

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

Tuesday September 28th 2021

Administrative stuff

- Your 2nd Programming Assignment has been posted
 - Due *Thursday Oct 6th*
- 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

Today's lecture

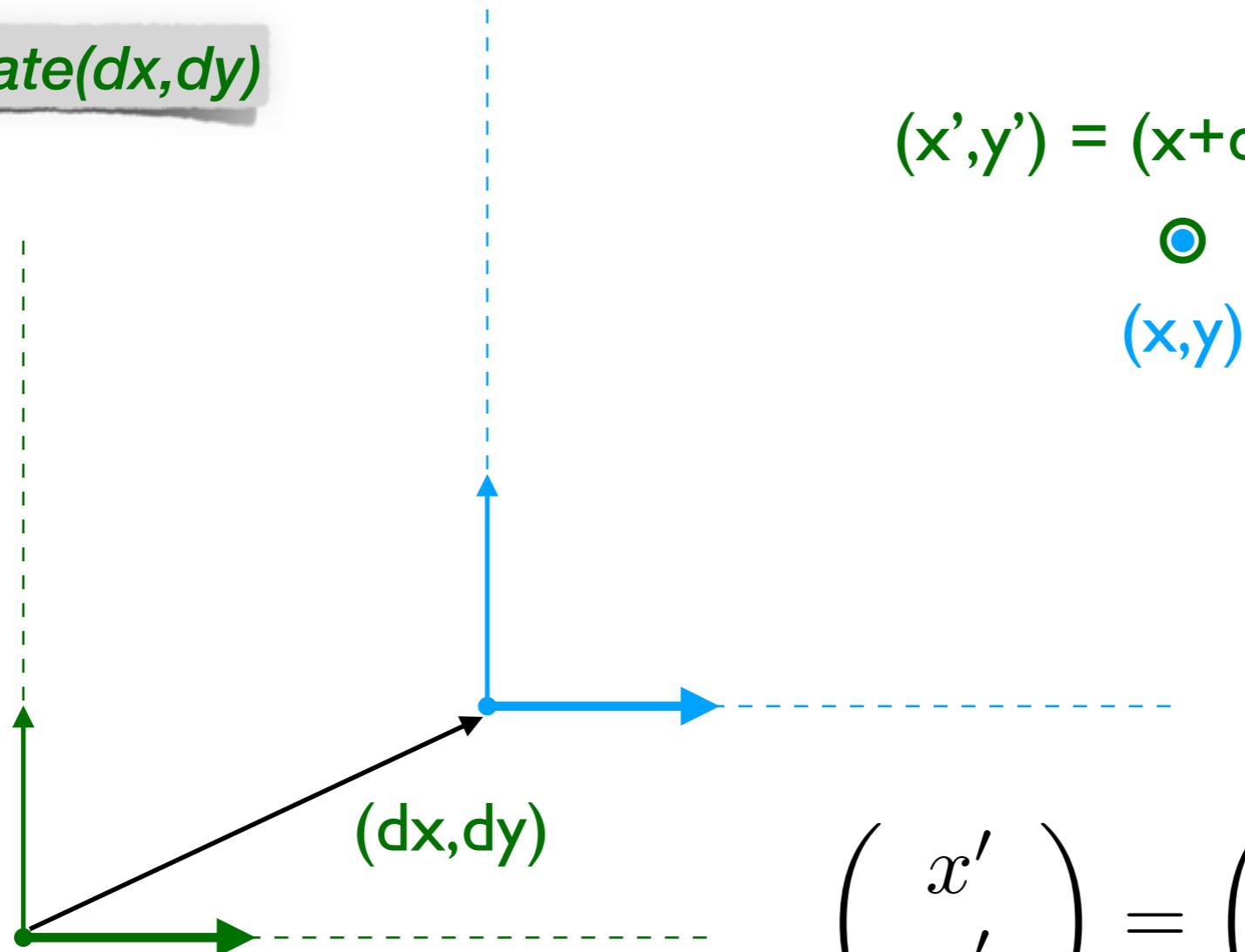
- Mathematical representation of elementary transforms
- Algebra of transform compositions and point/vector transformations
- An introductory look at homogeneous coordinates
- Implementation of hierarchical modeling and transform compositions via linear algebra libraries

Transforms and their semantics

jsbin.com/hozeyar

Week3/Demo0

Translate(dx, dy)



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

⊕
 (x, y)

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

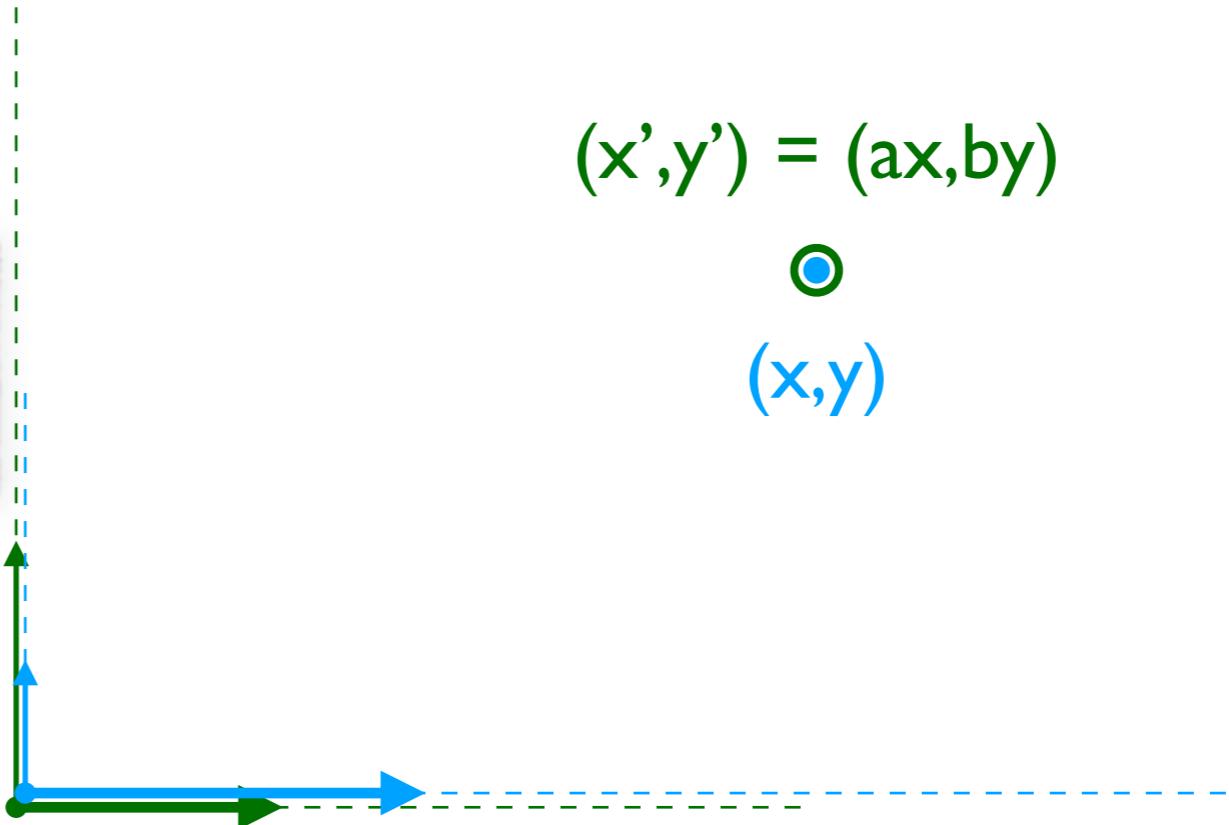
Transforms and their semantics

jsbin.com/hozeyar

Week3/Demo0

Scale(a,b)

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



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



$$(x,y)$$

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

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

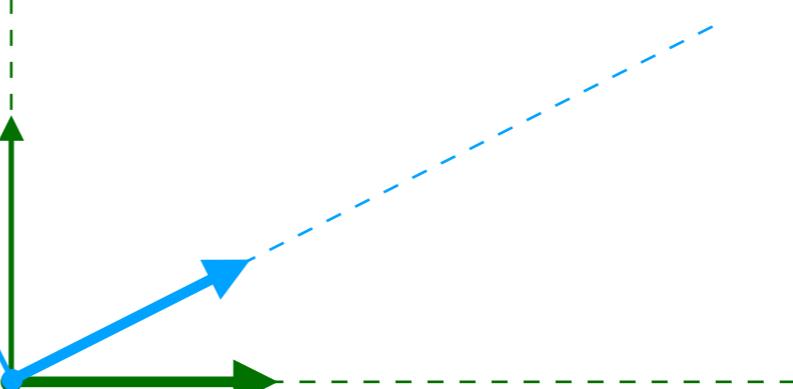
Transforms and their semantics

jsbin.com/hozeyar

Week3/Demo0

Rotate(θ)

(x' , y') = ?? (rotation by angle 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)
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

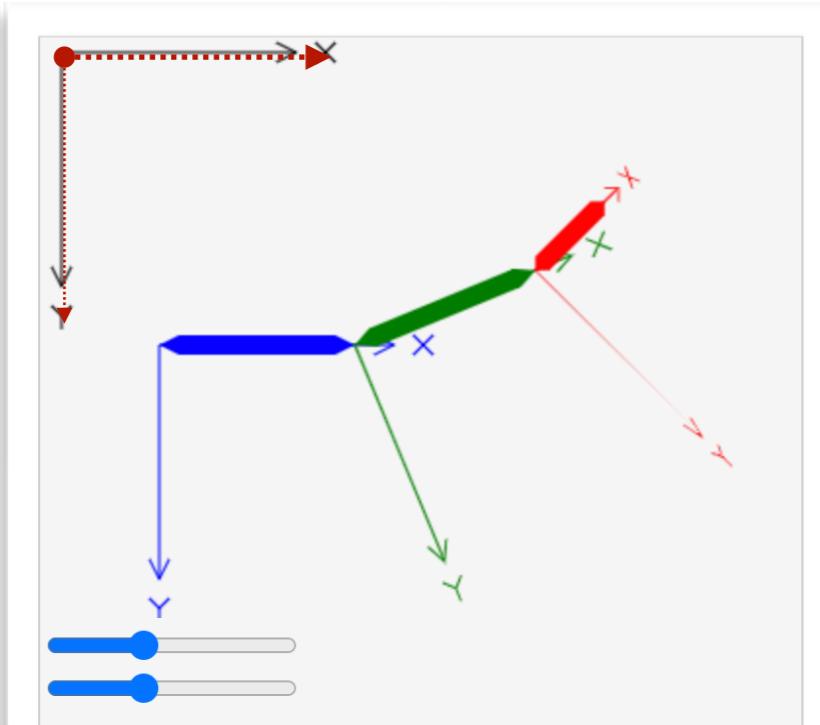
Rotate(θ)
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

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?



JavaScript

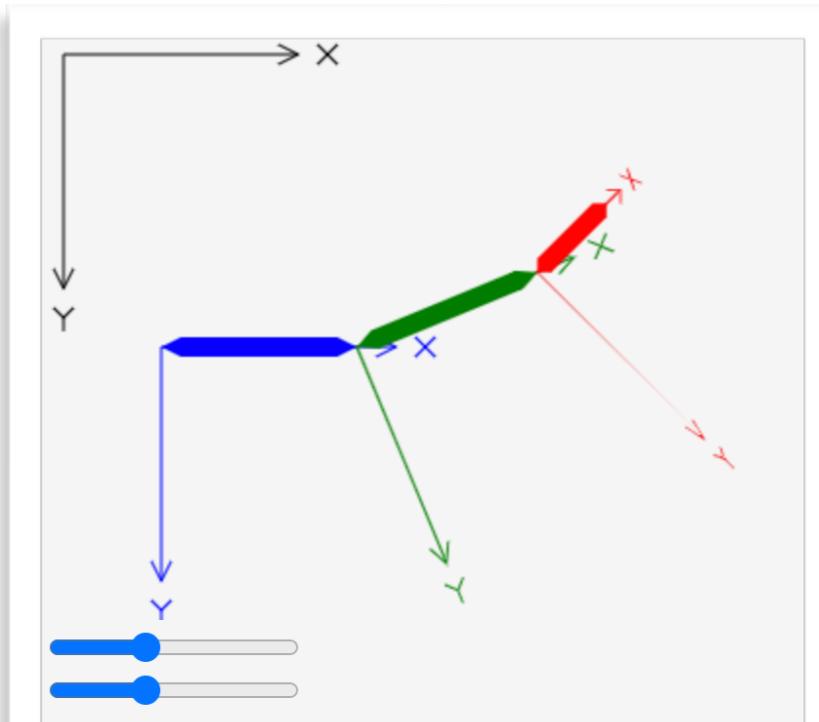
[...]

```
// 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?



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)*



Translate
(50,50)

Translate
(100,0)

Rotate
(θ_1)

Translate
(100,0)

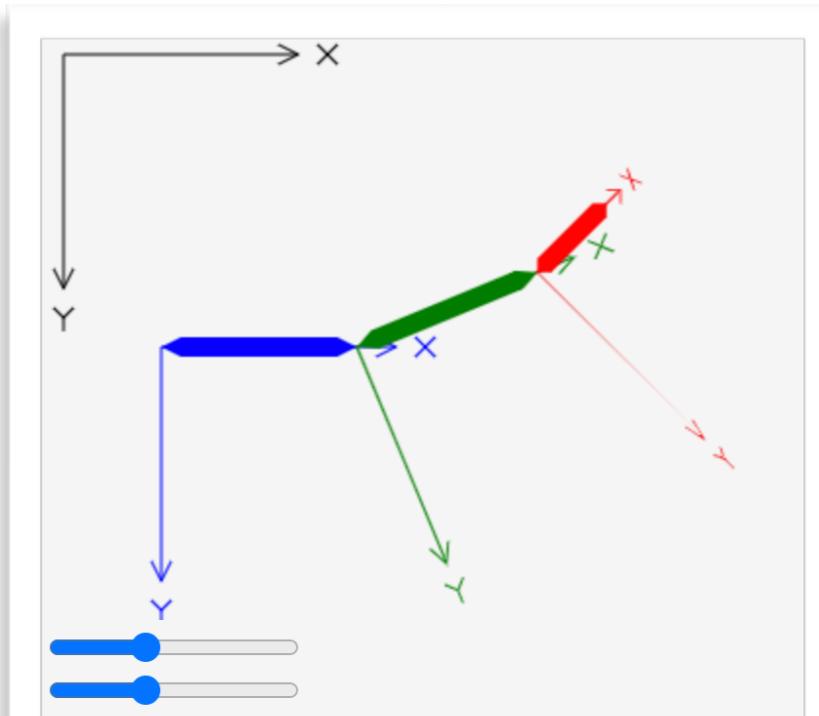
Rotate
(φ_1)

Scale
(0.5,1)



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?

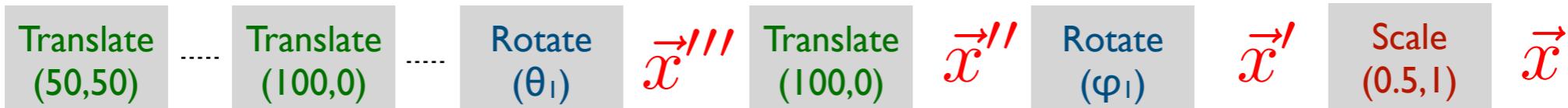


JavaScript

[...]

```
// 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");
```

[...]



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

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

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

Two successive transforms

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

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

Combined transform
is also affine ...

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

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

... 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

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

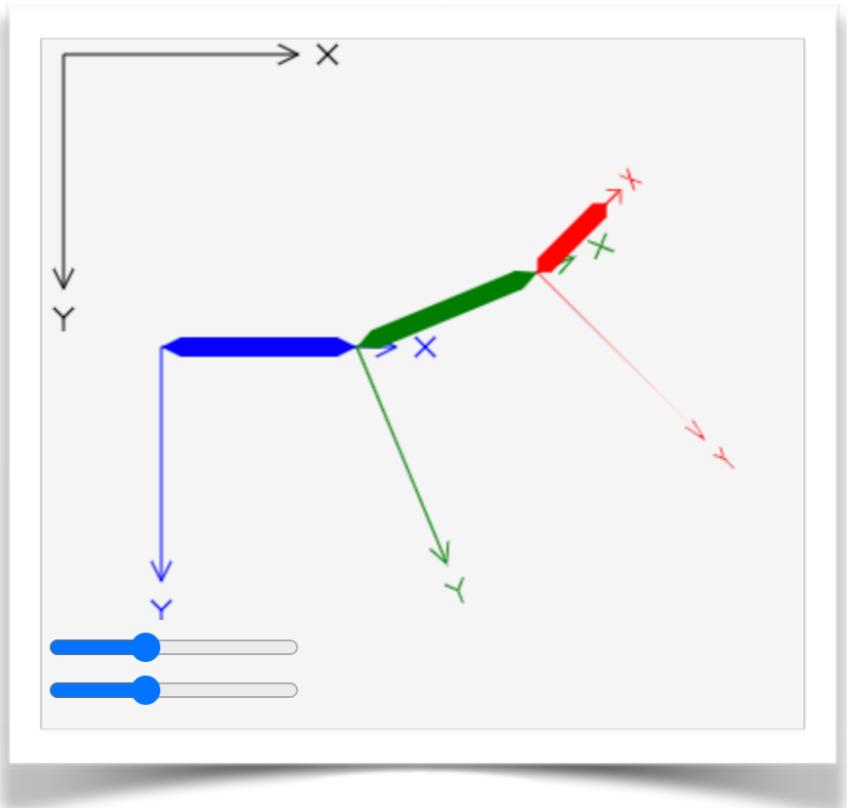
$$\begin{pmatrix} x'' \\ y'' \\ 1 \end{pmatrix} = \mathbf{T}_2 \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$$

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

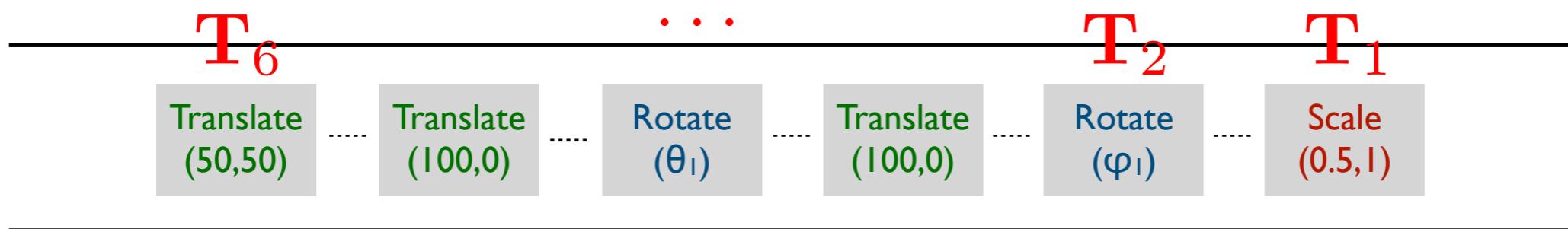
(contrasted with this ...)

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

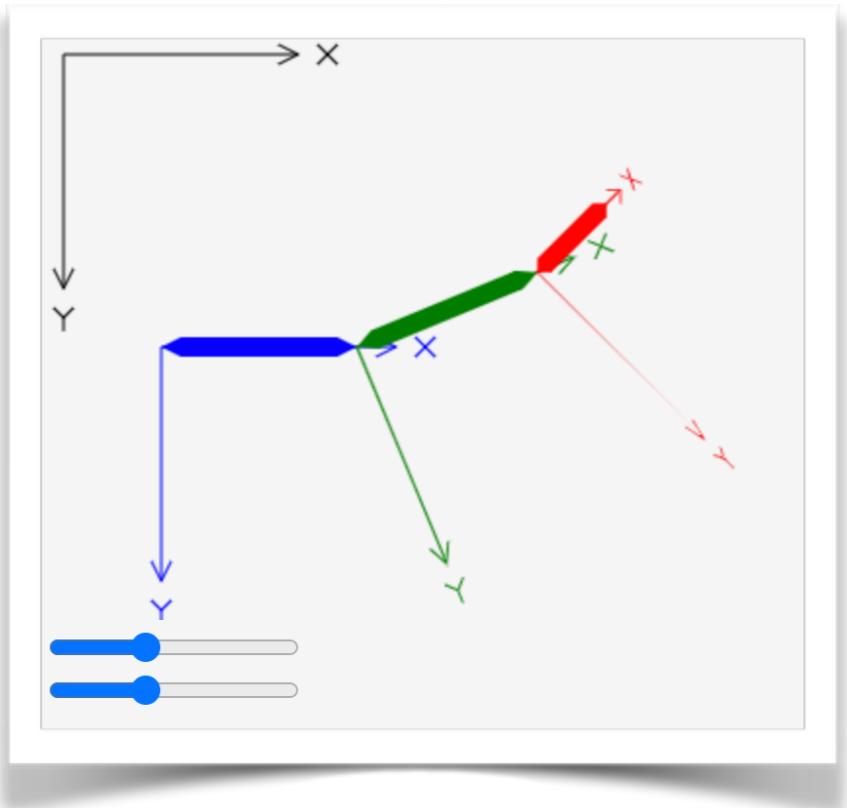
How are transforms combined?



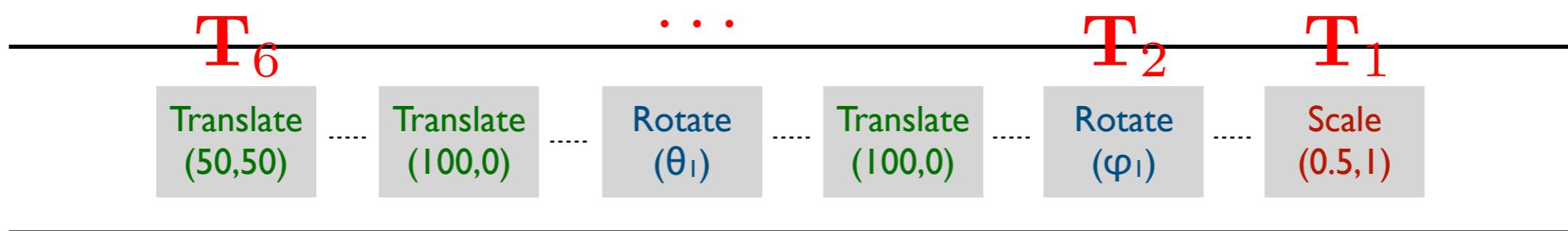
And this works with longer transform chains!



How are transforms combined?



And this works with longer transform chains!



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

(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)

demo.js

[...]

```

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

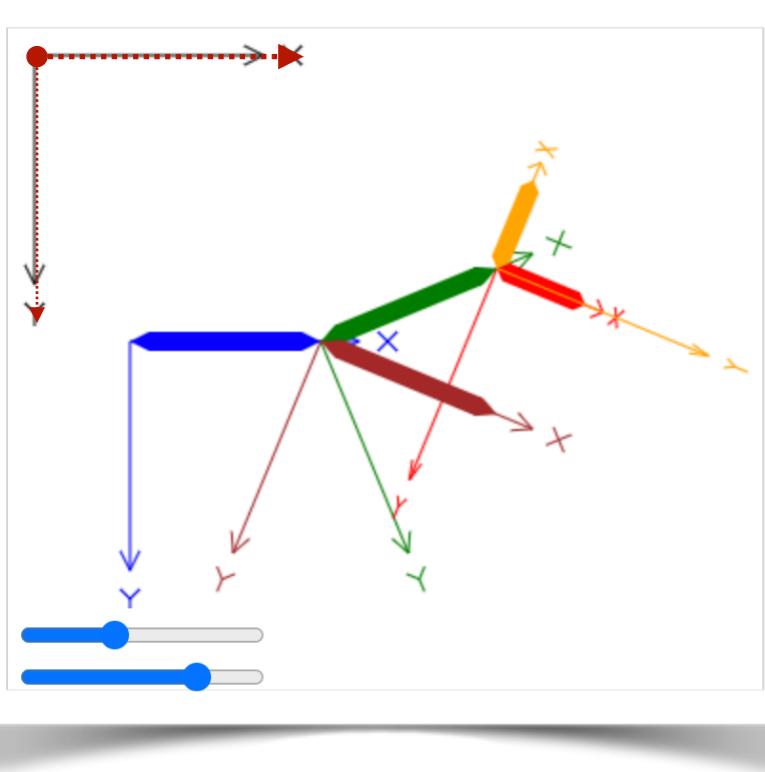
function lineToTx(x,y,Tx)
{var res=vec2.create(); vec2.transformMat3(res,[x,y],Tx);
 context.lineTo(res[0],res[1]);}

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();
}

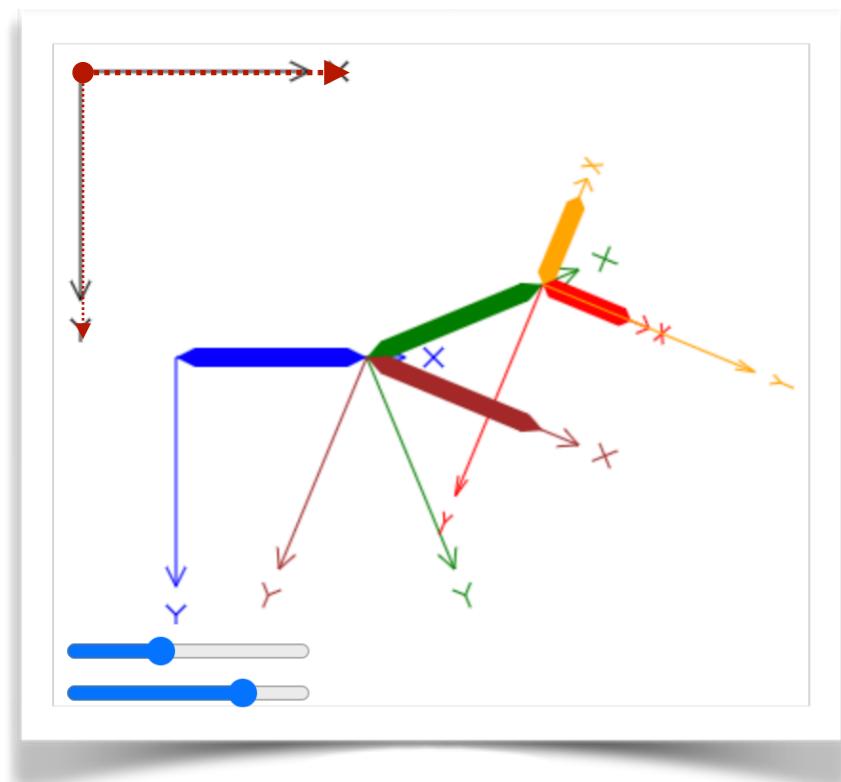
```

[...]



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

Transforms in algebraic form (via *glmatrix*)



*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);
}

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)

demo.js

[...]

```

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

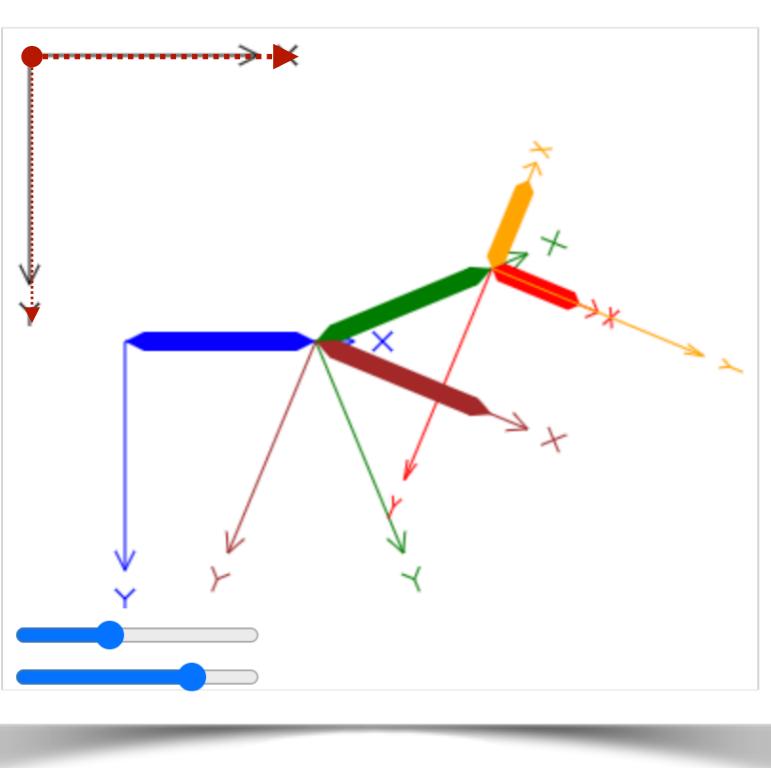
function lineToTx(x,y,Tx)
{var res=vec2.create(); vec2.transformMat3(res,[x,y],Tx);
 context.lineTo(res[0],res[1]);}

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();
}

```

[...]



We apply the *input transformation* explicitly via a call to the *glMatrix transformMat3()* method within the *vec2* module
[\(vec2 documentation\)](#)

Transforms in algebraic form (via glmatrix)

demo.js

[...]

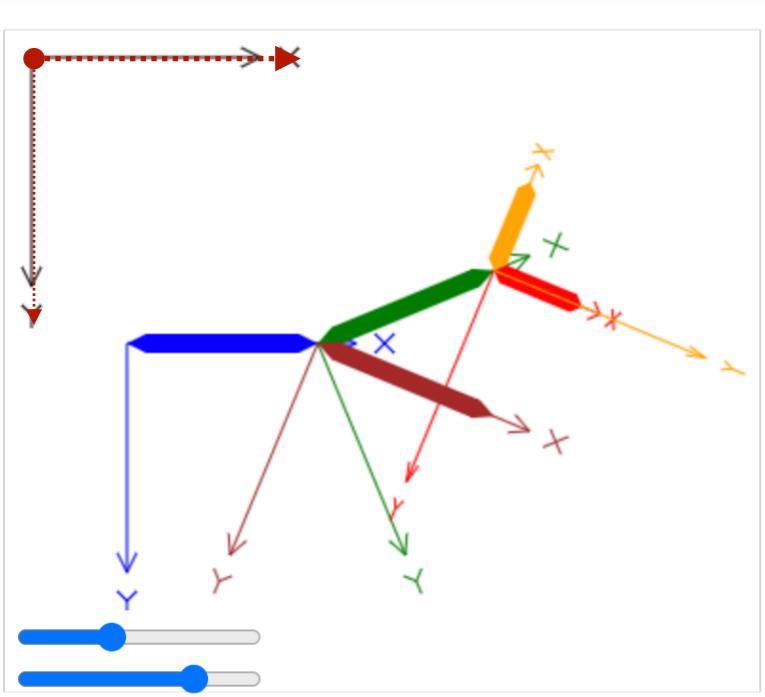
```
function moveToTx(x,y,Tx)
{var res=vec2.create(); vec2.transformMat3(res,[x,y],Tx);
 context.moveTo(res[0],res[1]);}
```

```
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);
}
```

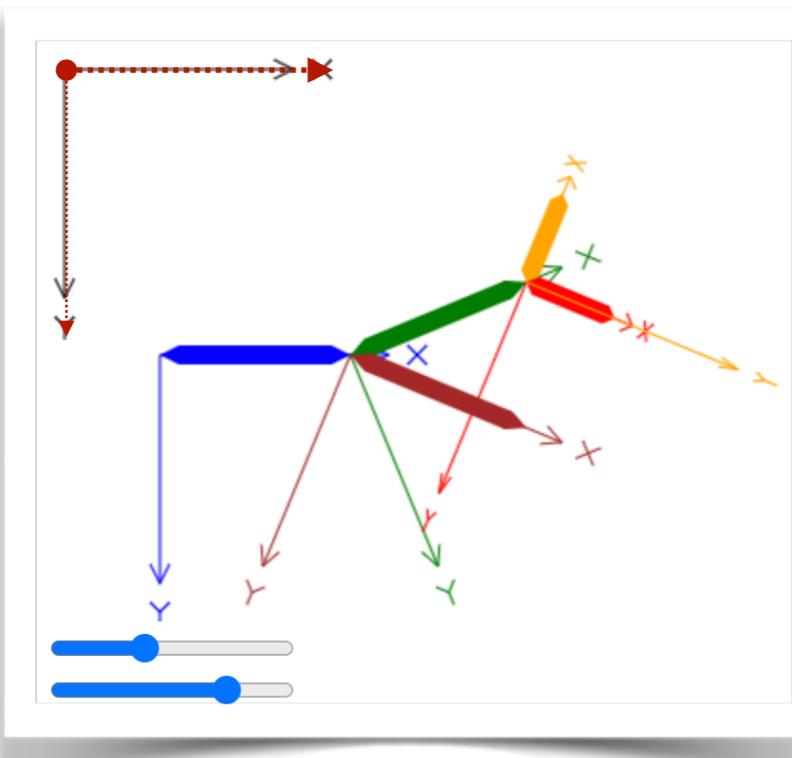
```
function axes(color,Tx) {
  context.strokeStyle=color;
  context.beginPath();
  // Axes
  moveToTx(120,0,Tx);lineToTx(0,0,Tx);lineToTx(0,120,Tx);
  // Arrowheads
  moveToTx(110,5,Tx);lineToTx(120,0,Tx);lineToTx(110,-5,Tx);
  moveToTx(5,110,Tx);lineToTx(0,120,Tx);lineToTx(-5,110,Tx);
  // X-label
  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();
}
```

[...]



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

Transforms in algebraic form (via *glmatrix*)



demo.js

[...]

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

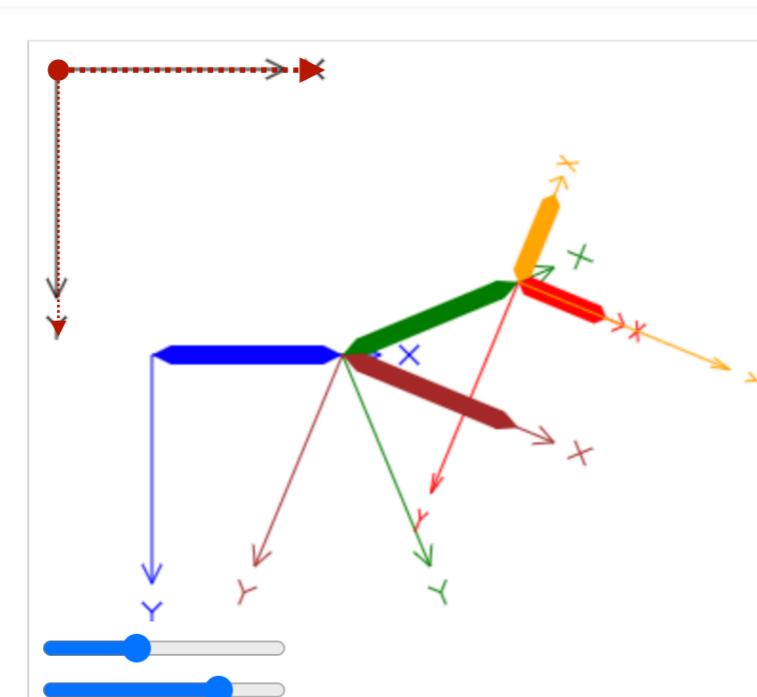
var Tgreen_to_blue = mat3.create();
mat3.fromTranslation(Tgreen_to_blue,[100,0]);
mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
var Tgreen_to_canvas = mat3.create();
mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
linkage("green",Tgreen_to_canvas);

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)



$$T_{\text{blue2canvas}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

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);

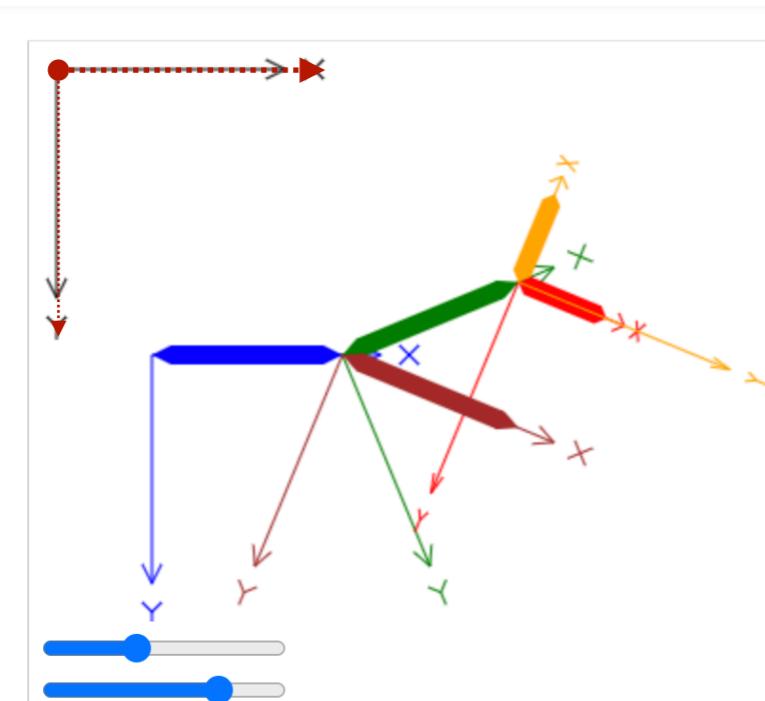
var Tred_to_green = mat3.create();
mat3.fromTranslation(Tred_to_green, [100, 0]);
mat3.rotate(Tred_to_green, Tred_to_green, phi1);
mat3.scale(Tred_to_green, Tred_to_green, [0.5, 1]);
var Tred_to_canvas = mat3.create();
mat3.multiply(Tred_to_canvas, Tgreen_to_canvas, Tred_to_green);
linkage("red", Tred_to_canvas);

```

Translate
(50,50)

blue-to-canvas

Transforms in algebraic form (via glmatrix)



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);

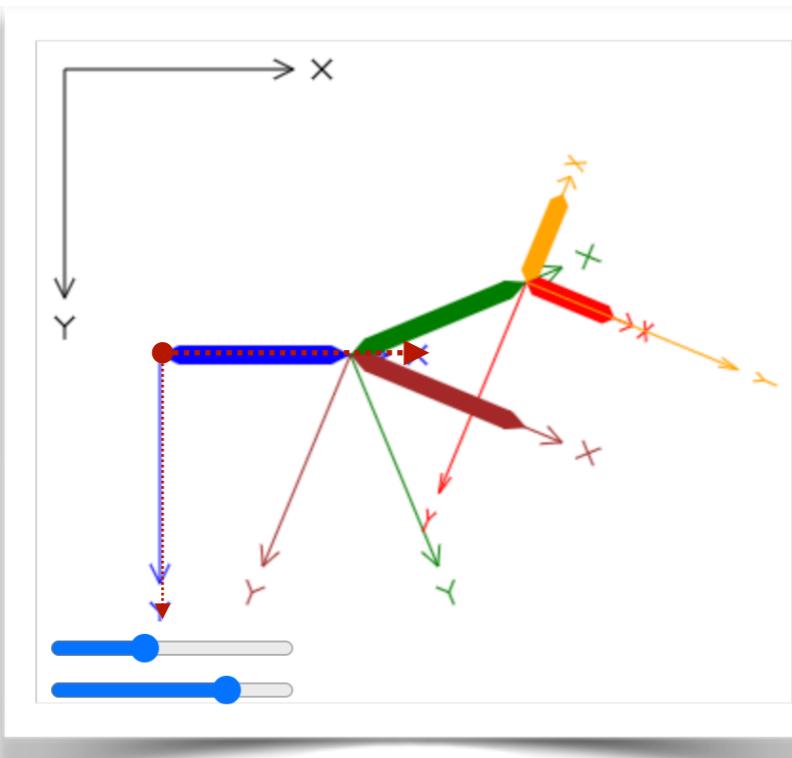
var Tred_to_green = mat3.create();
mat3.fromTranslation(Tred_to_green, [100, 0]);
mat3.rotate(Tred_to_green, Tred_to_green, phi1);
mat3.scale(Tred_to_green, Tred_to_green, [0.5, 1]);
var Tred_to_canvas = mat3.create();
mat3.multiply(Tred_to_canvas, Tgreen_to_canvas, Tred_to_green);
linkage("red", Tred_to_canvas);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

$$T_{\text{blue2canvas}} = \begin{pmatrix} 1 & 0 & 50 \\ 0 & 1 & 150 \\ 0 & 0 & 1 \end{pmatrix}$$

Translate
(50,50)

blue-to-canvas

Transforms in algebraic form (via *glmatrix*)



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);
```

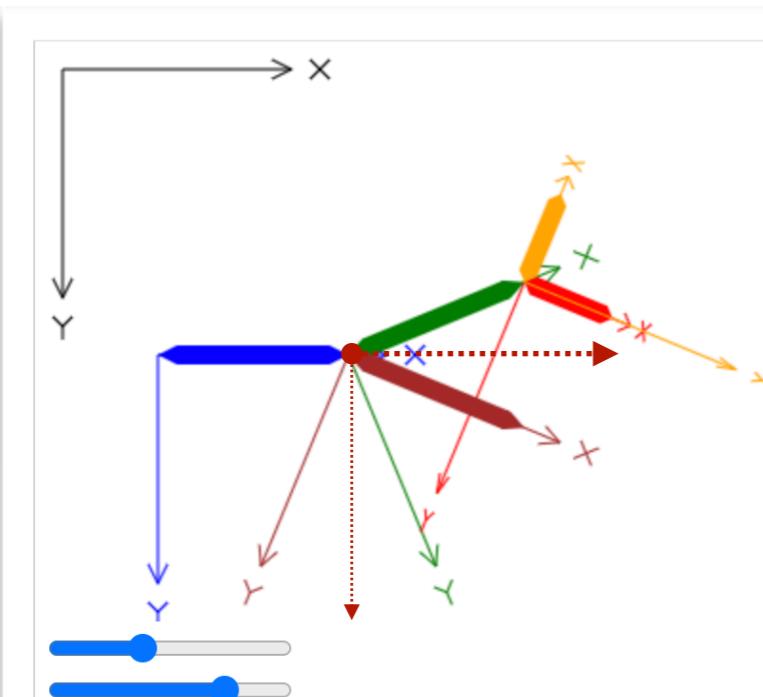
```
var Tred_to_green = mat3.create();
mat3.fromTranslation(Tred_to_green, [100, 0]);
mat3.rotate(Tred_to_green, Tred_to_green, phi1);
mat3.scale(Tred_to_green, Tred_to_green, [0.5, 1]);
var Tred_to_canvas = mat3.create();
mat3.multiply(Tred_to_canvas, Tgreen_to_canvas, Tred_to_green);
linkage("red",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

[...]

Translate
(50,50)

blue-to-canvas

Transforms in algebraic form (via glmatrix)



$$T_{\text{green2blue}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

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]);
mat3.rotate(Tred_to_green, Tred_to_green, phi1);
Tred_to_canvas = mat3.create();
ly(Tred_to_canvas, Tgreen_to_canvas, Tred_to_green);
d", Tred_to_canvas);

```

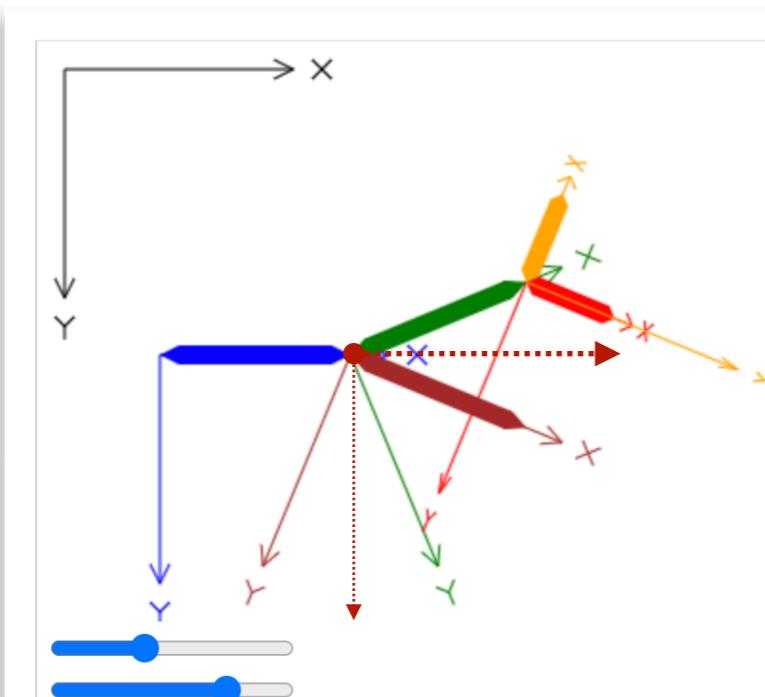
Translate
(50,50)

Translate
(100,0)

blue-to-canvas

green-to-blue

Transforms in algebraic form (via glmatrix)



$$T_{\text{green2blue}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 100 \\ 0 & 0 & 1 \end{pmatrix}$$

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);
var Tred_to_canvas = mat3.create();
mat3.multiply(Tred_to_canvas, Tgreen_to_canvas, Tred_to_green);
linkage("red", Tred_to_canvas);
```

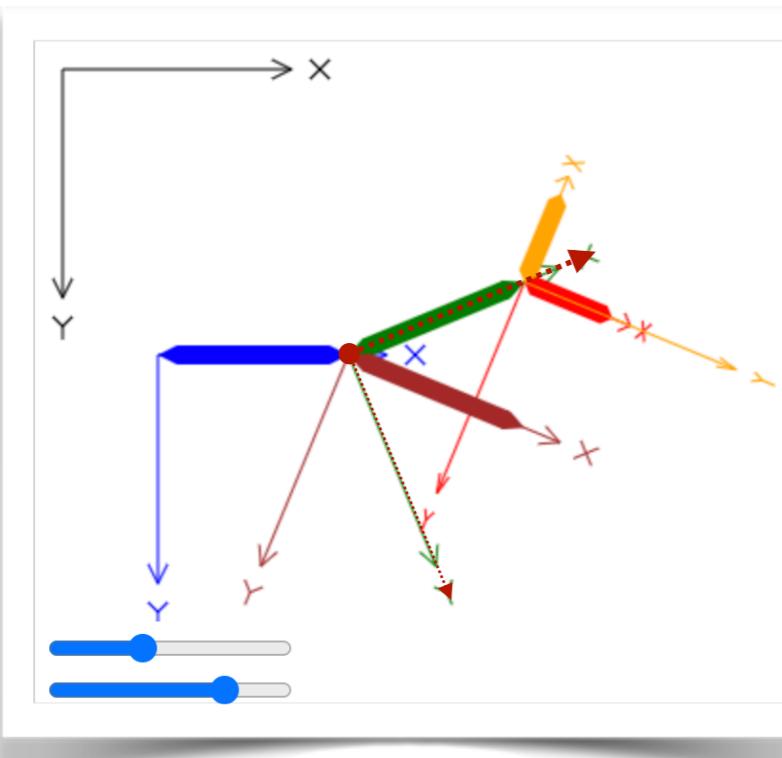
Translate
(50,50)

Translate
(100,0)

blue-to-canvas

green-to-blue

Transforms in algebraic form (via glmatrix)



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.multiply(Tred_to_green, Tred_to_green, phi1);
Tred_to_green = mat3.create();
mat3.multiply(Tred_to_green, Tgreen_to_canvas, Tred_to_green);
linkage("red", Tred_to_green);

```

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

Translate
(50,50)

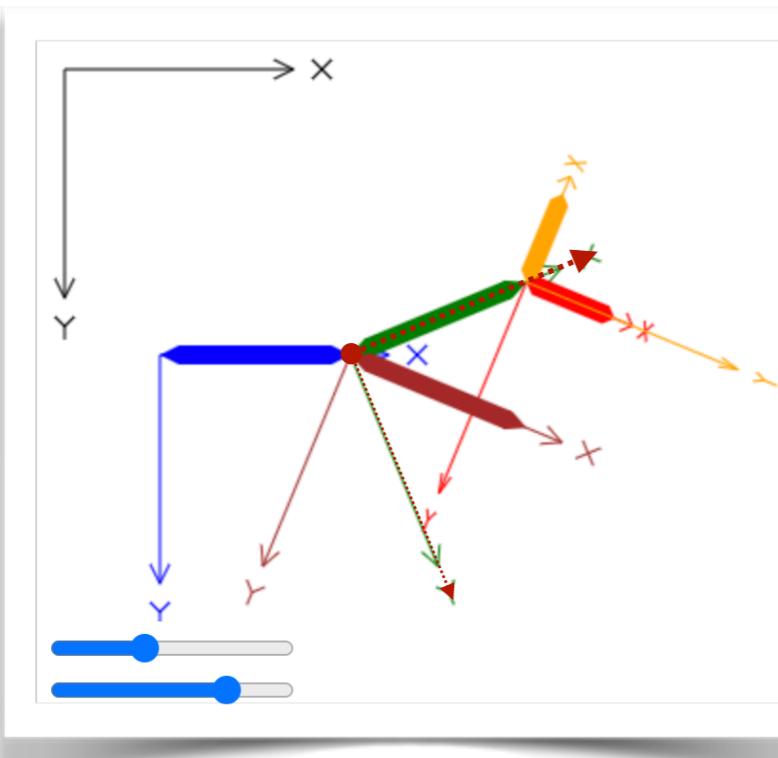
Translate
(100,0)

Rotate
(θ₁)

blue-to-canvas

green-to-blue

Transforms in algebraic form (via *glmatrix*)



demo.js

[...]

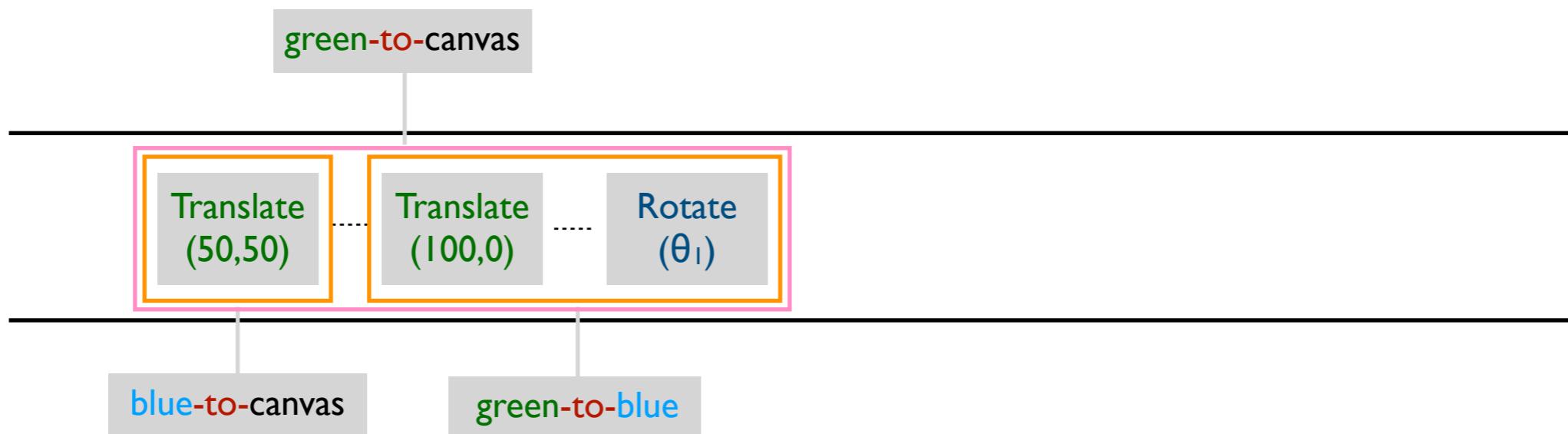
```

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

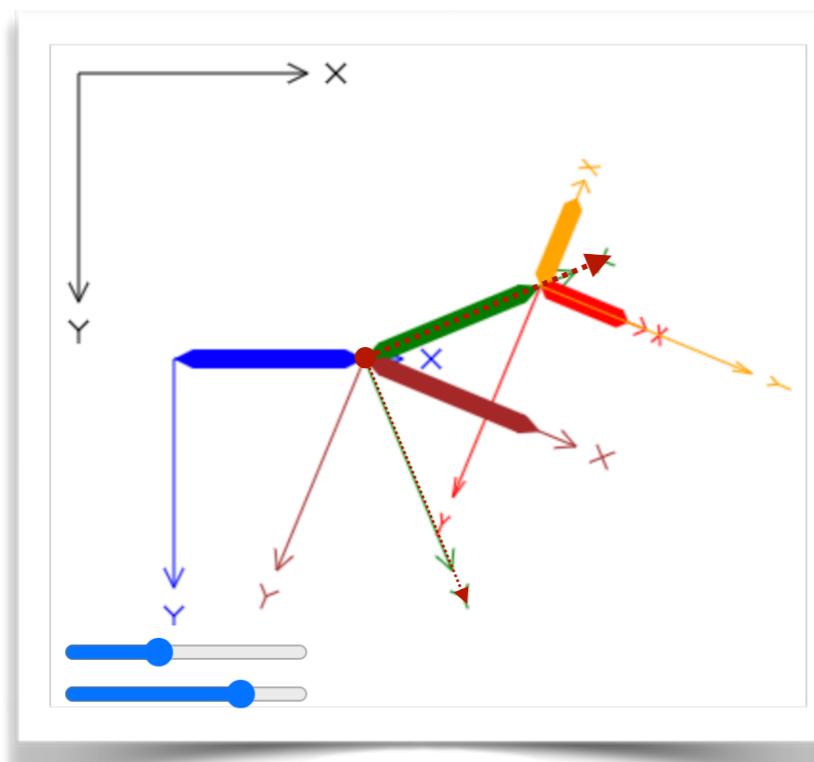
var Tgreen_to_blue = mat3.create();
mat3.fromTranslation(Tgreen_to_blue, [100, 0]);
mat3.rotate(Tgreen_to_blue, Tgreen_to_blue, theta1);
var Tgreen_to_canvas = mat3.create();
mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
linkage("green", Tgreen_to_canvas);

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

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



Transforms in algebraic form (via *glmatrix*)



demo.js

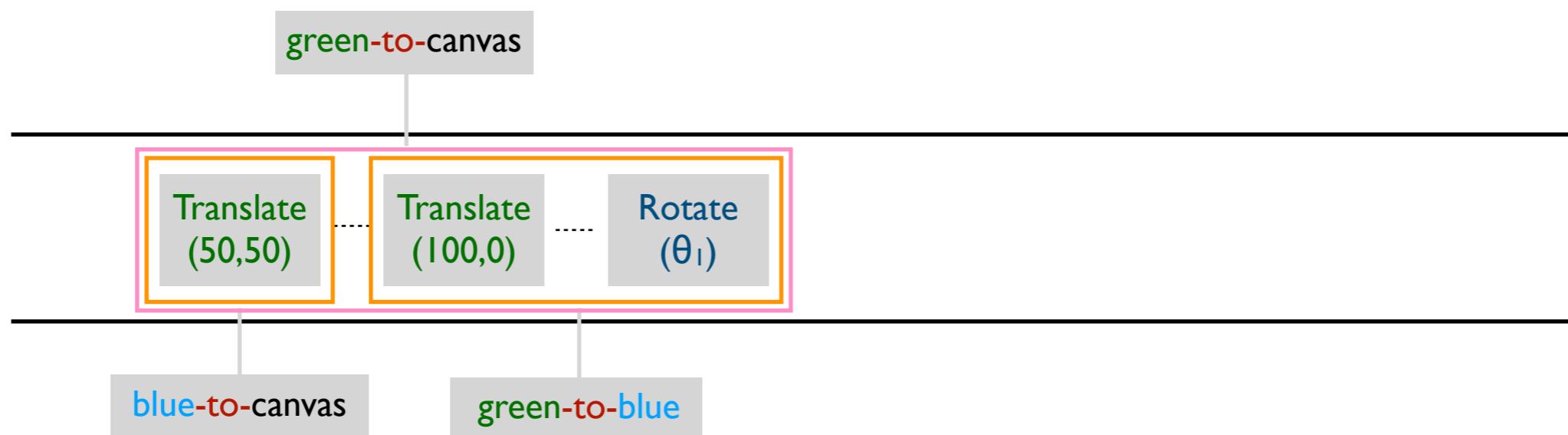
[...]

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

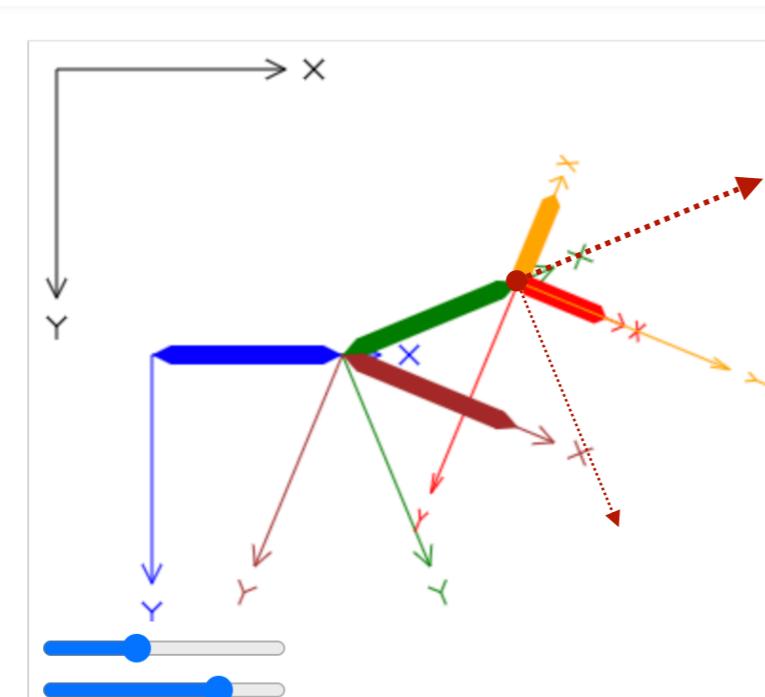
var Tgreen_to_blue = mat3.create();
mat3.fromTranslation(Tgreen_to_blue, [100, 0]);
mat3.rotate(Tgreen_to_blue, Tgreen_to_blue, theta1);
var Tgreen_to_canvas = mat3.create();
mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
linkage("green", Tgreen_to_canvas);
```

```
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*)



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

```

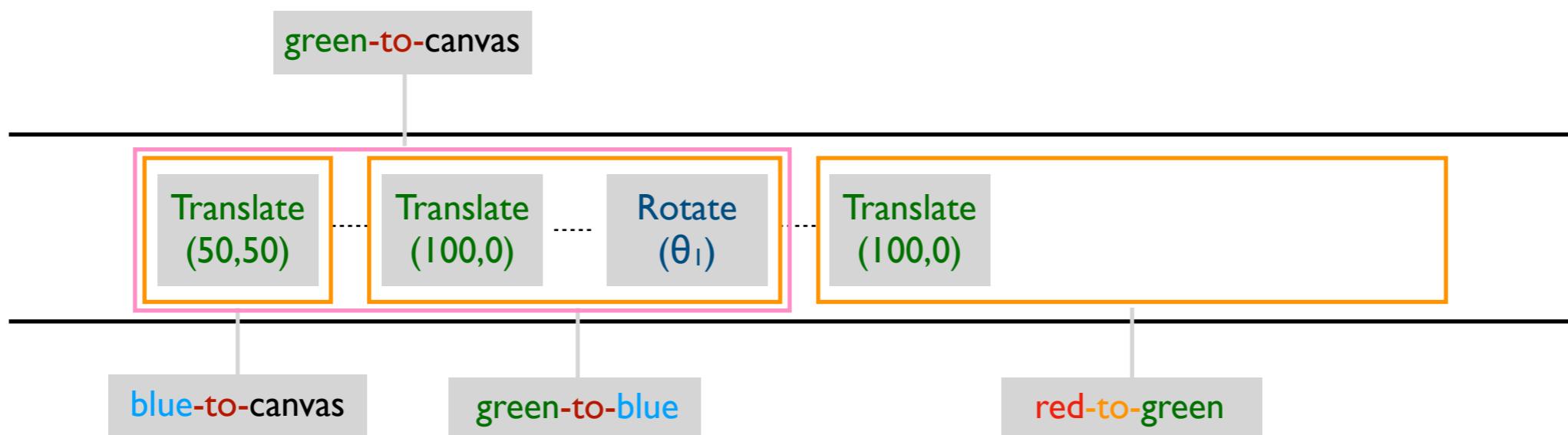
mat3.fromTranslation(Tblue_to_canvas, [50, 150]);
linkage("blue", Tblue_to_canvas);

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);

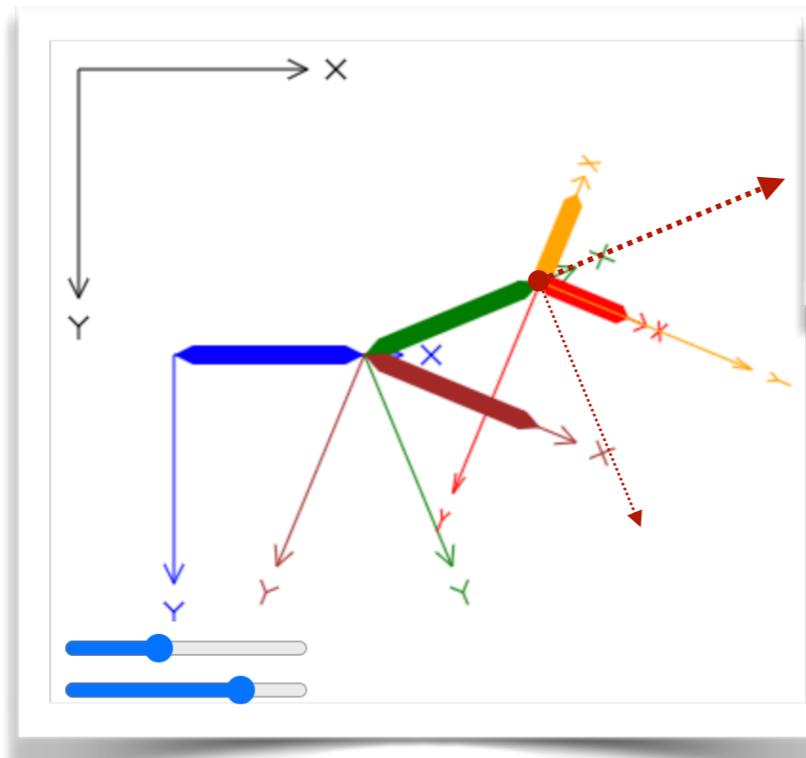
var Tred_to_green = mat3.create();
mat3.fromTranslation(Tred_to_green, [100, 0]);
mat3.rotate(Tred_to_green, Tred_to_green, phi1);
mat3.scale(Tred_to_green, Tred_to_green, [0.5, 1]);
var Tred_to_canvas = mat3.create();
mat3.multiply(Tred_to_canvas, Tgreen_to_canvas, Tred_to_green);
linkage("red", Tred_to_canvas);

[...]

```



Transforms in algebraic form (via *glmatrix*)



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

```

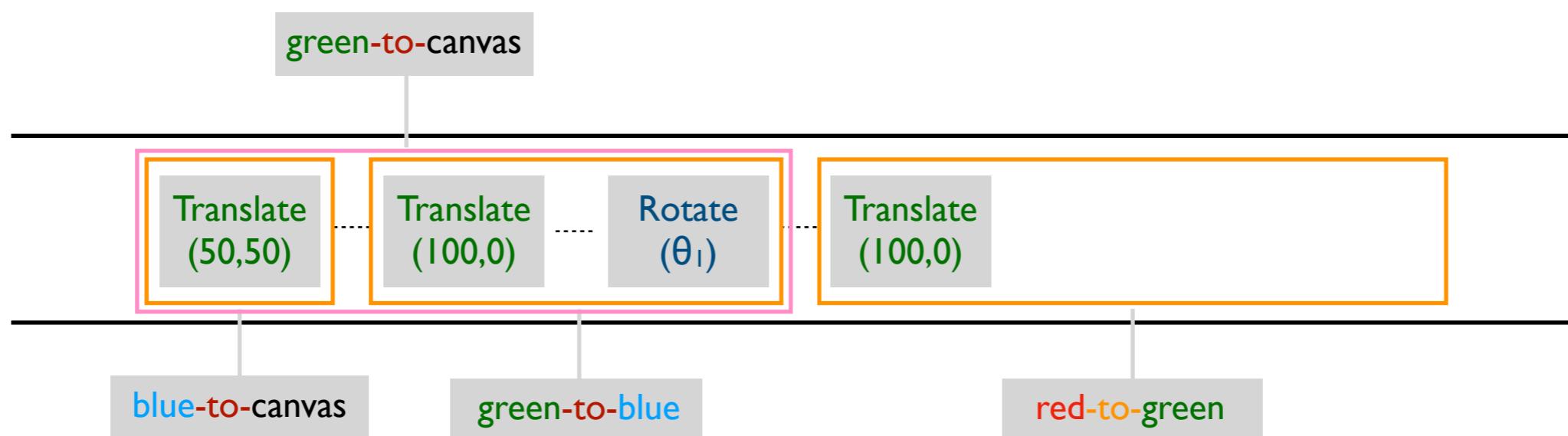
mat3.fromTranslation(Tblue_to_canvas,[50,150]);
linkage("blue",Tblue_to_canvas);

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);

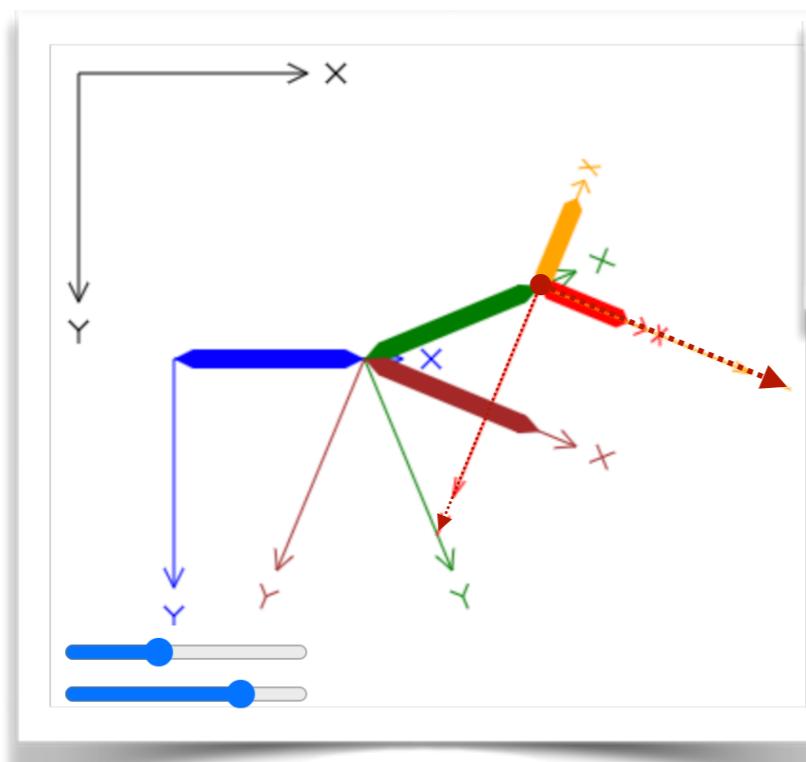
var Tred_to_green = mat3.create();
mat3.fromTranslation(Tred_to_green,[100,0]);
mat3.rotate(Tred_to_green,Tred_to_green,phi1);
mat3.scale(Tred_to_green,Tred_to_green,[0.5,1]);
var Tred_to_canvas = mat3.create();
mat3.multiply(Tred_to_canvas,Tgreen_to_canvas,Tred_to_green);
linkage("red",Tred_to_canvas);

[...]

```



Transforms in algebraic form (via *glmatrix*)



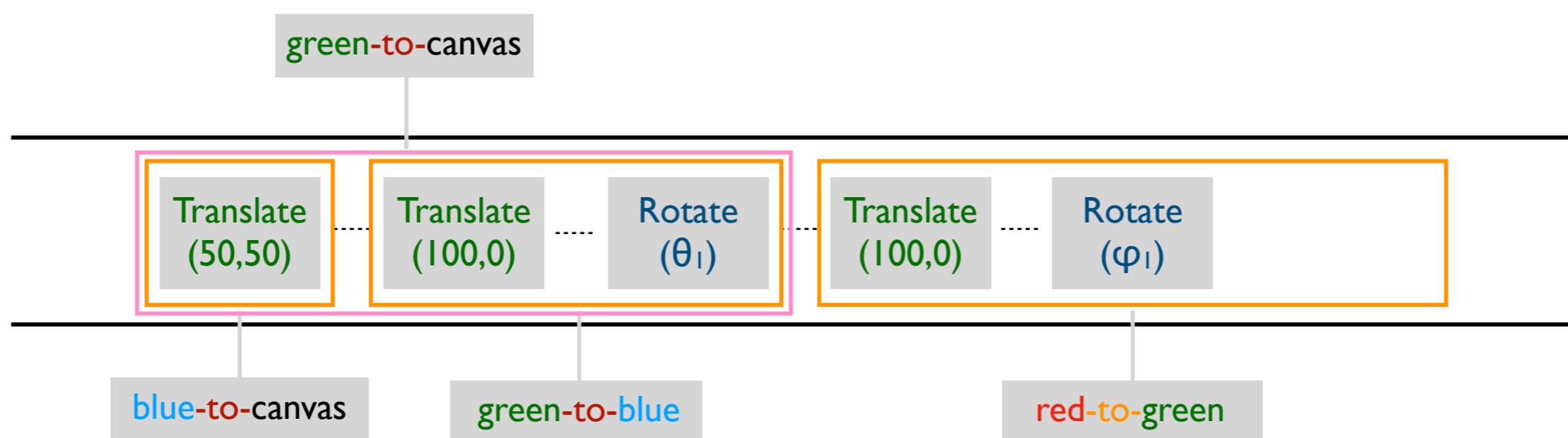
$$\mathbf{T}_{\text{red2green}} = \begin{pmatrix} 1 & 0 & 100 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \phi_1 & -\sin \phi_1 & 0 \\ \sin \phi_1 & \cos \phi_1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

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

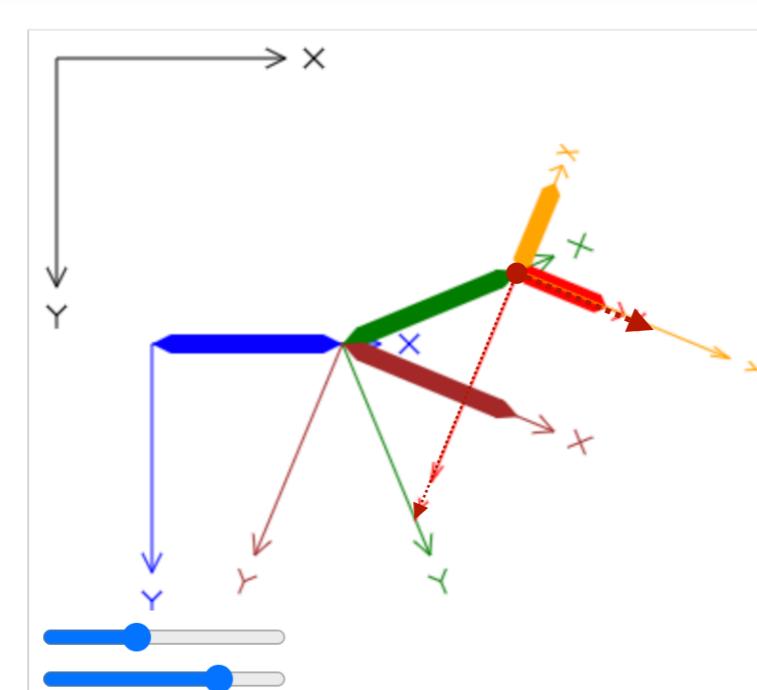
var Tgreen_to_blue = mat3.create();
mat3.fromTranslation(Tgreen_to_blue,[100,0]);
mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
var Tgreen_to_canvas = mat3.create();
mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
linkage("green",Tgreen_to_canvas);

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*)



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

```

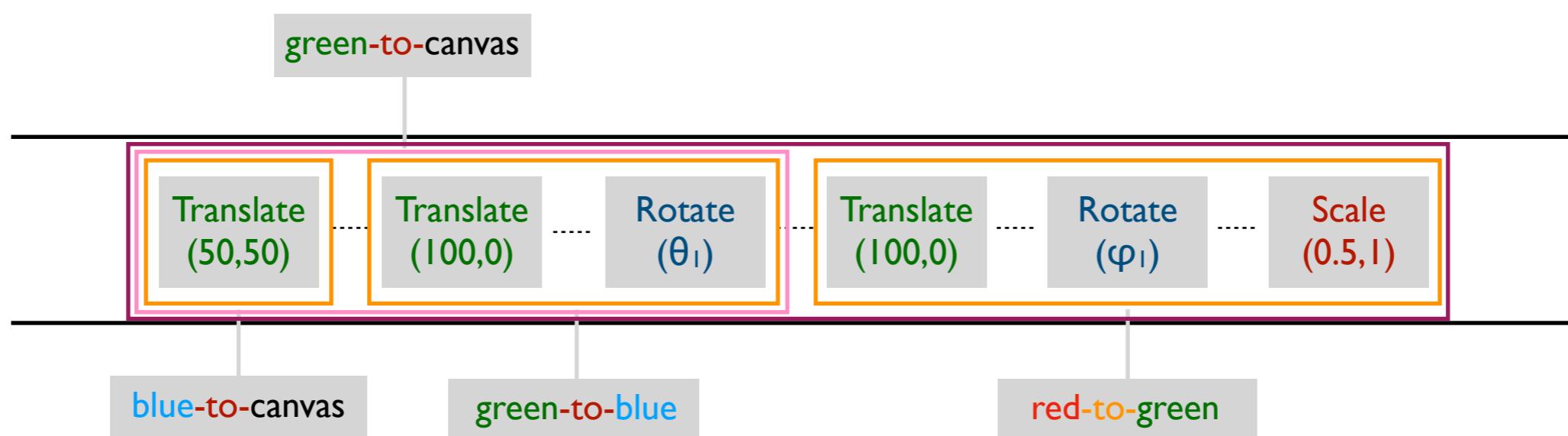
mat3.fromTranslation(Tblue_to_canvas, [50, 150]);
linkage("blue", Tblue_to_canvas);

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);

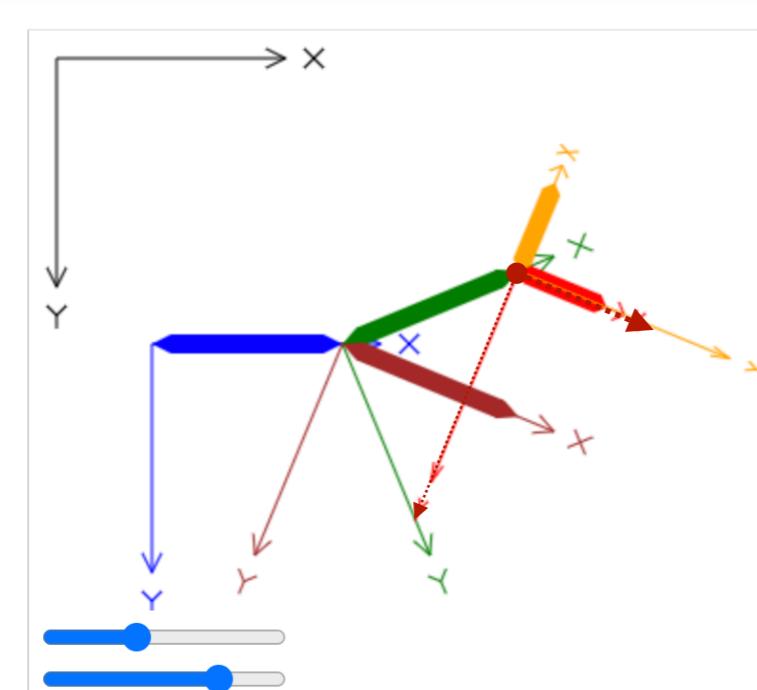
var Tred_to_green = mat3.create();
mat3.fromTranslation(Tred_to_green, [100, 0]);
mat3.rotate(Tred_to_green, Tred_to_green, phi1);
mat3.scale(Tred_to_green, Tred_to_green, [0.5, 1]);
var Tred_to_canvas = mat3.create();
mat3.multiply(Tred_to_canvas, Tgreen_to_canvas, Tred_to_green);
linkage("red", Tred_to_canvas);

[...]

```



Transforms in algebraic form (via *glmatrix*)



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

```

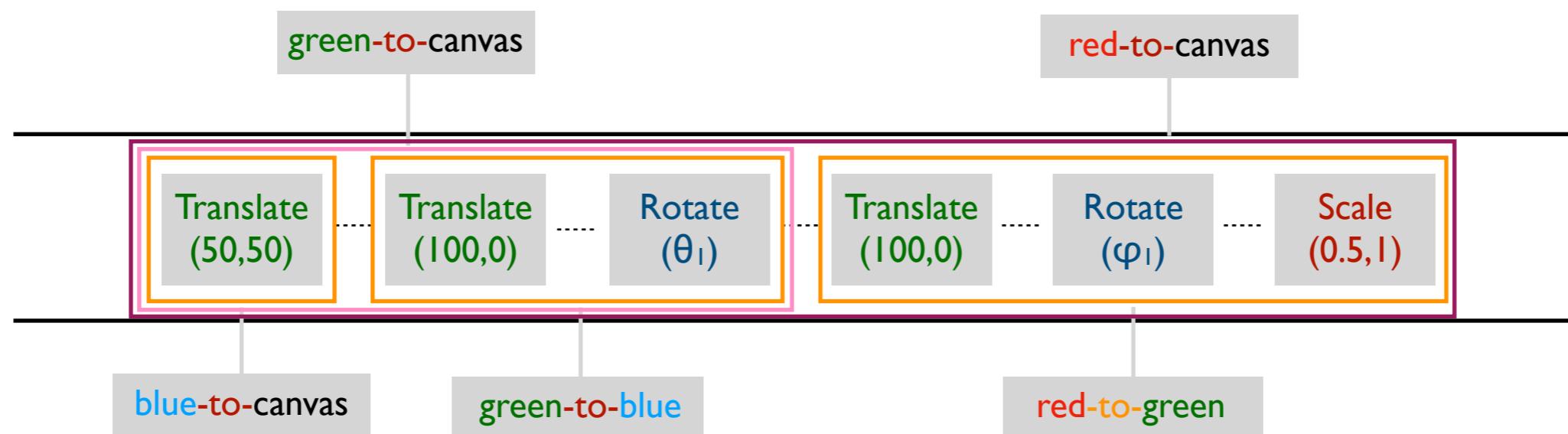
mat3.fromTranslation(Tblue_to_canvas, [50, 150]);
linkage("blue", Tblue_to_canvas);

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);

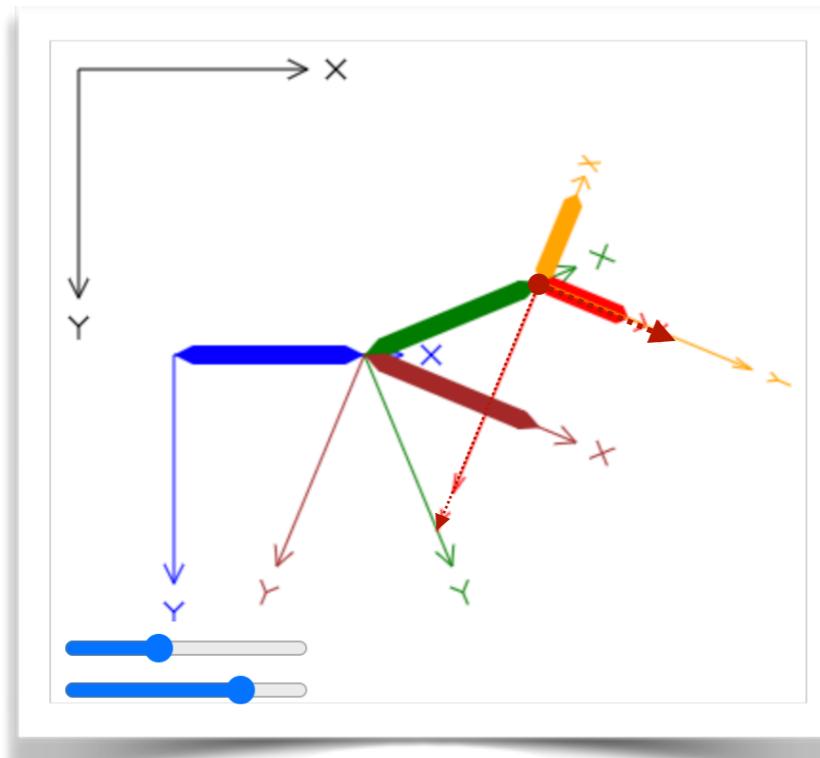
var Tred_to_green = mat3.create();
mat3.fromTranslation(Tred_to_green, [100, 0]);
mat3.rotate(Tred_to_green, Tred_to_green, phi1);
mat3.scale(Tred_to_green, Tred_to_green, [0.5, 1]);
var Tred_to_canvas = mat3.create();
mat3.multiply(Tred_to_canvas, Tgreen_to_canvas, Tred_to_green);
linkage("red", Tred_to_canvas);

[...]

```



Transforms in algebraic form (via *glmatrix*)



demo.js

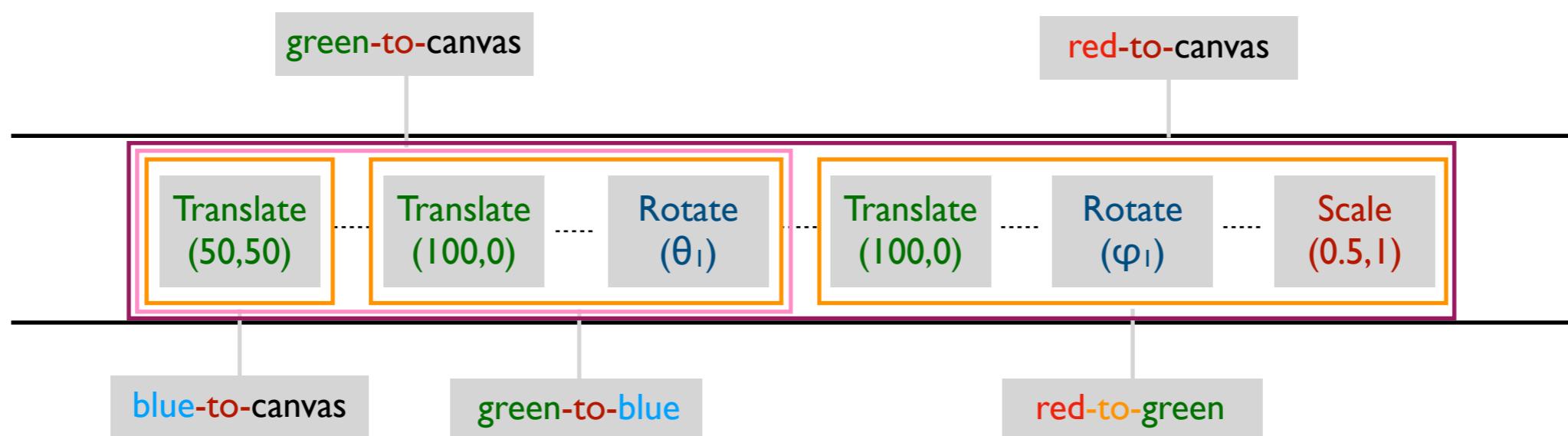
[...]

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

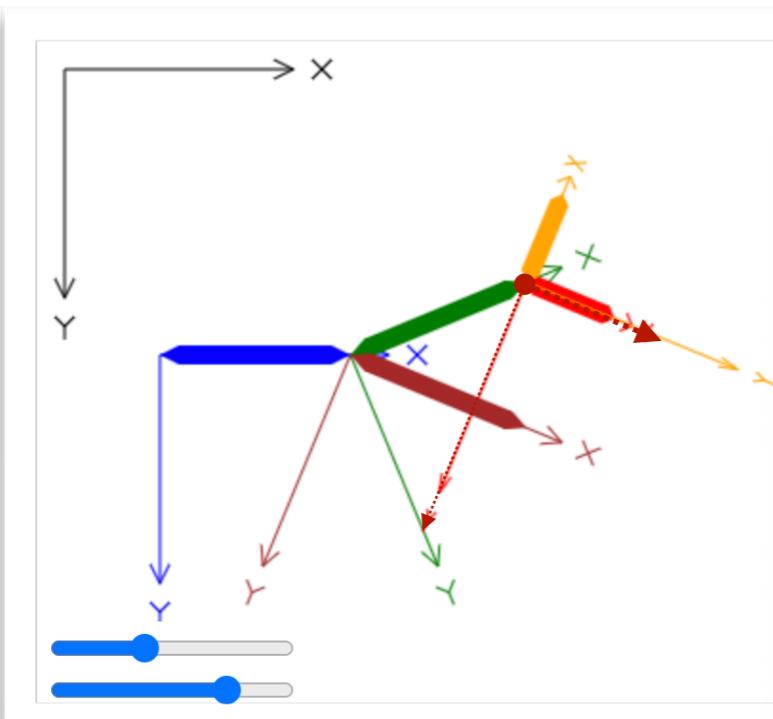
var Tgreen_to_blue = mat3.create();
mat3.fromTranslation(Tgreen_to_blue, [100, 0]);
mat3.rotate(Tgreen_to_blue, Tgreen_to_blue, theta1);
var Tgreen_to_canvas = mat3.create();
mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
linkage("green", Tgreen_to_canvas);
```

```
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)



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);

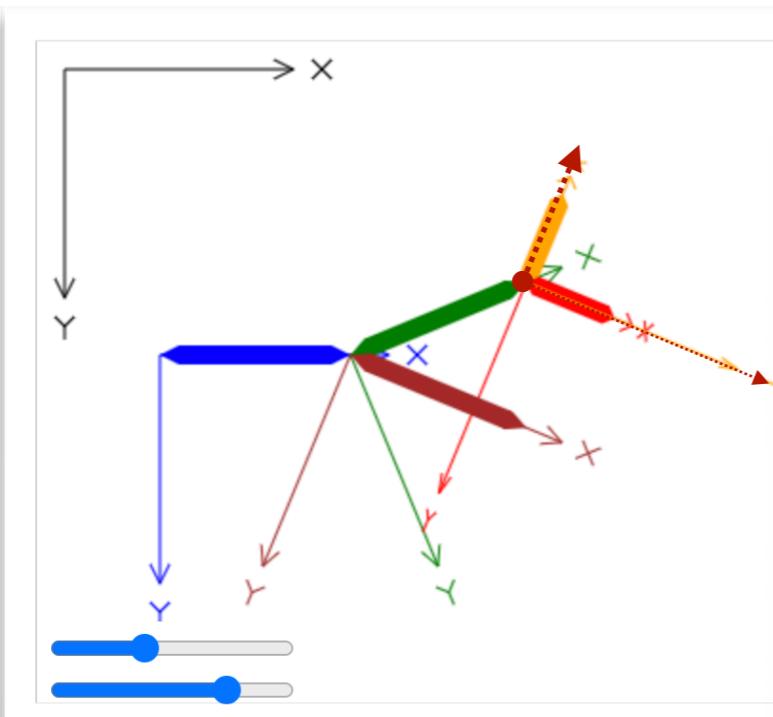
var Tred_to_green = mat3.create();
mat3.fromTranslation(Tred_to_green, [100,0]);
mat3.rotate(Tred_to_green,Tred_to_green,phi1);
mat3.scale(Tred_to_green,Tred_to_green,[0.5,1]);
var Tred_to_canvas = mat3.create();
mat3.multiply(Tred_to_canvas,Tgreen_to_canvas,Tred_to_green);
linkage("red",Tred_to_canvas);

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)



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);

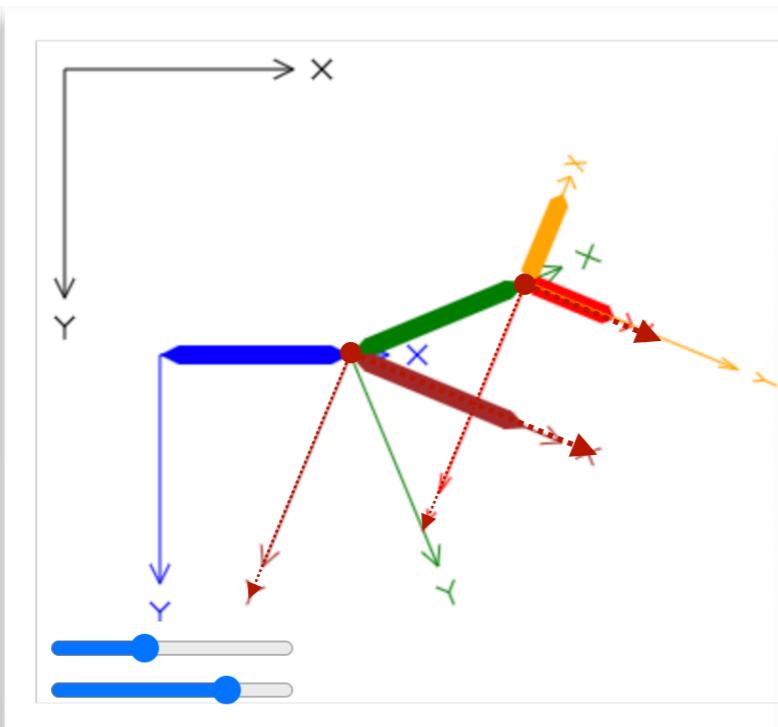
var Torange_to_green = mat3.create();
mat3.fromTranslation(Torange_to_green, [100,0]);
mat3.rotate(Torange_to_green,Torange_to_green,phi2);
mat3.scale(Torange_to_green,Torange_to_green,[0.5,1]);
var Torange_to_canvas = mat3.create();
mat3.multiply(Torange_to_canvas,Tgreen_to_canvas,Torange_to_green);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



demo.js

[...]

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

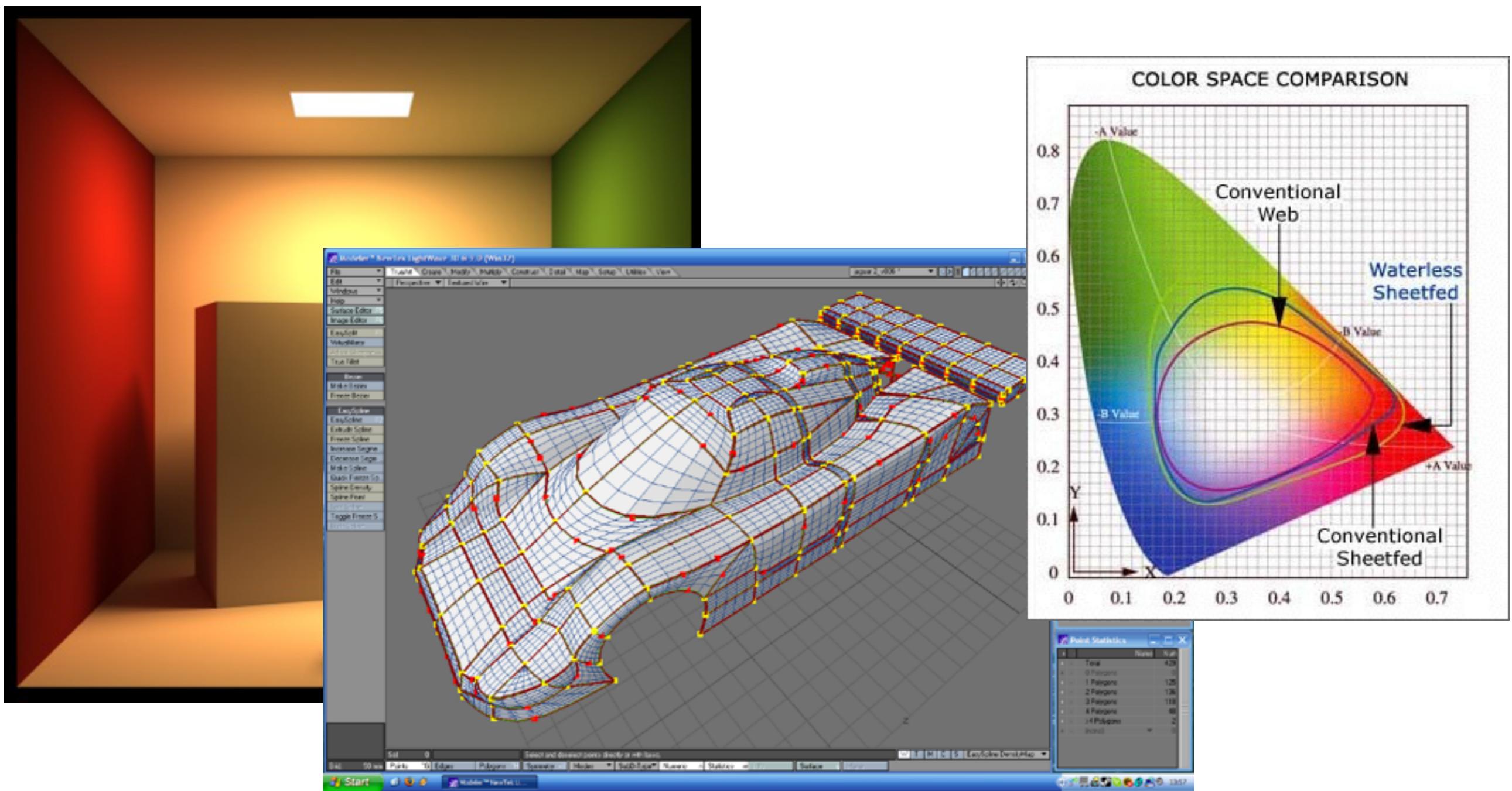
var Tgreen_to_blue = mat3.create();
mat3.fromTranslation(Tgreen_to_blue, [100,0]);
mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,theta1);
var Tgreen_to_canvas = mat3.create();
mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
linkage("green",Tgreen_to_canvas);

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 28th 2021