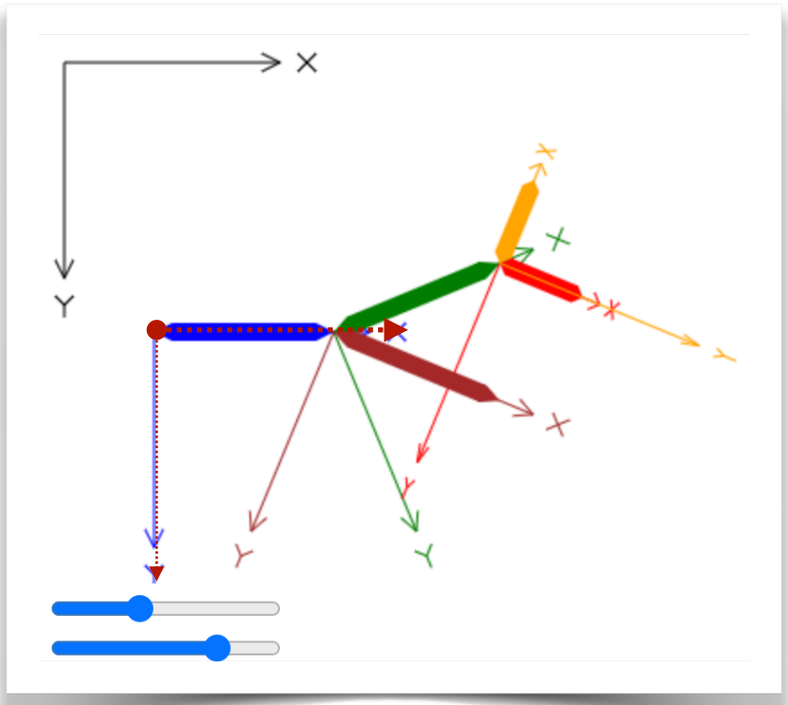
Canvas vs. Glmatrix

- Note differences of how we "go back" to "parent joints" in the hierarchical tree
- No explicitly maintained stack in Glmatrix!
  (We used names to "save" prior transforms)
- If we use explicit algebra, we need to either transform prior to drawing, or "set" the current transform

Transforms in algebraic form (via *glmatrix*)

jsbin.com/yifahoq

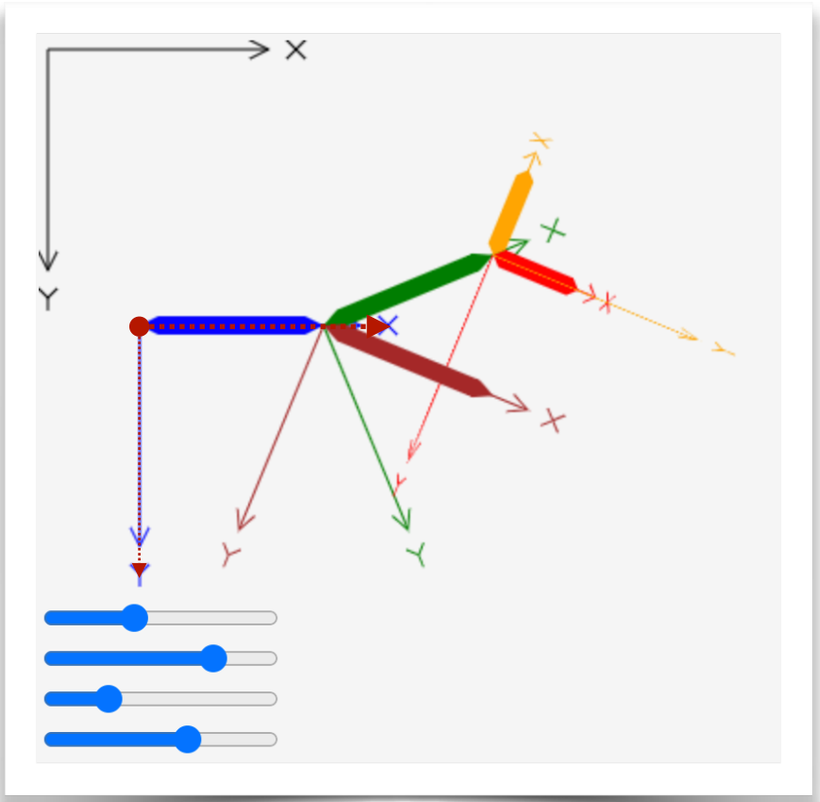
Week4/Demo0



demo.js

```
[...]

  var Tblue_to_canvas = mat3.create();
  mat3.fromTranslation(Tblue_to_canvas,[50,150]);
  linkage("blue",Tblue_to_canvas);

  var Tgreen_to_blue = mat3.create();
  mat3.fromTranslation(Tgreen_to_blue,[100,0]);
  mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
  var Tgreen_to_canvas = mat3.create();
  mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
  linkage("green",Tgreen_to_canvas);

  var Tred_to_green = mat3.create();
  mat3.fromTranslation(Tred_to_green,[100,0]);
  mat3.rotate(Tred_to_green,Tred_to_green,phi1);
  mat3.scale(Tred_to_green,Tred_to_green,[0.5,1]);
  var Tred_to_canvas = mat3.create();
  mat3.multiply(Tred_to_canvas,Tgreen_to_canvas,Tred_to_green);
  linkage("red",Tred_to_canvas);

[...]
```

Canvas transform stack



Canvas transform stack

canvas-to-blue

jsbin.com/wuyutirife

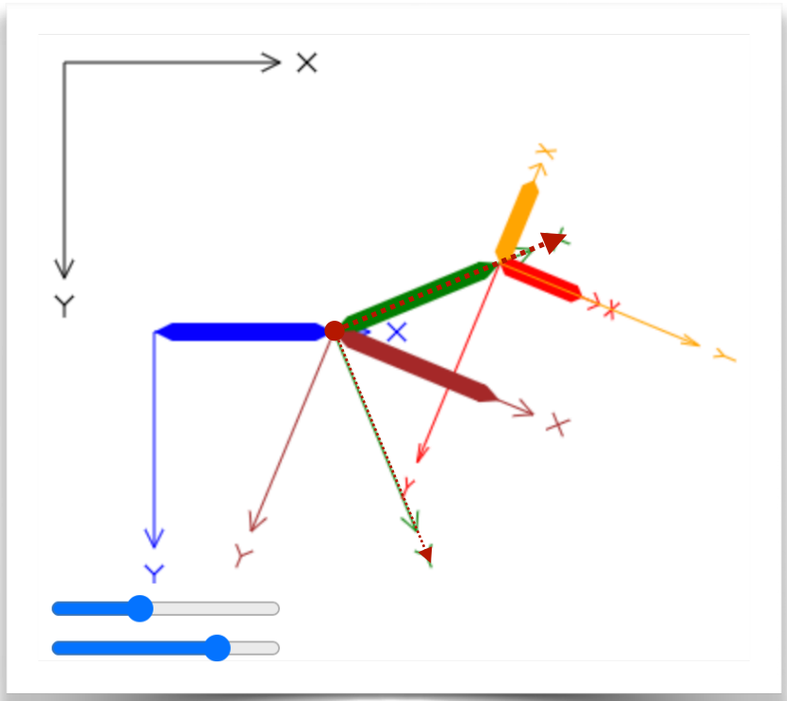
JavaScript

```
[...]
  // 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)
[...]
```

Transforms in algebraic form (via *glmatrix*)

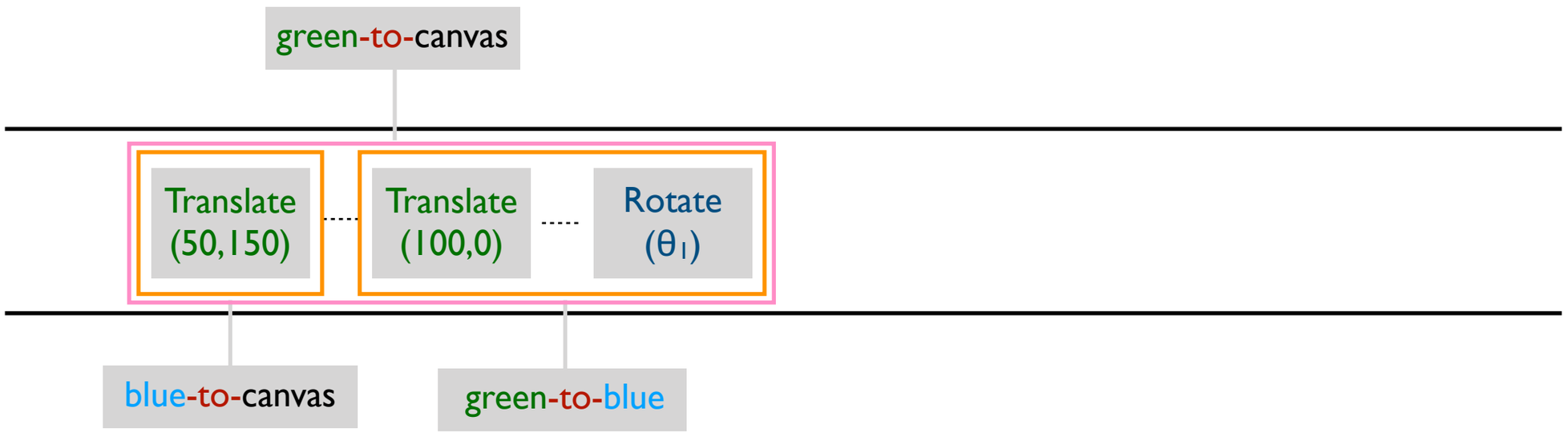
jsbin.com/yifahoq

Week4/Demo0

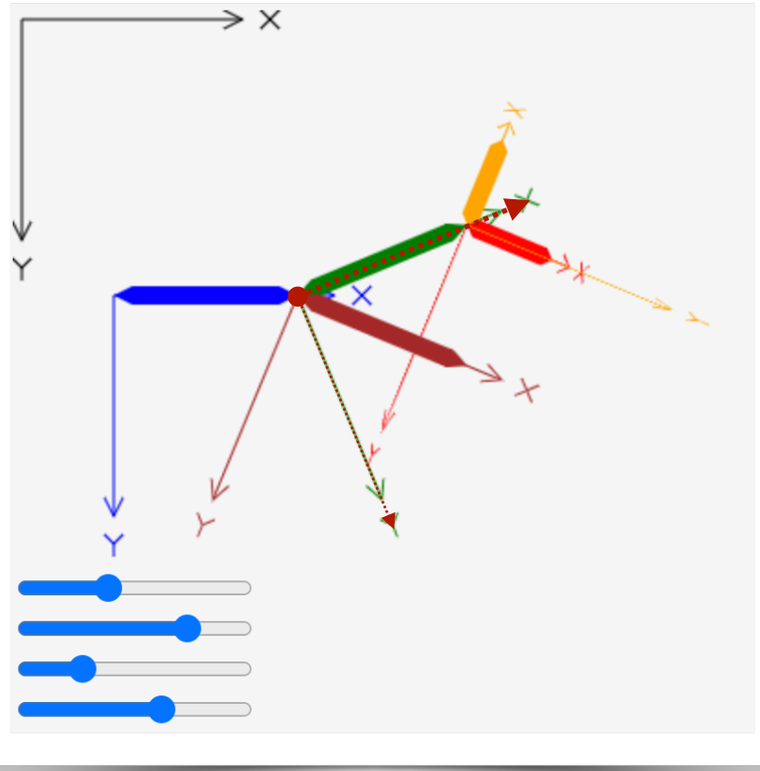


demo.js

```
[...]

    var Tblue_to_canvas = mat3.create();
    mat3.fromTranslation(Tblue_to_canvas,[50,150]);
    linkage("blue",Tblue_to_canvas);

    var Tgreen_to_blue = mat3.create();
    mat3.fromTranslation(Tgreen_to_blue,[100,0]);
    mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
    var Tgreen_to_canvas = mat3.create();
    mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
    linkage("green",Tgreen_to_canvas);

    var Tred_to_green = mat3.create();
    mat3.fromTranslation(Tred_to_green,[100,0]);
    mat3.rotate(Tred_to_green,Tred_to_green,phi1);
    mat3.scale(Tred_to_green,Tred_to_green,[0.5,1]);
    var Tred_to_canvas = mat3.create();
    mat3.multiply(Tred_to_canvas,Tgreen_to_canvas,Tred_to_green);
    linkage("red",Tred_to_canvas);

[...]
```



Canvas transform stack



Canvas transform stack

canvas-to-blue

blue-to-green

canvas-to-blue

jsbin.com/wuyutirife

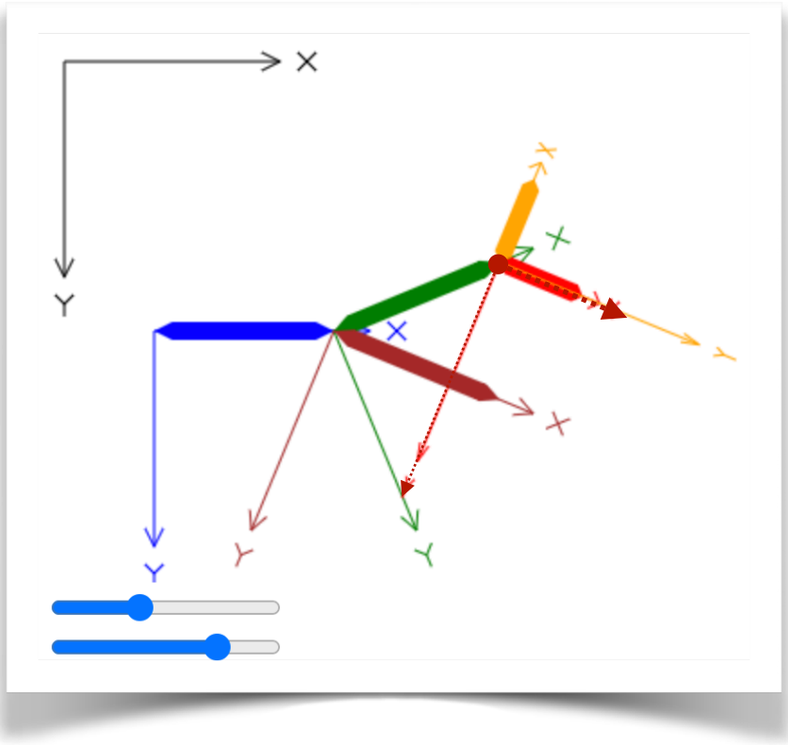
JavaScript

```
[...]
  // 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)
[...]
```

Transforms in algebraic form (via *glmatrix*)

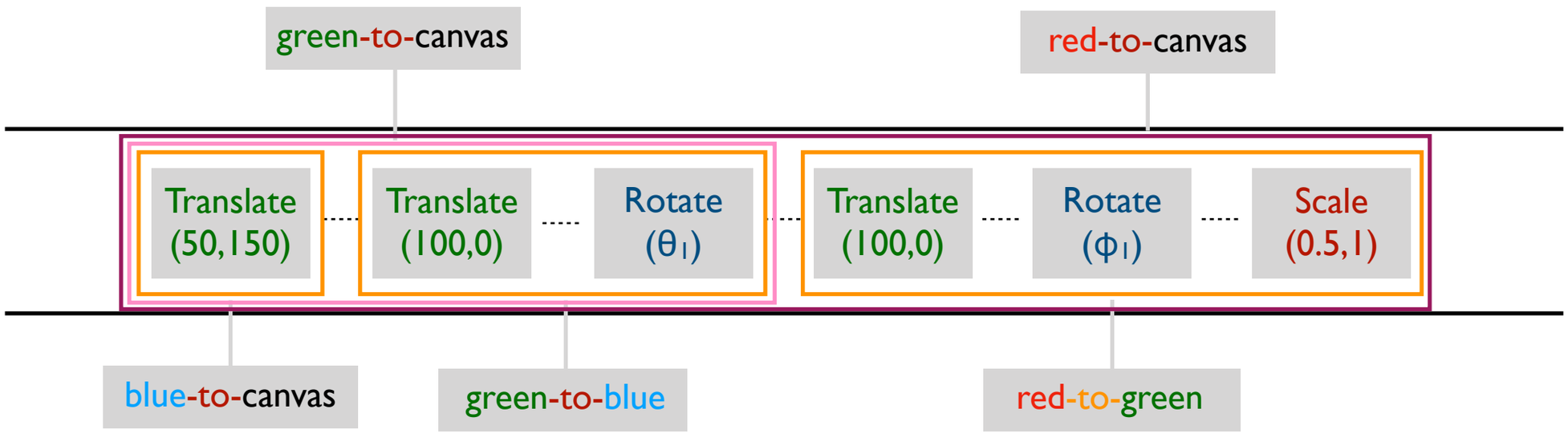
jsbin.com/yifahoq

Week4/Demo0

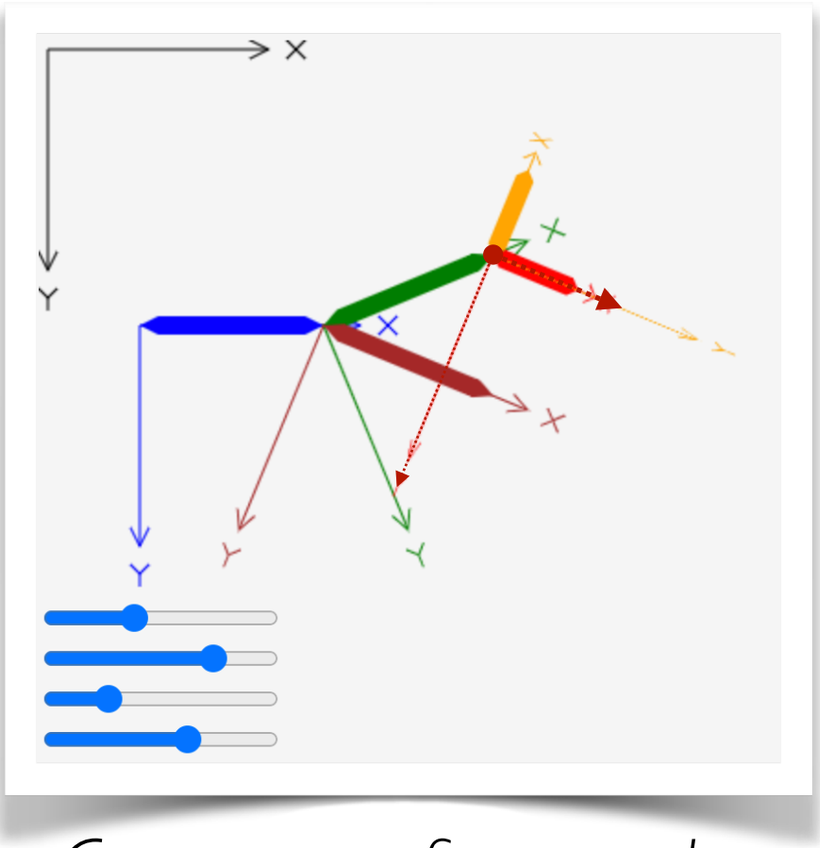


demo.js

```
[...]

    var Tblue_to_canvas = mat3.create();
    mat3.fromTranslation(Tblue_to_canvas,[50,150]);
    linkage("blue",Tblue_to_canvas);

    var Tgreen_to_blue = mat3.create();
    mat3.fromTranslation(Tgreen_to_blue,[100,0]);
    mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
    var Tgreen_to_canvas = mat3.create();
    mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
    linkage("green",Tgreen_to_canvas);

    var Tred_to_green = mat3.create();
    mat3.fromTranslation(Tred_to_green,[100,0]);
    mat3.rotate(Tred_to_green,Tred_to_green,phi1);
    mat3.scale(Tred_to_green,Tred_to_green,[0.5,1]);
    var Tred_to_canvas = mat3.create();
    mat3.multiply(Tred_to_canvas,Tgreen_to_canvas,Tred_to_green);
    linkage("red",Tred_to_canvas);

[...]
```



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

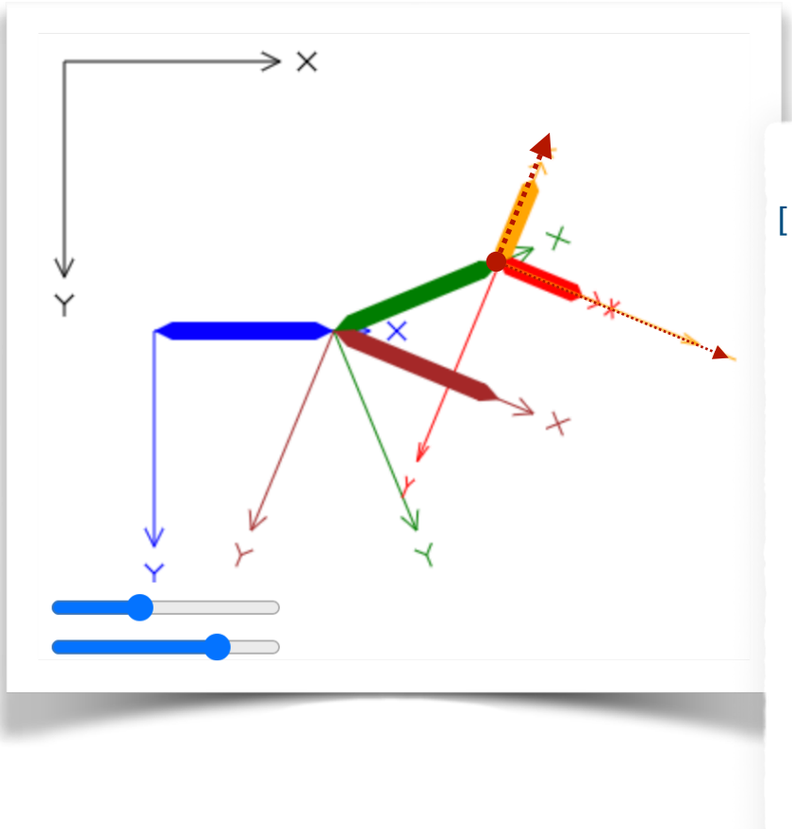
JavaScript

```
[...]
  // 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)
[...]
```

Transforms in algebraic form (via *glmatrix*)

jsbin.com/yifahoq

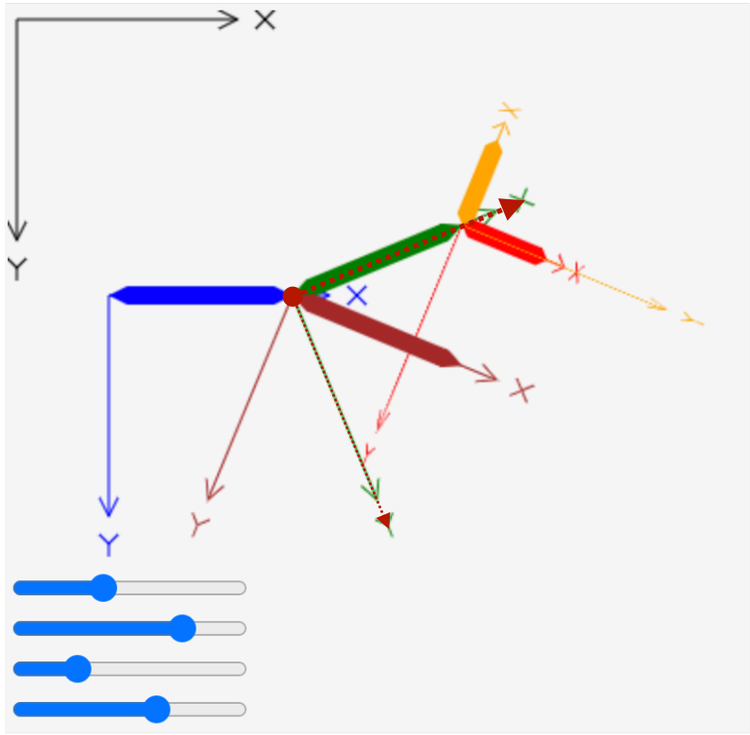
Week4/Demo0



demo.js

```
[...]

    var Tblue_to_canvas = mat3.create();
    mat3.fromTranslation(Tblue_to_canvas,[50,150]);
    linkage("blue",Tblue_to_canvas);

    var Tgreen_to_blue = mat3.create();
    mat3.fromTranslation(Tgreen_to_blue,[100,0]);
    mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
    var Tgreen_to_canvas = mat3.create();
    mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
    linkage("green",Tgreen_to_canvas);

    var Tred_to_green = mat3.create();
    mat3.fromTranslation(Tred_to_green,[100,0]);
    mat3.rotate(Tred_to_green,Tred_to_green,phi1);
    mat3.scale(Tred_to_green,Tred_to_green,[0.5,1]);
    var Tred_to_canvas = mat3.create();
    mat3.multiply(Tred_to_canvas,Tgreen_to_canvas,Tred_to_green);
    linkage("red",Tred_to_canvas);

    var Torange_to_green = mat3.create();
    mat3.fromTranslation(Torange_to_green,[100,0]);
    mat3.rotate(Torange_to_green,Torange_to_green,phi2);
    mat3.scale(Torange_to_green,Torange_to_green,[0.5,1]);
    var Torange_to_canvas = mat3.create();
    mat3.multiply(Torange_to_canvas,Tgreen_to_canvas,Torange_to_green);
    linkage("orange",Torange_to_canvas);

    var Tbrown_to_blue = mat3.create();
    mat3.fromTranslation(Tbrown_to_blue,[100,0]);
    mat3.rotate(Tbrown_to_blue,Tbrown_to_blue,theta2);
    var Tbrown_to_canvas = mat3.create();
    mat3.multiply(Tbrown_to_canvas, Tblue_to_canvas, Tbrown_to_blue);
    linkage("brown",Tbrown_to_canvas);

[...]
```

Canvas transform stack



Canvas transform stack

canvas-to-blue

blue-to-green

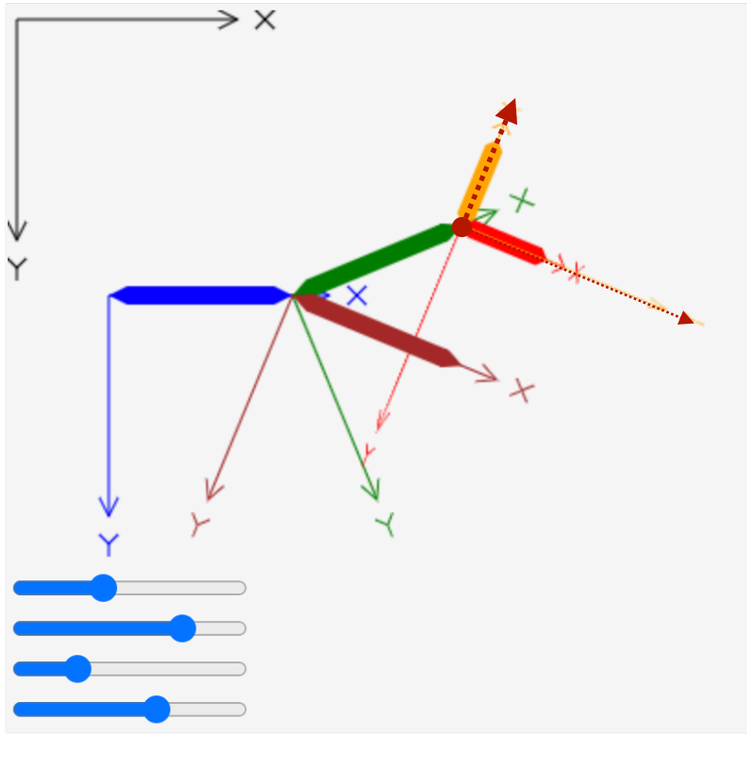
canvas-to-blue

jsbin.com/wuyutirife

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

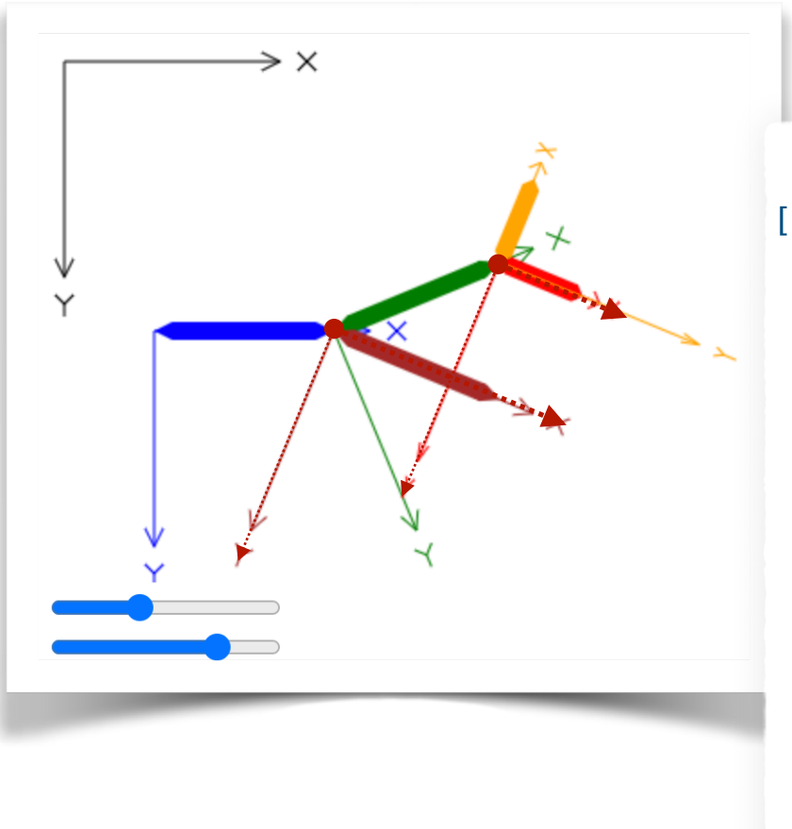
JavaScript

```
[...]
  // 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)
[...]
```

Transforms in algebraic form (via *glmatrix*)

jsbin.com/yifahoq

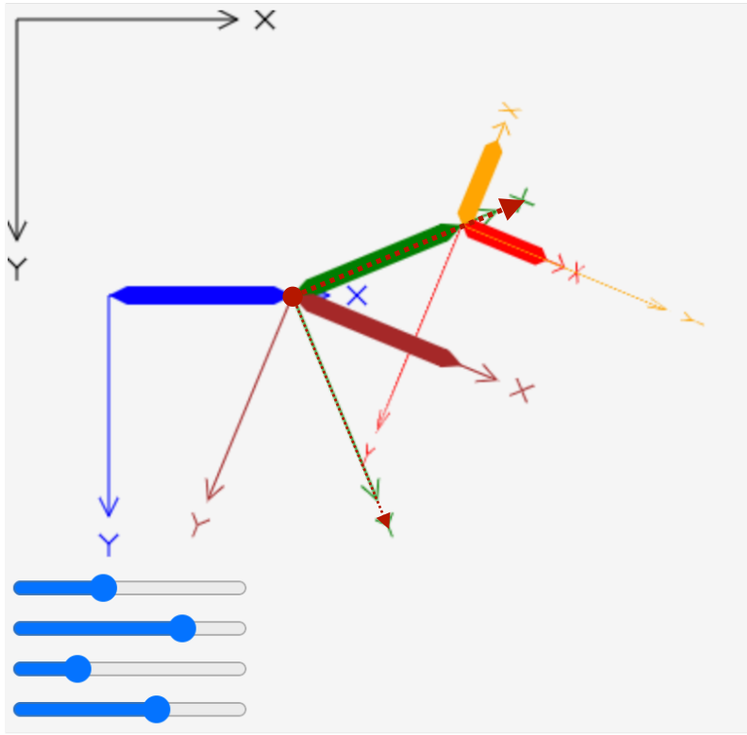
Week4/Demo0



demo.js

```
[...]

    var Tblue_to_canvas = mat3.create();
    mat3.fromTranslation(Tblue_to_canvas,[50,150]);
    linkage("blue",Tblue_to_canvas);

    var Tgreen_to_blue = mat3.create();
    mat3.fromTranslation(Tgreen_to_blue,[100,0]);
    mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
    var Tgreen_to_canvas = mat3.create();
    mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
    linkage("green",Tgreen_to_canvas);

    var Tred_to_green = mat3.create();
    mat3.fromTranslation(Tred_to_green,[100,0]);
    mat3.rotate(Tred_to_green,Tred_to_green,phi1);
    mat3.scale(Tred_to_green,Tred_to_green,[0.5,1]);
    var Tred_to_canvas = mat3.create();
    mat3.multiply(Tred_to_canvas,Tgreen_to_canvas,Tred_to_green);
    linkage("red",Tred_to_canvas);

    var Torange_to_green = mat3.create();
    mat3.fromTranslation(Torange_to_green,[100,0]);
    mat3.rotate(Torange_to_green,Torange_to_green,phi2);
    mat3.scale(Torange_to_green,Torange_to_green,[0.5,1]);
    var Torange_to_canvas = mat3.create();
    mat3.multiply(Torange_to_canvas,Tgreen_to_canvas,Torange_to_green);
    linkage("orange",Torange_to_canvas);

    var Tbrown_to_blue = mat3.create();
    mat3.fromTranslation(Tbrown_to_blue,[100,0]);
    mat3.rotate(Tbrown_to_blue,Tbrown_to_blue,theta2);
    var Tbrown_to_canvas = mat3.create();
    mat3.multiply(Tbrown_to_canvas, Tblue_to_canvas, Tbrown_to_blue);
    linkage("brown",Tbrown_to_canvas);

[...]
```

Canvas transform stack



Canvas transform stack

canvas-to-blue

blue-to-green

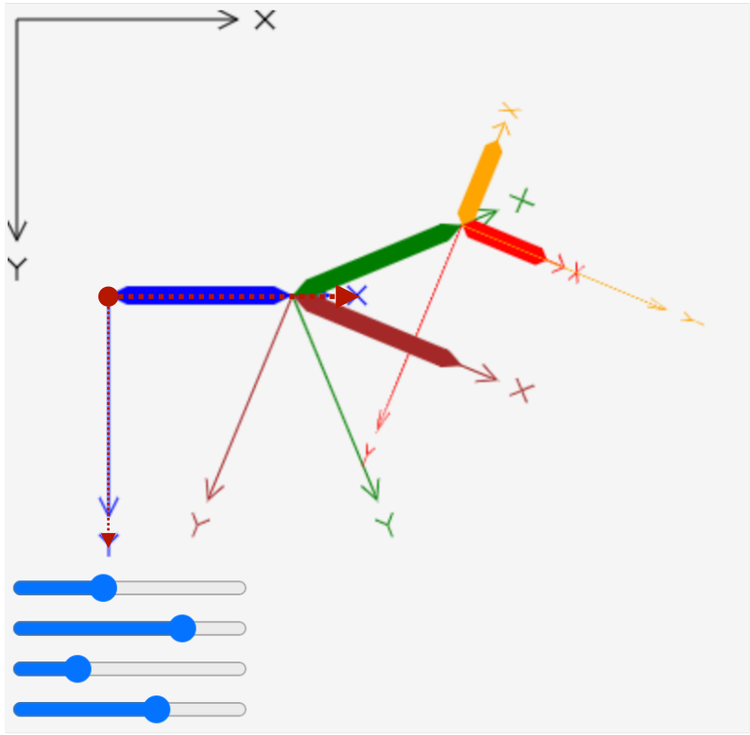
canvas-to-blue

jsbin.com/wuyutirife

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

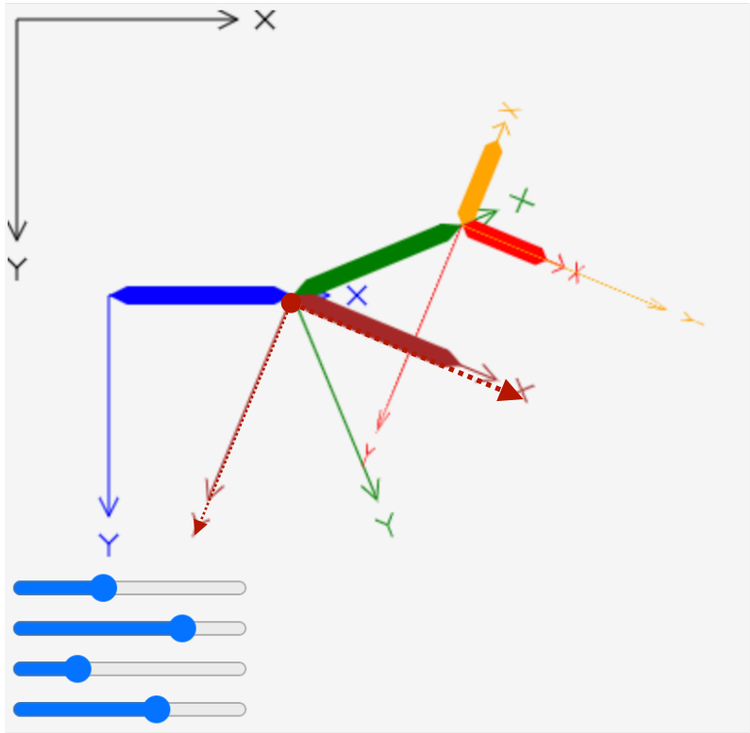
canvas-to-blue

jsbin.com/wuyutirife

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

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)
[...]
```

Setting the Canvas Transform?

jsbin.com/leropag

Week4/Demo0a

- If we use explicit algebra, we need to either transform prior to drawing, or "set" the current transform
- Glmatrix stores the homogeneous transform matrix as an array, that reflects a column-major traversal of the matrix entries

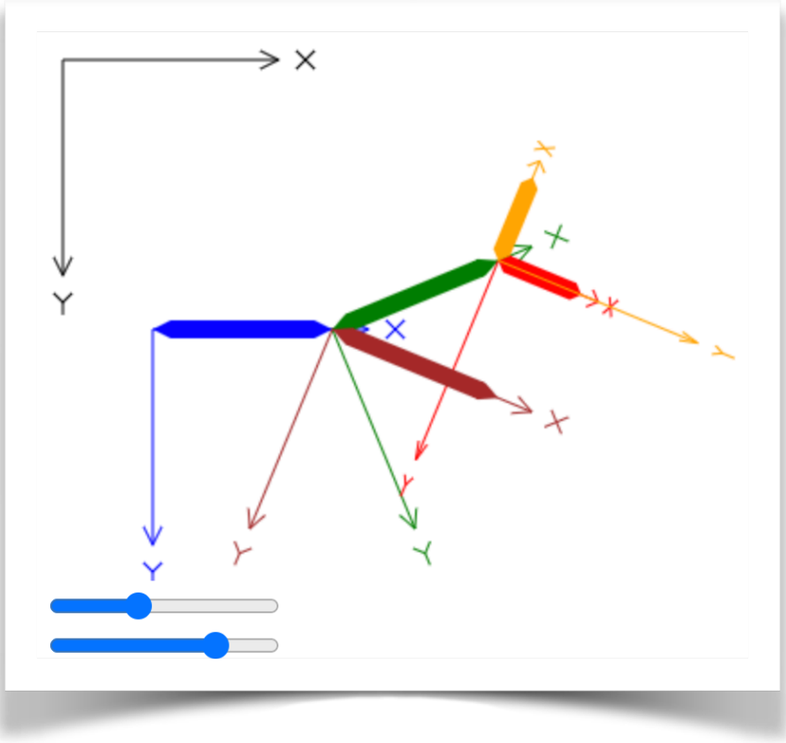
$$\begin{pmatrix} a[0] & a[3] & a[6] \\ a[1] & a[4] & a[7] \\ a[2] & a[5] & a[8] \end{pmatrix}$$

- The canvas setTransform() function requires as parameters the first two rows of this transform (which contain all nontrivial information)

Setting the Canvas Transform?

jsbin.com/leropag

Week4/Demo0a



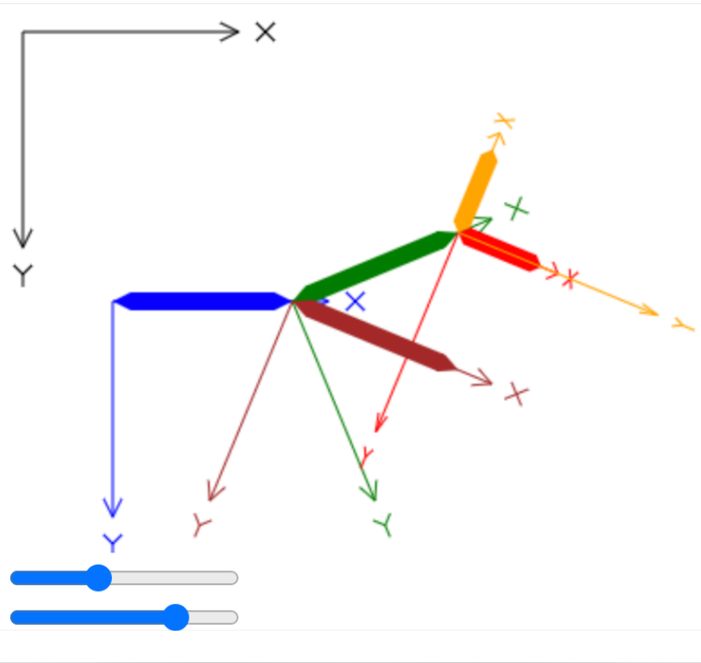
demo.js

```
[...]

    function setCanvasTransform(Tx) {
        context.setTransform(Tx[0],Tx[1],Tx[3],Tx[4],Tx[6],Tx[7]);
    }

    function linkage(color) {
        context.beginPath();
        context.fillStyle = color;
        context.moveTo(0,0);
        context.lineTo(10,5);
        [...]
    }

    [...]

    // make sure you understand these

    var Tblue_to_canvas = mat3.create();
    mat3.fromTranslation(Tblue_to_canvas,[50,150]);
    setCanvasTransform(Tblue_to_canvas);
    linkage("blue");

    var Tgreen_to_blue = mat3.create();
    mat3.fromTranslation(Tgreen_to_blue,[100,0]);
    mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
    var Tgreen_to_canvas = mat3.create();
    mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
    setCanvasTransform(Tgreen_to_canvas);
    linkage("green");

[...]
```

Setting the Canvas Transform?

jsbin.com/leropag

Week4/Demo0a



demo.js

```
[...]

   function setCanvasTransform(Tx) {
       context.setTransform(Tx[0],Tx[1],Tx[3],Tx[4],Tx[6],Tx[7]);
   }

   function linkage(color) {
       context.beginPath();
       context.fillStyle = color;
       context.moveTo(0,0);
       context.lineTo(10,5);
       [...]
   }

   [...]

   // make sure you understand these

   var Tblue_to_canvas = mat3.create();
   mat3.fromTranslation(Tblue_to_canvas,[50,150]);
   setCanvasTransform(Tblue_to_canvas);
   linkage("blue");

   var Tgreen_to_blue = mat3.create();
   mat3.fromTranslation(Tgreen_to_blue,[100,0]);
   mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
   var Tgreen_to_canvas = mat3.create();
   mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
   setCanvasTransform(Tgreen_to_canvas);
   linkage("green");

[...]
```

User-maintained stack

jsbin.com/zimaqos

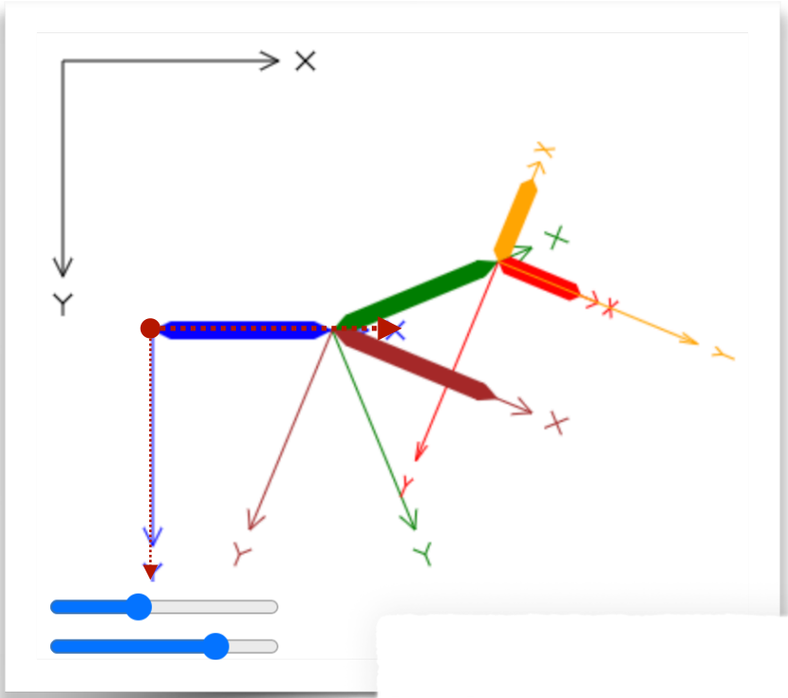
Week4/Demo1

- Our previous use of glmatrix did not explicitly use a transform stack, the same way that canvas does.
- As a substitute, we would name every individual transform we encounter or need with a distinct name
- This might not always work (or be convenient) if we do not know up-front how many components our hierarchical model has
- It is easy to simply maintain our own stack; all arrays in JS can be used as a stack (or as a queue) by using them with the right calls (shift/unshift, or pop/push)

User-maintained stack

jsbin.com/zimaqos

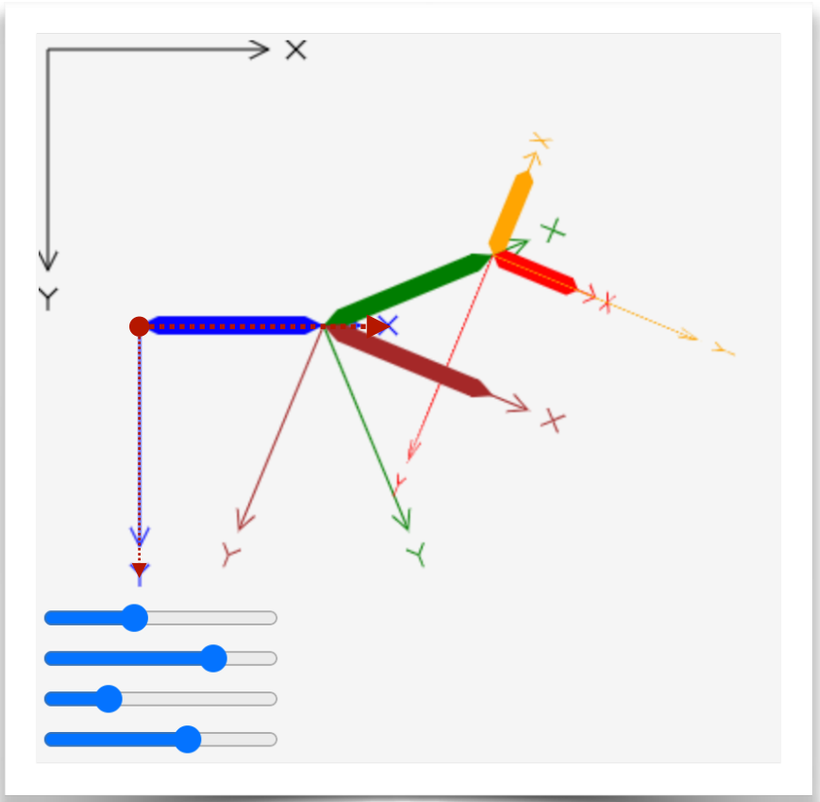
Week4/Demo1



demo.js

```
[...]

    var theta2 = slider4.value*0.005*Math.PI;

    var stack = [ mat3.create() ]; // Initialize stack with identity on top

    function moveToTx(x,y)
    {var res=vec2.create(); vec2.transformMat3(res,[x,y],stack[0]); context.moveTo(res[0],res[1]);}

    function lineToTx(x,y)
    {var res=vec2.create(); vec2.transformMat3(res,[x,y],stack[0]); context.lineTo(res[0],res[1]);}

    function linkage(color) {
        context.beginPath();
        context.fillStyle = color;
        moveToTx(0,0);
        lineToTx(10,5);
        lineToTx(90,5);

[...]
```

Canvas transform stack



Canvas transform stack

canvas-to-blue

jsbin.com/wuyutirife

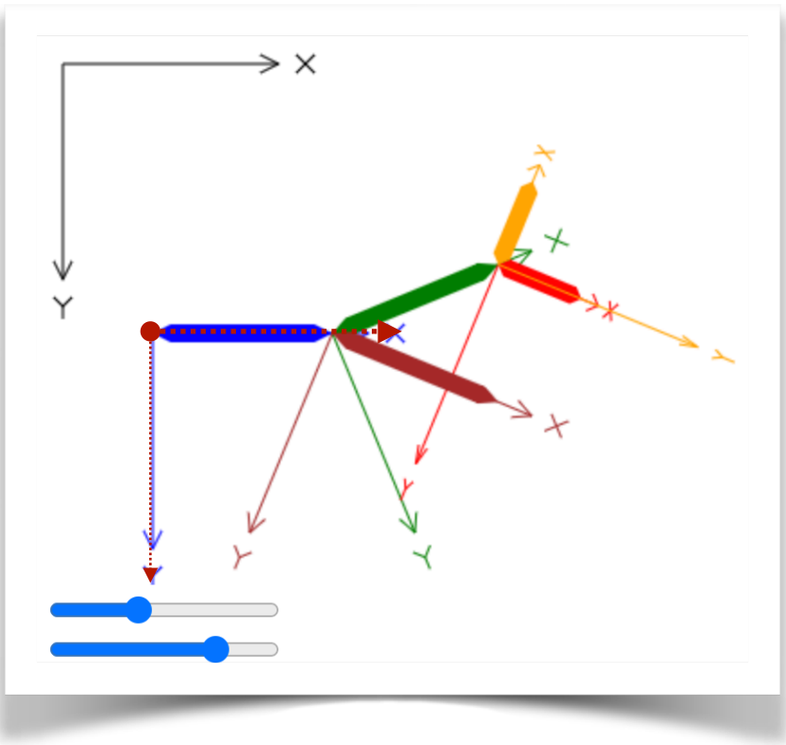
JavaScript

```
[...]
  // 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)
[...]
```

User-maintained stack

jsbin.com/zimaqos

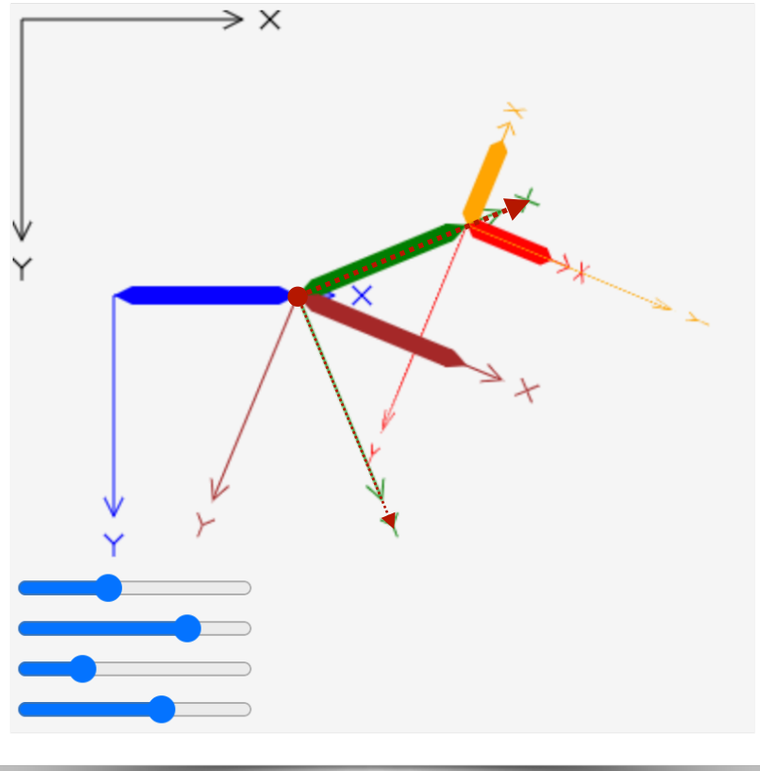
Week4/Demo1



demo.js

```
[...]
  var Tblue_to_canvas = mat3.create();
  mat3.fromTranslation(Tblue_to_canvas,[50,150]);
  mat3.multiply(stack[0],stack[0],Tblue_to_canvas);
   linkage("blue");

  stack.unshift(mat3.clone(stack[0])); // "save" (note: you *need* to clone)
   var Tgreen_to_blue = mat3.create();
  mat3.fromTranslation(Tgreen_to_blue,[100,0]);
  mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
  mat3.multiply(stack[0],stack[0],Tgreen_to_blue);
   linkage("green");

  stack.unshift(mat3.clone(stack[0])); // "save"
   var Tred_to_green = mat3.create();
  mat3.fromTranslation(Tred_to_green,[100,0]);
  mat3.rotate(Tred_to_green,Tred_to_green,phi1);
  mat3.scale(Tred_to_green,Tred_to_green,[0.5,1]);
  mat3.multiply(stack[0],stack[0],Tred_to_green);
   linkage("red");
   stack.shift();          // "restore"

  stack.unshift(mat3.clone(stack[0])); // "save"
  var Torange_to_green = mat3.create();
  mat3.fromTranslation(Torange_to_green,[100,0]);
  mat3.rotate(Torange_to_green,Torange_to_green,phi2);
  mat3.scale(Torange_to_green,Torange_to_green,[0.5,1]);
  mat3.multiply(stack[0],stack[0],Torange_to_green);
   linkage("orange");
   stack.shift();          // "restore"
   stack.shift();          // "restore"

  stack.unshift(mat3.clone(stack[0])); // "save"
   var Tbrown_to_blue = mat3.create();
  mat3.fromTranslation(Tbrown_to_blue,[100,0]);
  mat3.rotate(Tbrown_to_blue,Tbrown_to_blue,theta2);
  mat3.multiply(stack[0],stack[0],Tbrown_to_blue);
   linkage("brown");
   stack.shift();          // “restore"

[...]
```

Canvas transform stack



Canvas transform stack

canvas-to-blue

blue-to-green

canvas-to-blue

jsbin.com/wuyutirife

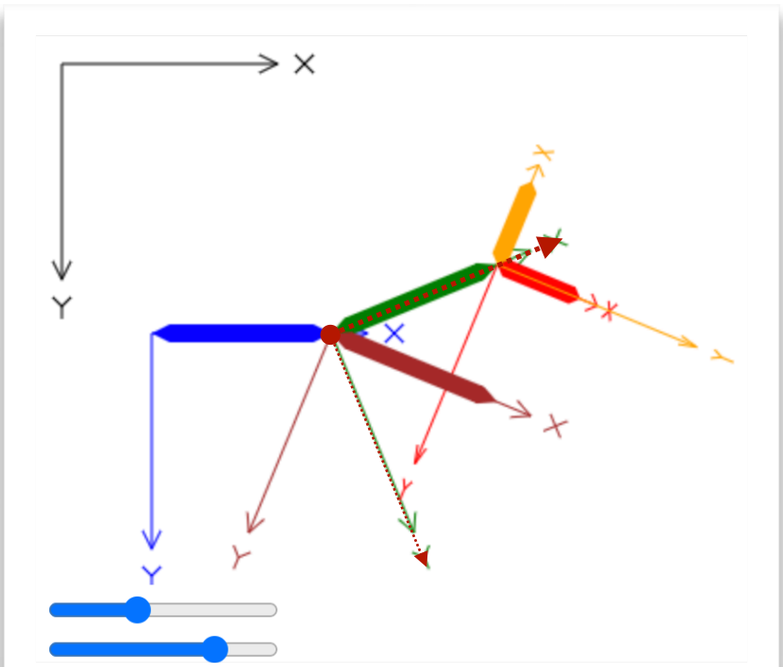
JavaScript

```
[...]
  // 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)
[...]
```

User-maintained stack

jsbin.com/zimaqos

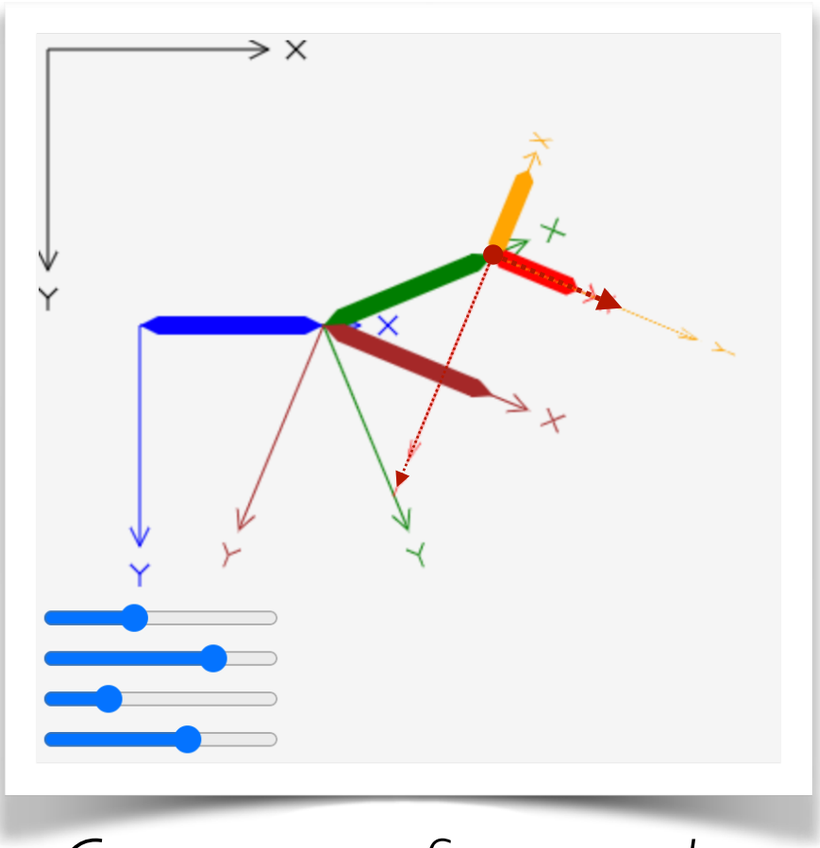
Week4/Demo1



demo.js

```
[...]

    var Tblue_to_canvas = mat3.create();
    mat3.fromTranslation(Tblue_to_canvas,[50,150]);
    mat3.multiply(stack[0],stack[0],Tblue_to_canvas);
     linkage("blue");

    stack.unshift(mat3.clone(stack[0])); // "save" (note: you *need* to clone)
     var Tgreen_to_blue = mat3.create();
    mat3.fromTranslation(Tgreen_to_blue,[100,0]);
    mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
    mat3.multiply(stack[0],stack[0],Tgreen_to_blue);
     linkage("green");

    stack.unshift(mat3.clone(stack[0])); // "save"
     var Tred_to_green = mat3.create();
    mat3.fromTranslation(Tred_to_green,[100,0]);
    mat3.rotate(Tred_to_green,Tred_to_green,phi1);
    mat3.scale(Tred_to_green,Tred_to_green,[0.5,1]);
    mat3.multiply(stack[0],stack[0],Tred_to_green);
     linkage("red");
     stack.shift();          // "restore"

    stack.unshift(mat3.clone(stack[0])); // "save"
    var Torange_to_green = mat3.create();
    mat3.fromTranslation(Torange_to_green,[100,0]);
    mat3.rotate(Torange_to_green,Torange_to_green,phi2);
    mat3.scale(Torange_to_green,Torange_to_green,[0.5,1]);
    mat3.multiply(stack[0],stack[0],Torange_to_green);
     linkage("orange");
     stack.shift();          // "restore"
     stack.shift();          // "restore"

    stack.unshift(mat3.clone(stack[0])); // "save"
     var Tbrown_to_blue = mat3.create();
    mat3.fromTranslation(Tbrown_to_blue,[100,0]);
    mat3.rotate(Tbrown_to_blue,Tbrown_to_blue,theta2);
    mat3.multiply(stack[0],stack[0],Tbrown_to_blue);
     linkage("brown");
     stack.shift();          // “restore"

[...]
```

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

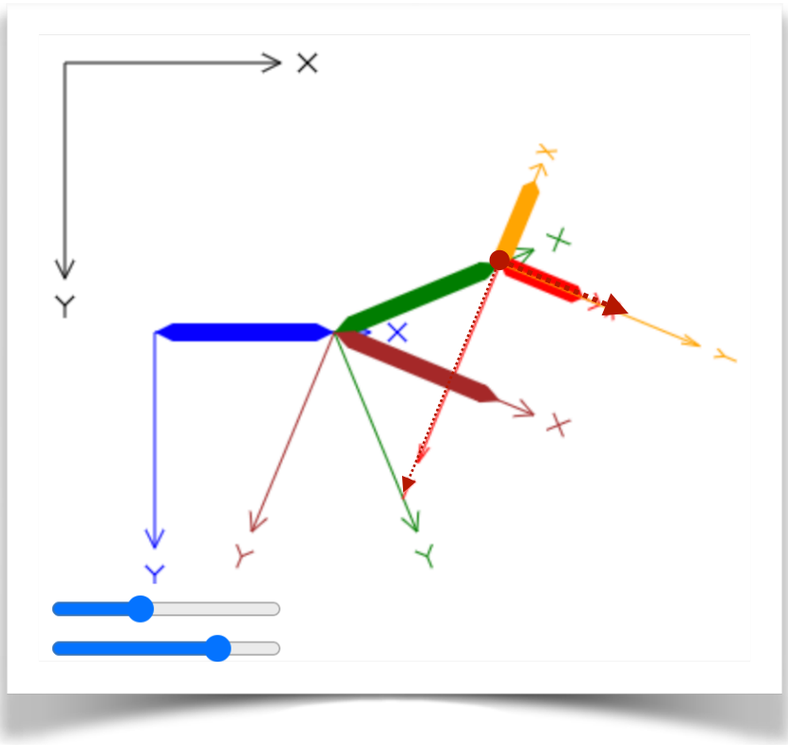
JavaScript

```
[...]
  // 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)
[...]
```

User-maintained stack

jsbin.com/zimaqos

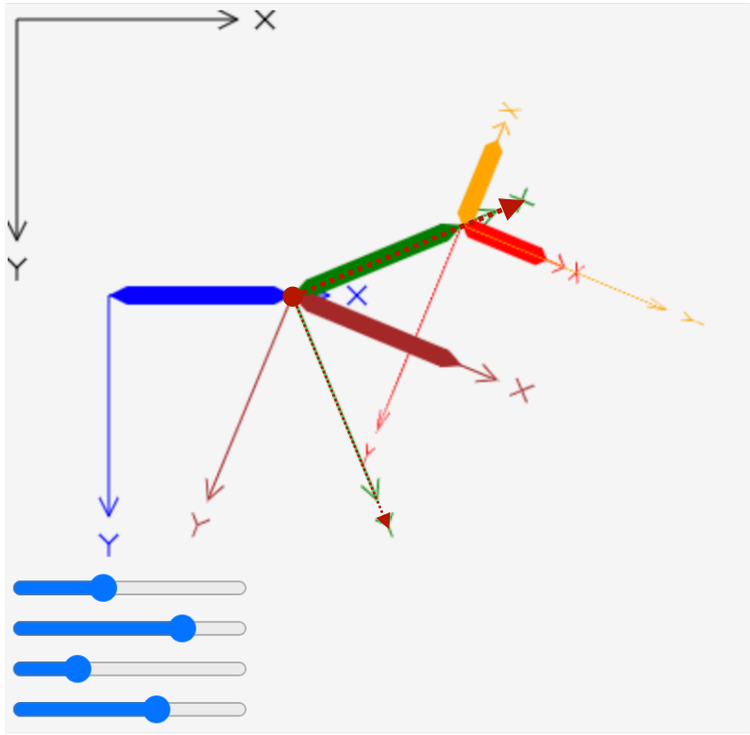
Week4/Demo1



demo.js

```
[...]

   var Tblue_to_canvas = mat3.create();
   mat3.fromTranslation(Tblue_to_canvas,[50,150]);
   mat3.multiply(stack[0],stack[0],Tblue_to_canvas);
    linkage("blue");

   stack.unshift(mat3.clone(stack[0])); // "save" (note: you *need* to clone)
    var Tgreen_to_blue = mat3.create();
   mat3.fromTranslation(Tgreen_to_blue,[100,0]);
   mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
   mat3.multiply(stack[0],stack[0],Tgreen_to_blue);
    linkage("green");

   stack.unshift(mat3.clone(stack[0])); // "save"
    var Tred_to_green = mat3.create();
   mat3.fromTranslation(Tred_to_green,[100,0]);
   mat3.rotate(Tred_to_green,Tred_to_green,phi1);
   mat3.scale(Tred_to_green,Tred_to_green,[0.5,1]);
   mat3.multiply(stack[0],stack[0],Tred_to_green);
    linkage("red");
    stack.shift();          // "restore"

   stack.unshift(mat3.clone(stack[0])); // "save"
   var Torange_to_green = mat3.create();
   mat3.fromTranslation(Torange_to_green,[100,0]);
   mat3.rotate(Torange_to_green,Torange_to_green,phi2);
   mat3.scale(Torange_to_green,Torange_to_green,[0.5,1]);
   mat3.multiply(stack[0],stack[0],Torange_to_green);
    linkage("orange");
    stack.shift();          // "restore"
    stack.shift();          // "restore"

   stack.unshift(mat3.clone(stack[0])); // "save"
    var Tbrown_to_blue = mat3.create();
   mat3.fromTranslation(Tbrown_to_blue,[100,0]);
   mat3.rotate(Tbrown_to_blue,Tbrown_to_blue,theta2);
   mat3.multiply(stack[0],stack[0],Tbrown_to_blue);
    linkage("brown");
    stack.shift();          // “restore"

[...]
```

Canvas transform stack



Canvas transform stack

canvas-to-blue

blue-to-green

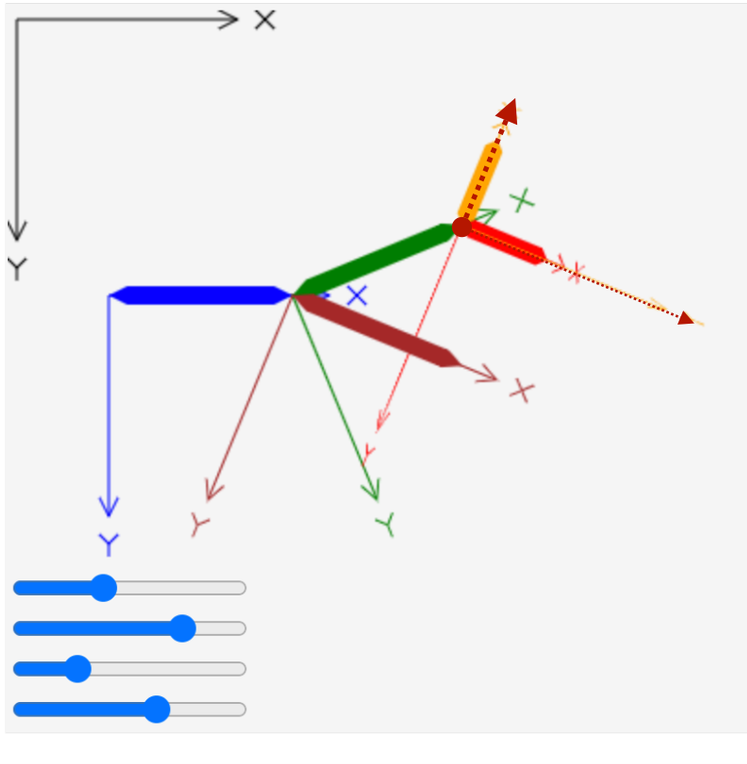
canvas-to-blue

jsbin.com/wuyutirife

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

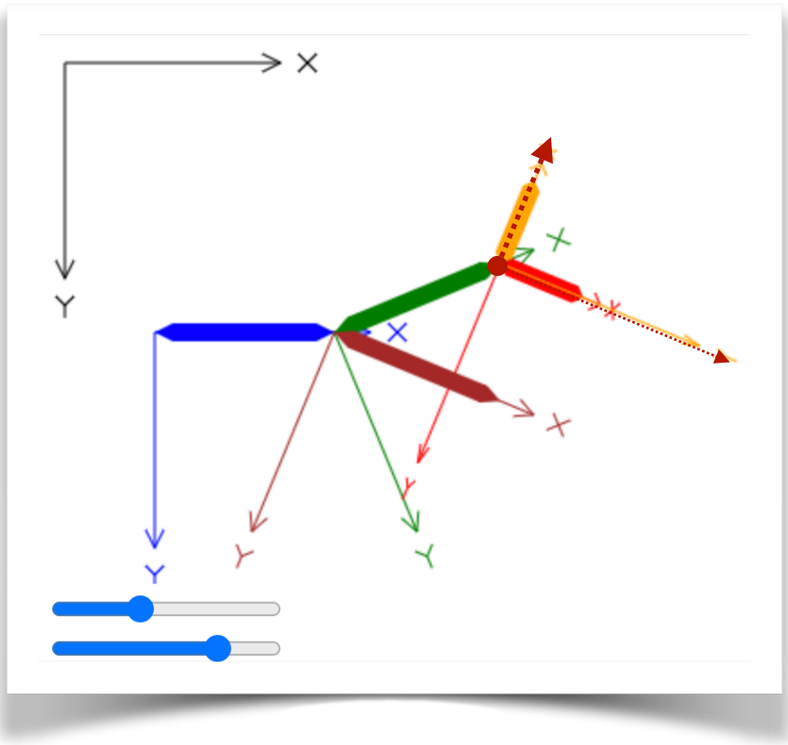
JavaScript

```
[...]
  // 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)
[...]
```

User-maintained stack

jsbin.com/zimaqos

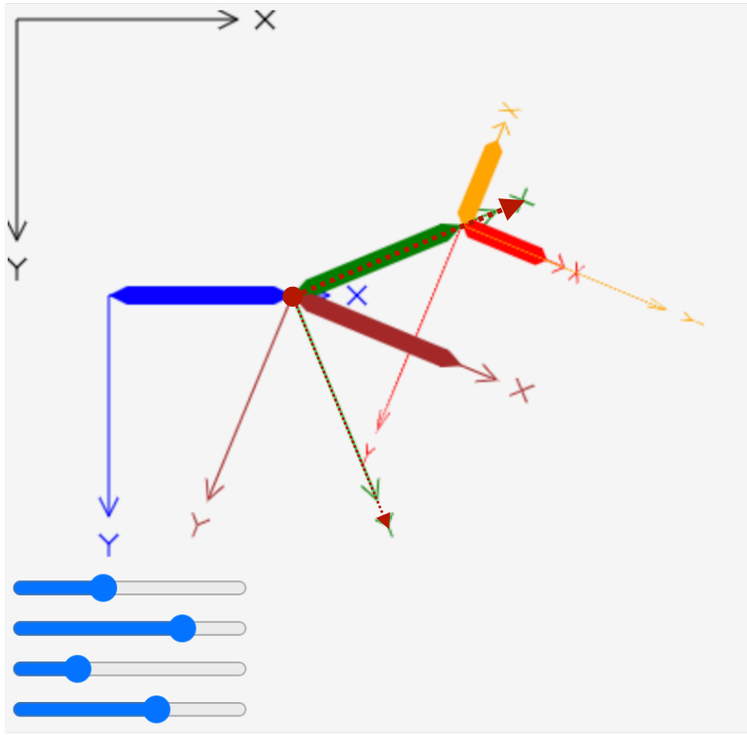
Week4/Demo1



demo.js

```
[...]

    var Tblue_to_canvas = mat3.create();
    mat3.fromTranslation(Tblue_to_canvas,[50,150]);
    mat3.multiply(stack[0],stack[0],Tblue_to_canvas);
     linkage("blue");

    stack.unshift(mat3.clone(stack[0])); // "save" (note: you *need* to clone)
     var Tgreen_to_blue = mat3.create();
    mat3.fromTranslation(Tgreen_to_blue,[100,0]);
    mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
    mat3.multiply(stack[0],stack[0],Tgreen_to_blue);
     linkage("green");

    stack.unshift(mat3.clone(stack[0])); // "save"
     var Tred_to_green = mat3.create();
    mat3.fromTranslation(Tred_to_green,[100,0]);
    mat3.rotate(Tred_to_green,Tred_to_green,phi1);
    mat3.scale(Tred_to_green,Tred_to_green,[0.5,1]);
    mat3.multiply(stack[0],stack[0],Tred_to_green);
     linkage("red");
     stack.shift();          // "restore"

    stack.unshift(mat3.clone(stack[0])); // "save"
    var Torange_to_green = mat3.create();
    mat3.fromTranslation(Torange_to_green,[100,0]);
    mat3.rotate(Torange_to_green,Torange_to_green,phi2);
    mat3.scale(Torange_to_green,Torange_to_green,[0.5,1]);
    mat3.multiply(stack[0],stack[0],Torange_to_green);
     linkage("orange");
     stack.shift();          // "restore"
     stack.shift();          // "restore"

    stack.unshift(mat3.clone(stack[0])); // "save"
     var Tbrown_to_blue = mat3.create();
    mat3.fromTranslation(Tbrown_to_blue,[100,0]);
    mat3.rotate(Tbrown_to_blue,Tbrown_to_blue,theta2);
    mat3.multiply(stack[0],stack[0],Tbrown_to_blue);
     linkage("brown");
     stack.shift();          // “restore"

[...]
```

Canvas transform stack



Canvas transform stack

canvas-to-blue

blue-to-green

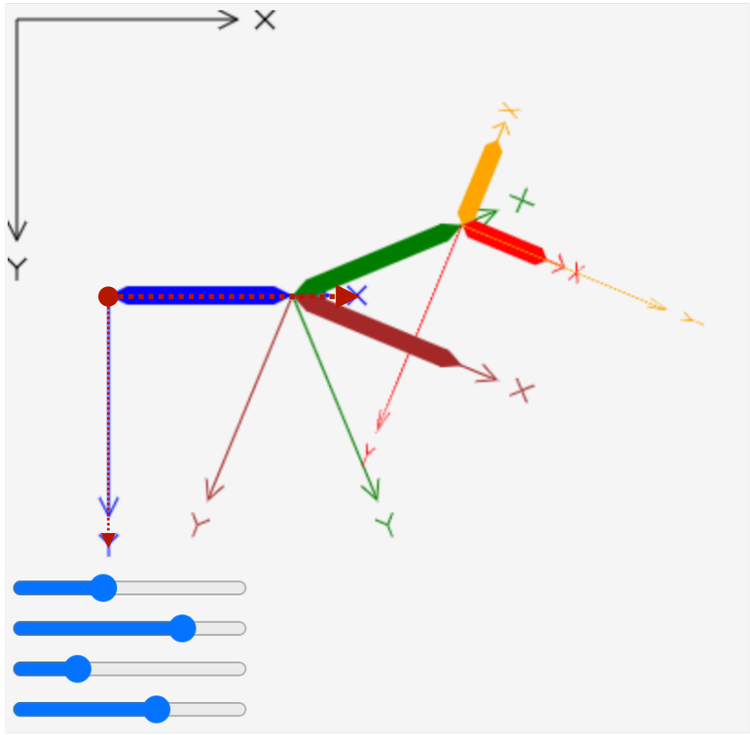
canvas-to-blue

jsbin.com/wuyutirife

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

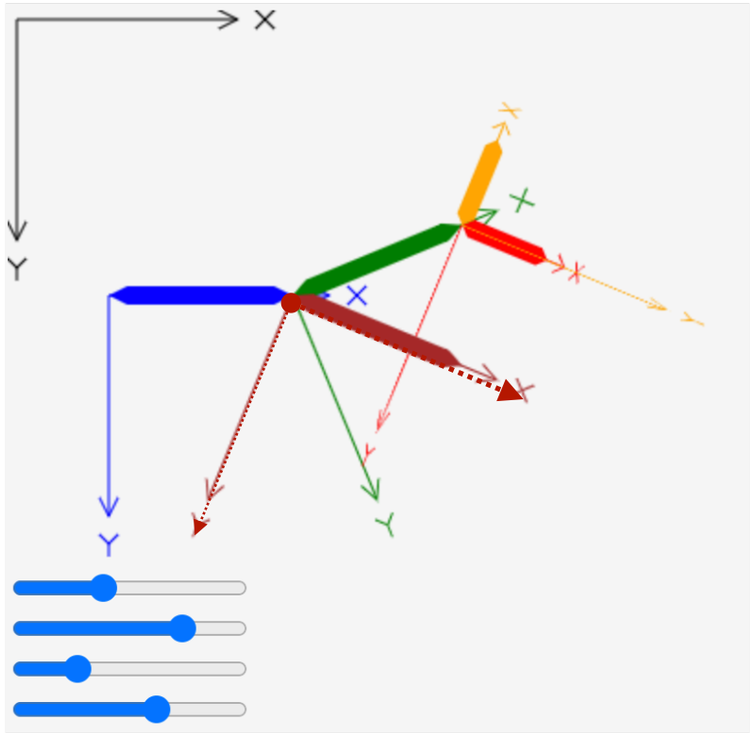
canvas-to-blue

jsbin.com/wuyutirife

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

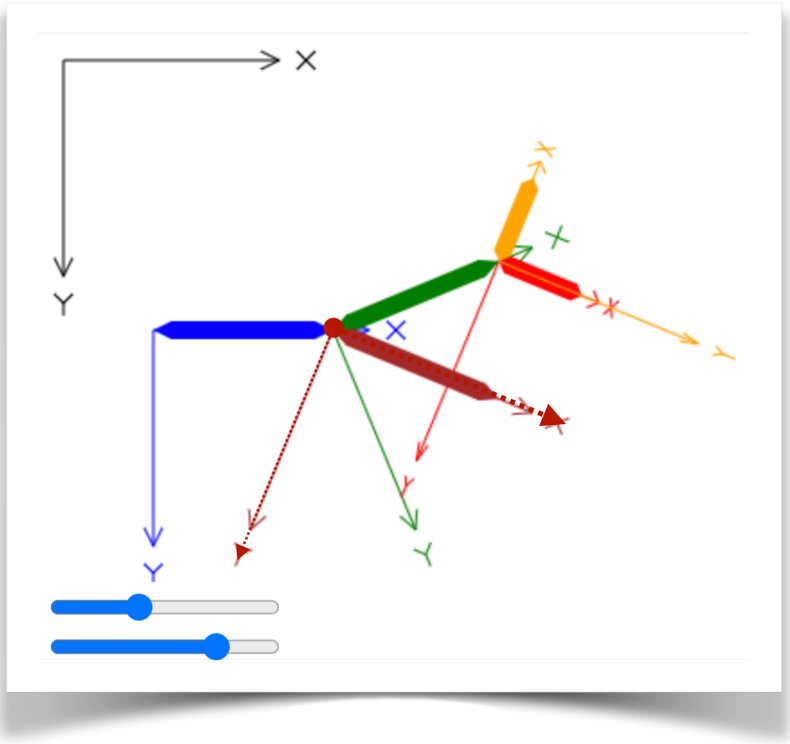
JavaScript

```
[...]
  // 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)
[...]
```

User-maintained stack

jsbin.com/zimaqos

Week4/Demo1



demo.js

```
[...]

    var Tblue_to_canvas = mat3.create();
    mat3.fromTranslation(Tblue_to_canvas,[50,150]);
    mat3.multiply(stack[0],stack[0],Tblue_to_canvas);
     linkage("blue");

    stack.unshift(mat3.clone(stack[0])); // "save" (note: you *need* to clone)
     var Tgreen_to_blue = mat3.create();
    mat3.fromTranslation(Tgreen_to_blue,[100,0]);
    mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
    mat3.multiply(stack[0],stack[0],Tgreen_to_blue);
     linkage("green");

    stack.unshift(mat3.clone(stack[0])); // "save"
     var Tred_to_green = mat3.create();
    mat3.fromTranslation(Tred_to_green,[100,0]);
    mat3.rotate(Tred_to_green,Tred_to_green,phi1);
    mat3.scale(Tred_to_green,Tred_to_green,[0.5,1]);
    mat3.multiply(stack[0],stack[0],Tred_to_green);
     linkage("red");
     stack.shift();          // "restore"

    stack.unshift(mat3.clone(stack[0])); // "save"
    var Torange_to_green = mat3.create();
    mat3.fromTranslation(Torange_to_green,[100,0]);
    mat3.rotate(Torange_to_green,Torange_to_green,phi2);
    mat3.scale(Torange_to_green,Torange_to_green,[0.5,1]);
    mat3.multiply(stack[0],stack[0],Torange_to_green);
     linkage("orange");
     stack.shift();          // "restore"
     stack.shift();          // "restore"

    stack.unshift(mat3.clone(stack[0])); // "save"
     var Tbrown_to_blue = mat3.create();
    mat3.fromTranslation(Tbrown_to_blue,[100,0]);
    mat3.rotate(Tbrown_to_blue,Tbrown_to_blue,theta2);
    mat3.multiply(stack[0],stack[0],Tbrown_to_blue);
     linkage("brown");
     stack.shift();          // “restore"

[...]
```