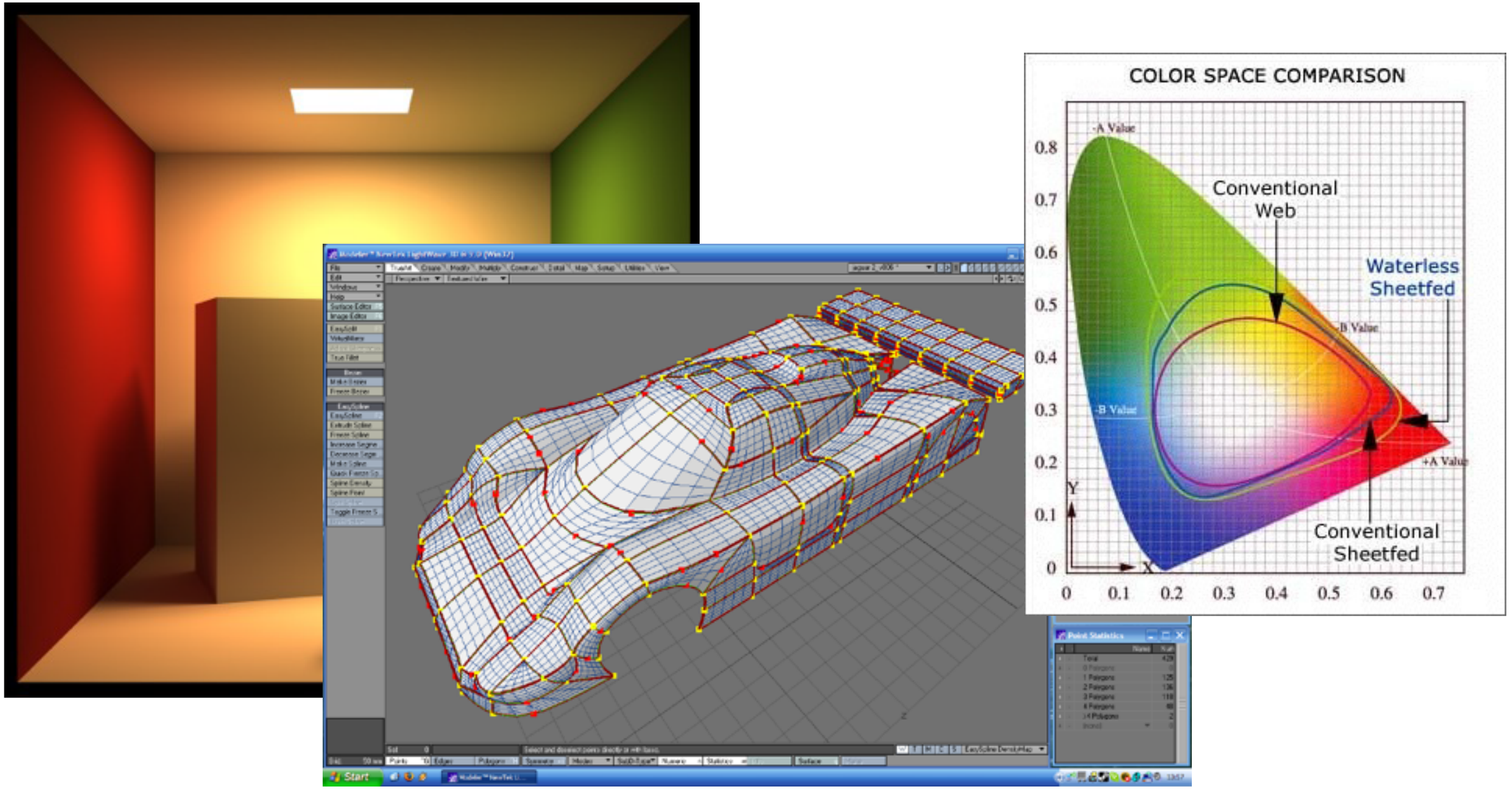


Lecture 7 : Practical use of 2D transforms in code (in algebraic form, via glMatrix, contrasting to Canvas)

Thursday September 30th 2021

Administrative stuff

- Considering change to midterm time/format due to unforeseen personal circumstances
  (Possibility to shift from Friday 10/29 to in-class on Tuesday 11/2; see poll on Canvas)
- Will make necessary accommodations
- Remember that your homework #2 is due next Tuesday, October 7th
- We're working on grading #1, hope to have grades posted by the time assignment #2 is due.

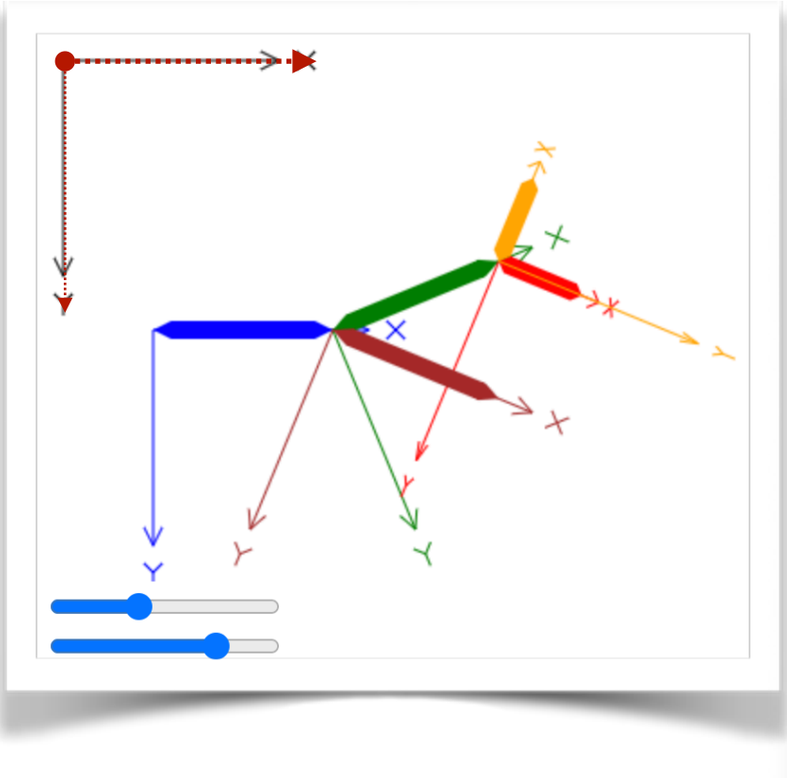
Today's lecture

- Another walkthrough of hierarchical modeling and transforms via glMatrix-assisted matrix algebra
- Comparison between explicit transform formation and canvas transform operations
- A couple paradigms for setting the canvas transform, or performing the "master" transform manually
- An example of a user-maintained stack
- A quick look at TWGL (just to see how a different library might be structured)

Transforms in algebraic form (via *glmatrix*)

jsbin.com/yifahoq

Week4/Demo0



demo.js

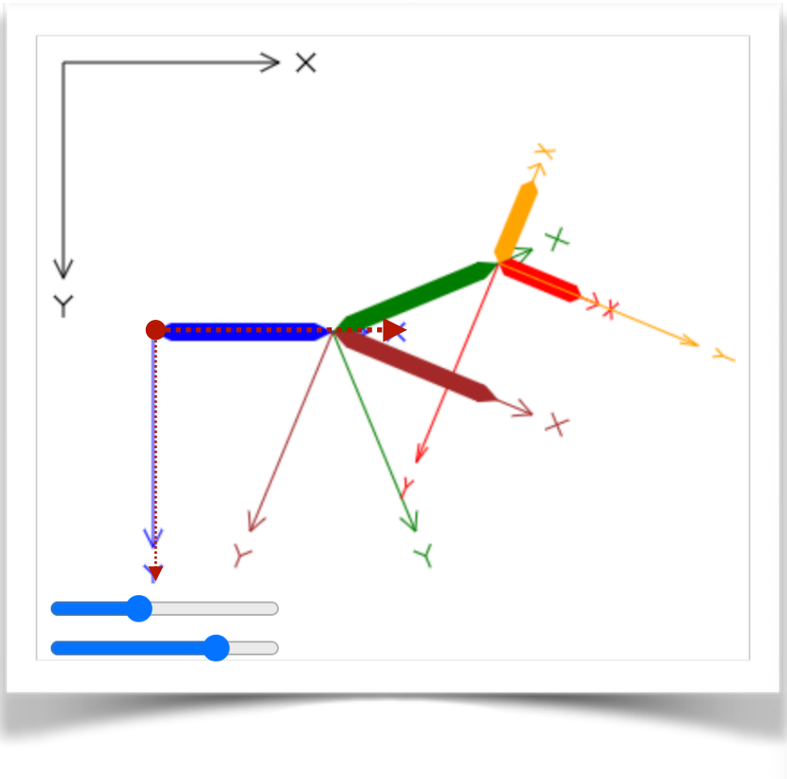
```
[...]

    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);

[...]
```

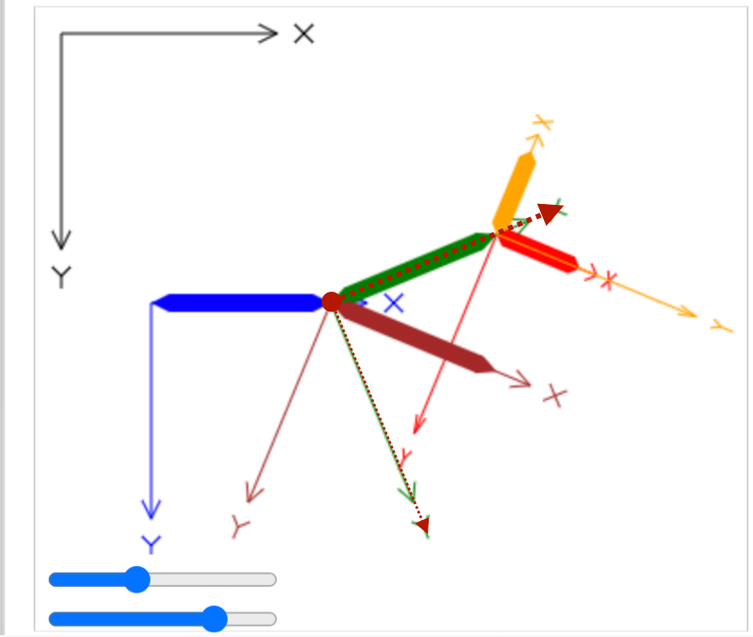
Translate
(50,150)

blue-to-canvas

Transforms in algebraic form (via *glmatrix*)

jsbin.com/yifahoq

Week4/Demo0

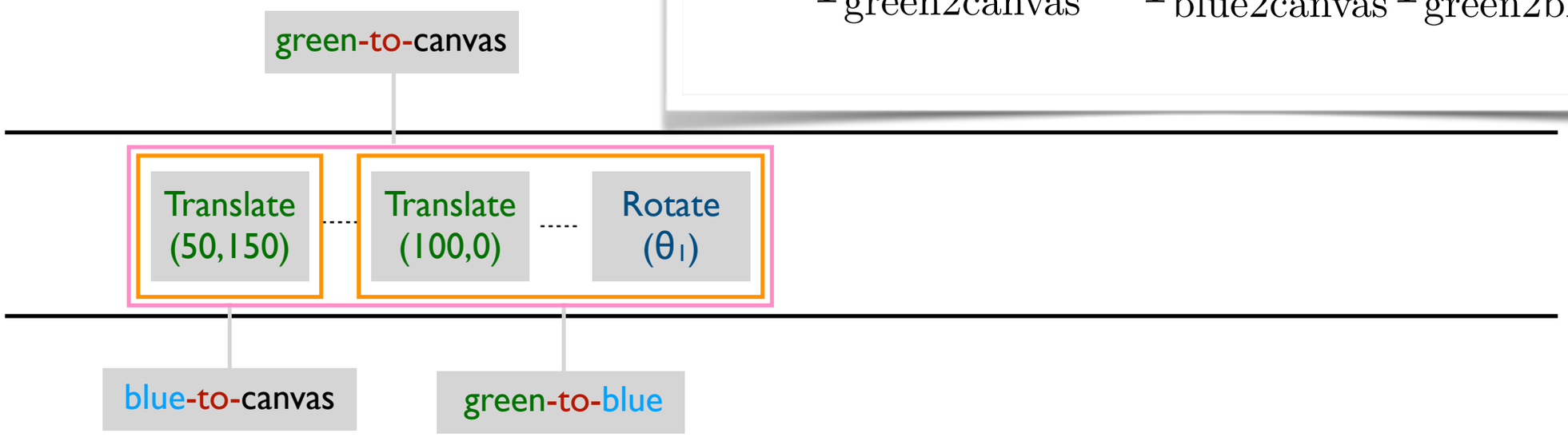


$$\mathbf{T}_{\text{green2blue}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 100 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

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);

een = mat3.create();
lation(Tred_to_green,[100,0]);
ed_to_green,Tred_to_green,phi1);
d_to_green,Tred_to_green,[0.5,1]);
nvas = mat3.create();
Tred_to_canvas,Tgreen_to_canvas,Tred_to_green);
Tred_to_canvas);
```

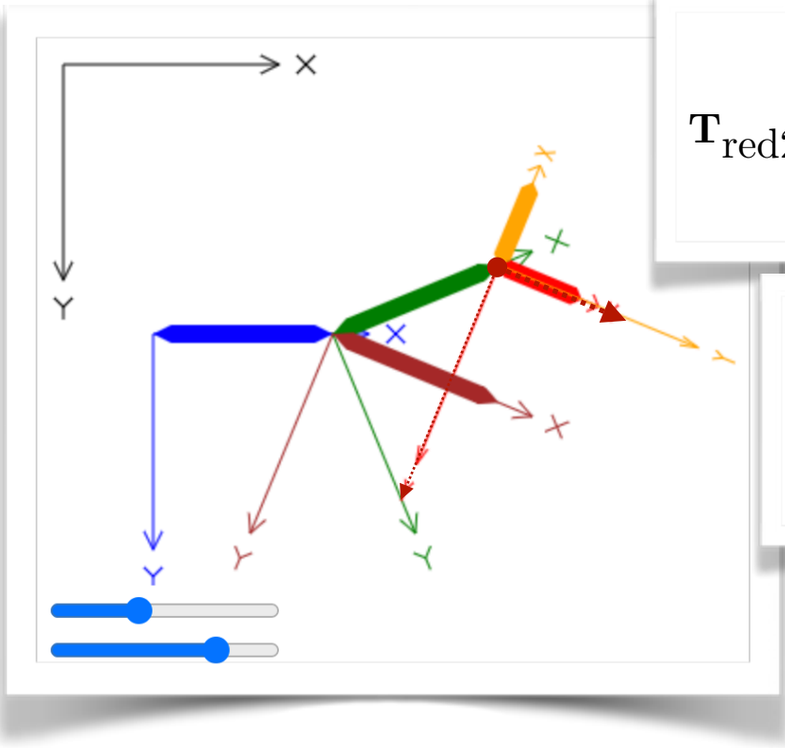
$$\mathbf{T}_{\text{green2canvas}} = \mathbf{T}_{\text{blue2canvas}}\mathbf{T}_{\text{green2blue}}$$



Transforms in algebraic form (via *glmatrix*)

jsbin.com/yifahoq

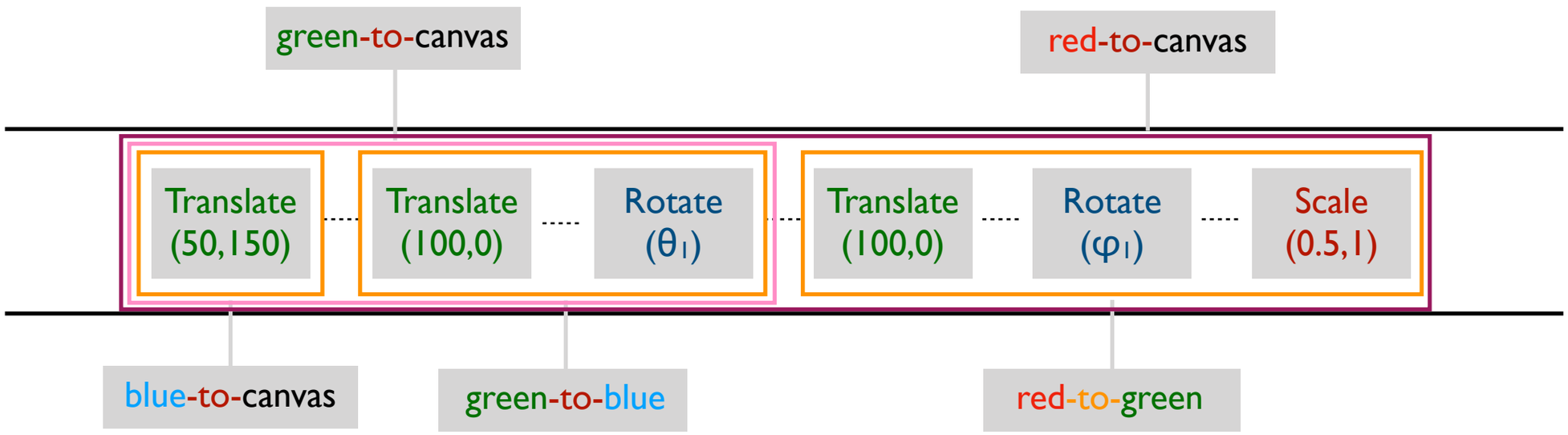
Week4/Demo0



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

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

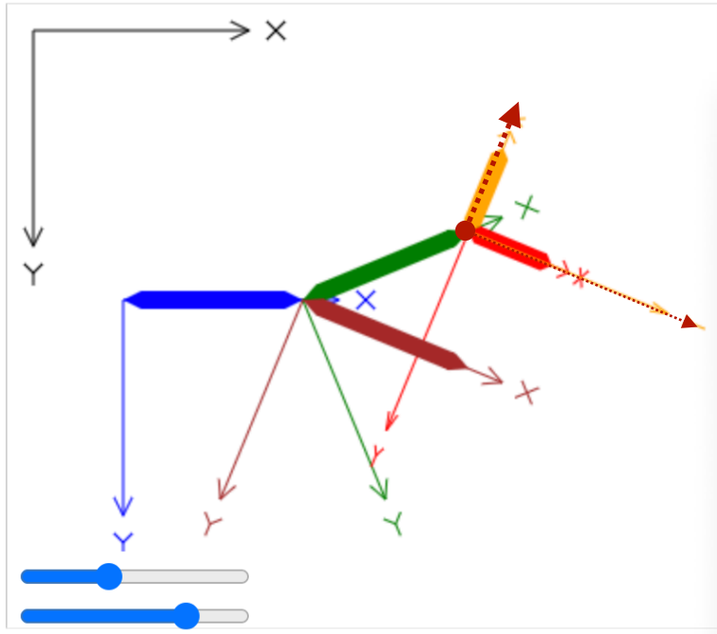
```
mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
linkage("green",Tgreen_to_canvas);

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



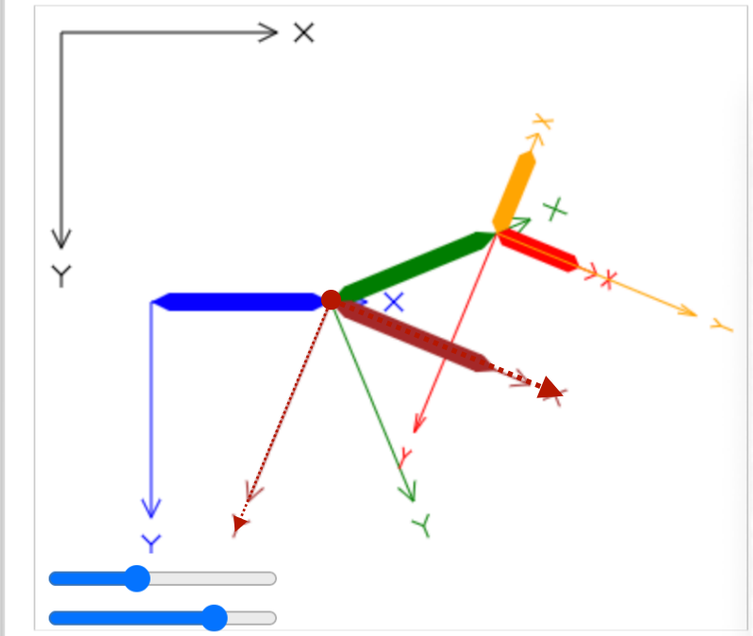
demo.js

```
[...]

    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);

[...]
```

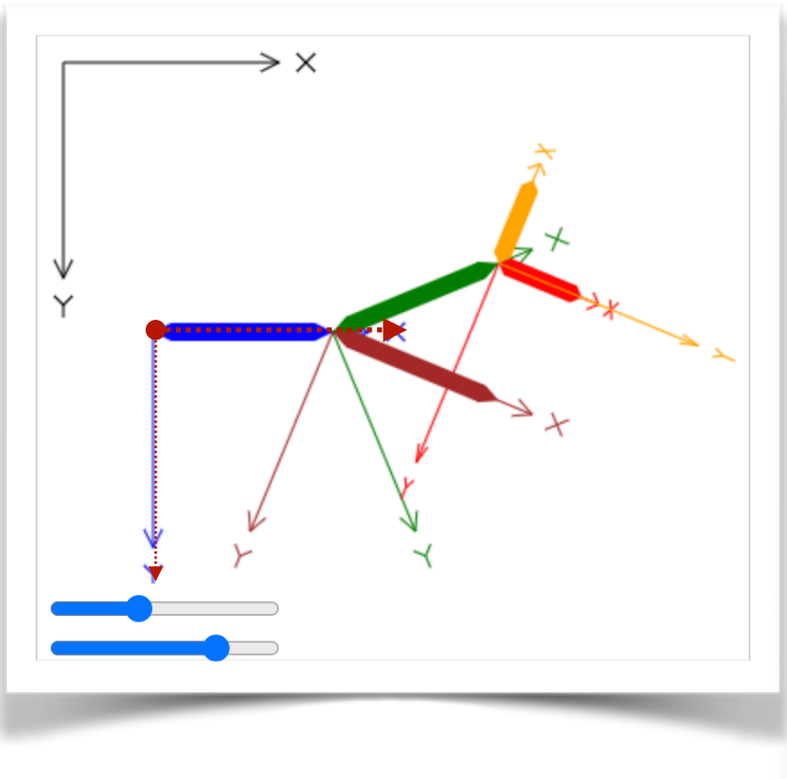
Canvas vs. Glmatrix

- 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



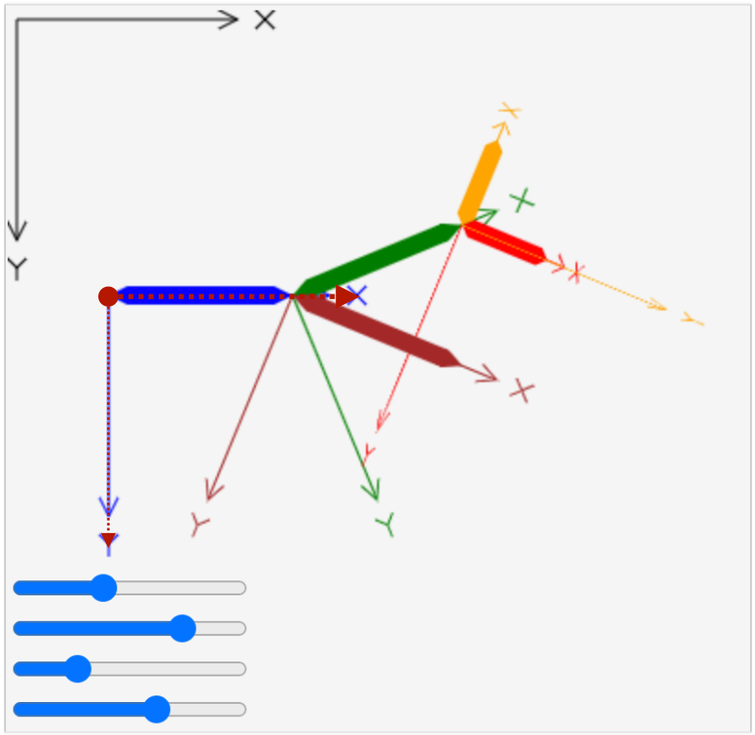
demo.js

```
[...]

   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);

[...]
```

Translate
(50,150)

blue-to-canvas

Canvas transform stack



Canvas transform stack

canvas-to-blue

jsbin.com/wuyutirife

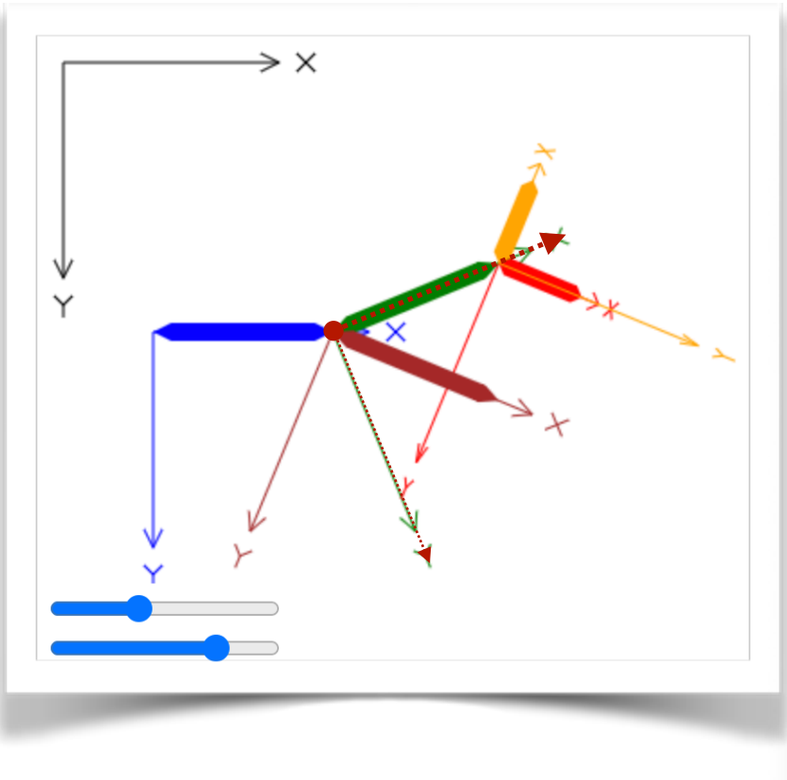
JavaScript

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

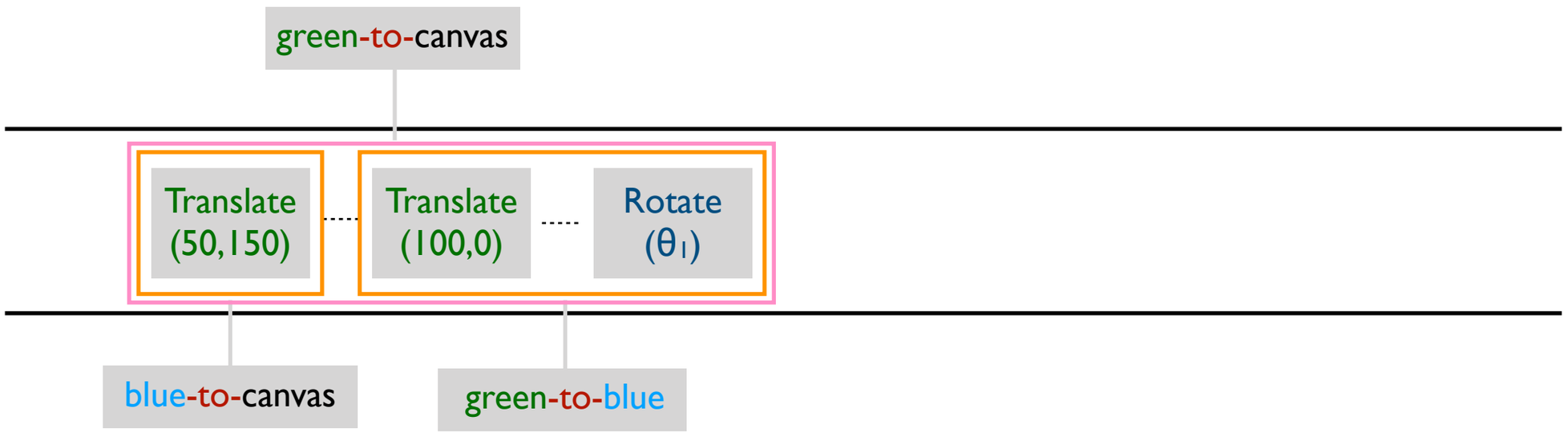
jsbin.com/yifahoq

Week4/Demo0

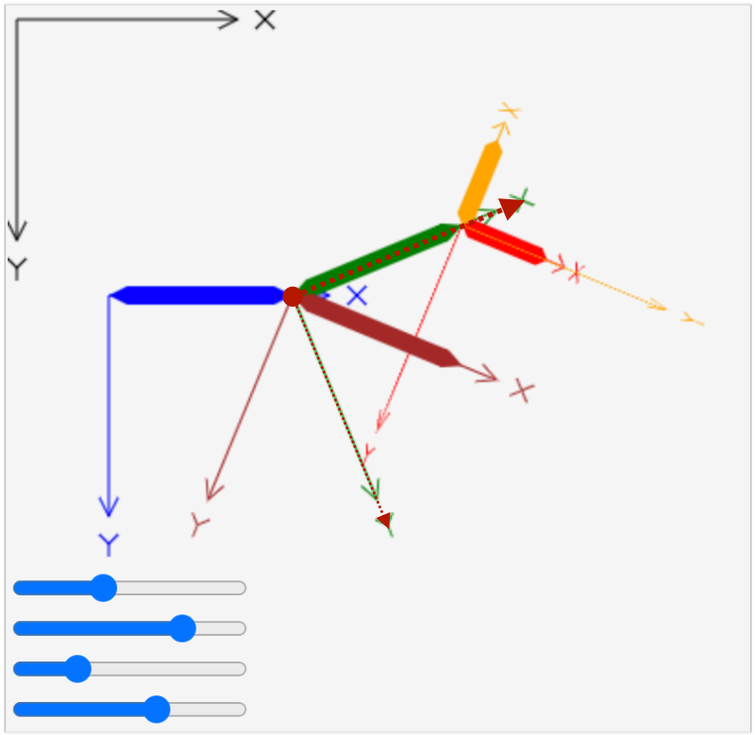


demo.js

```
[...]

    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

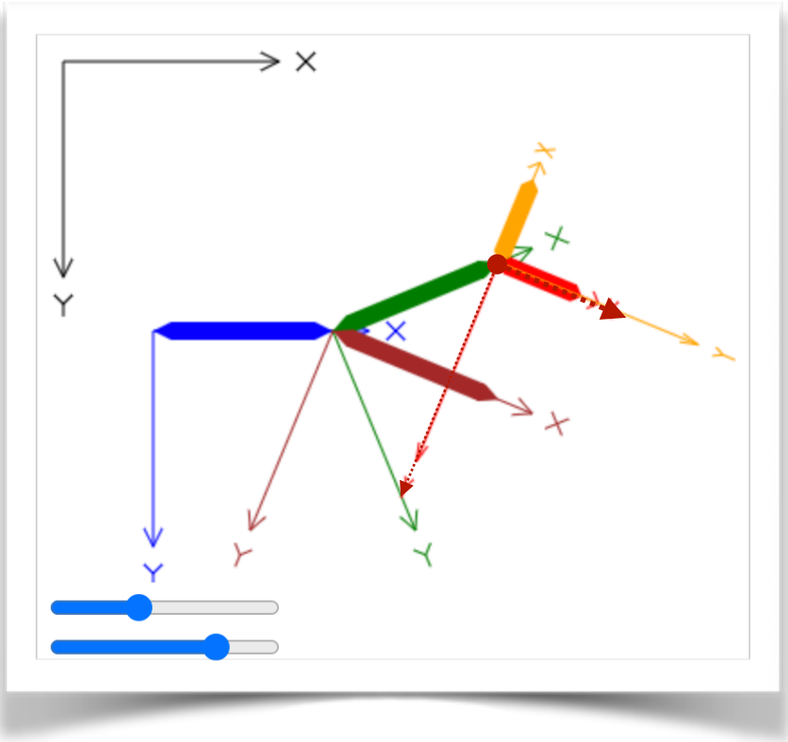
JavaScript

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

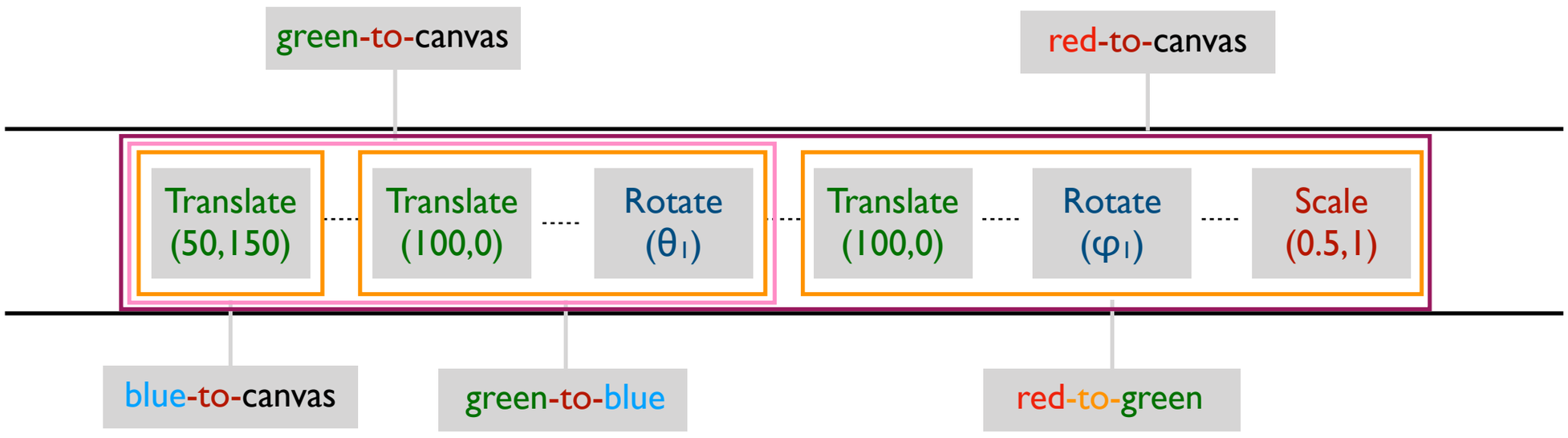
jsbin.com/yifahoq

Week4/Demo0

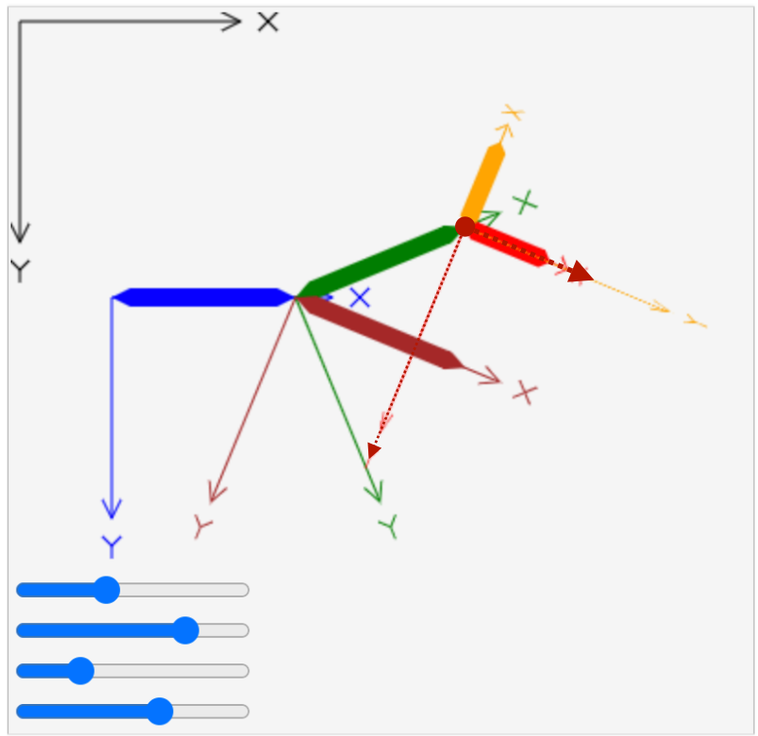


demo.js

```
[...]

    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

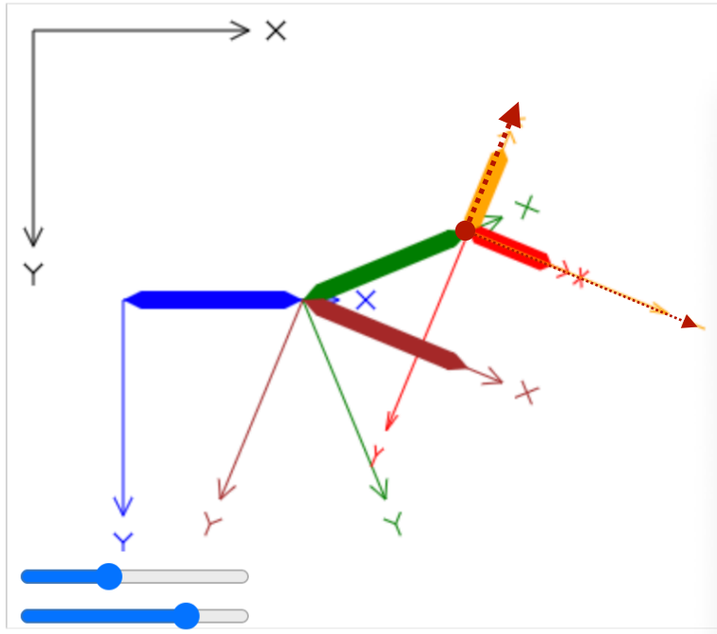
JavaScript

```
[...]
  // 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

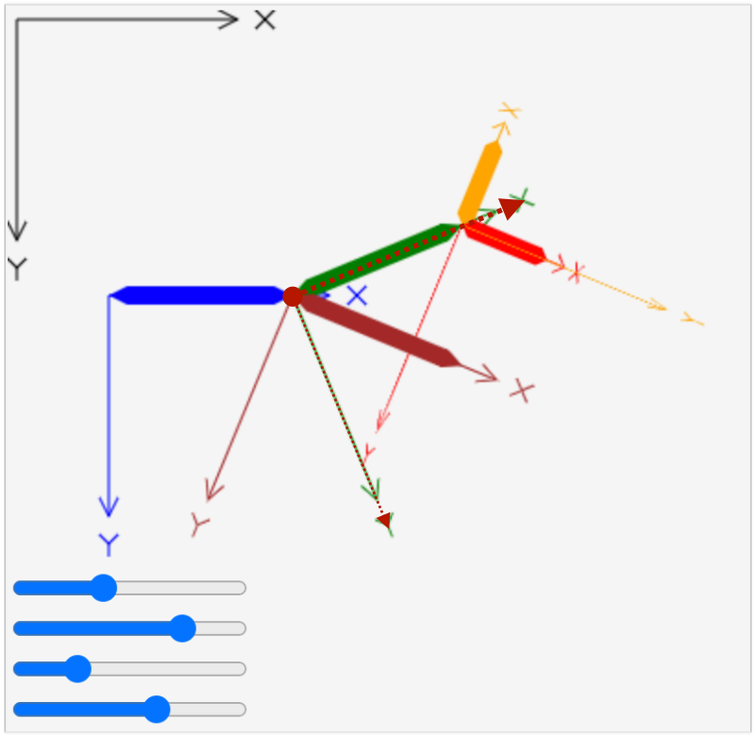
Week4/Demo0



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

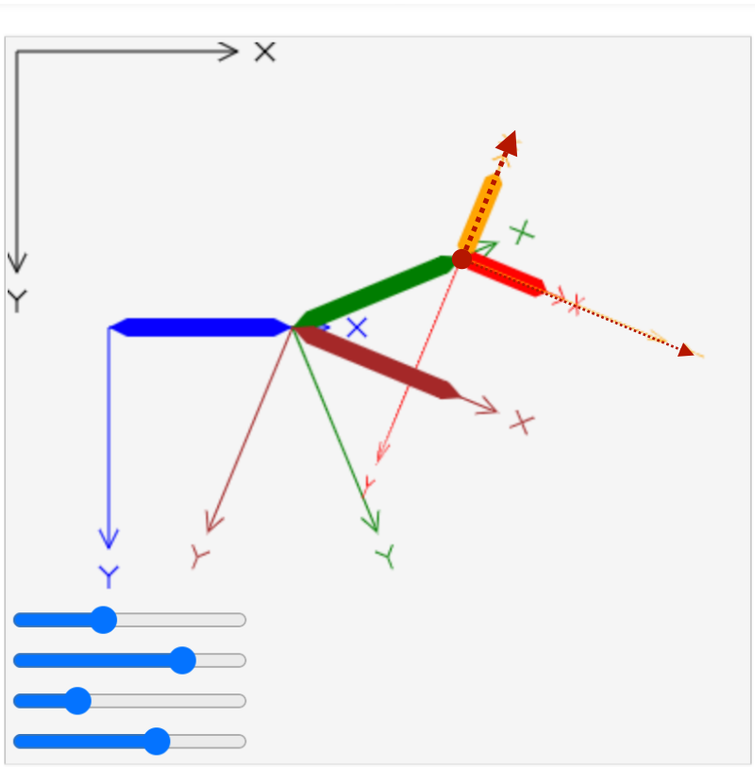
canvas-to-blue

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

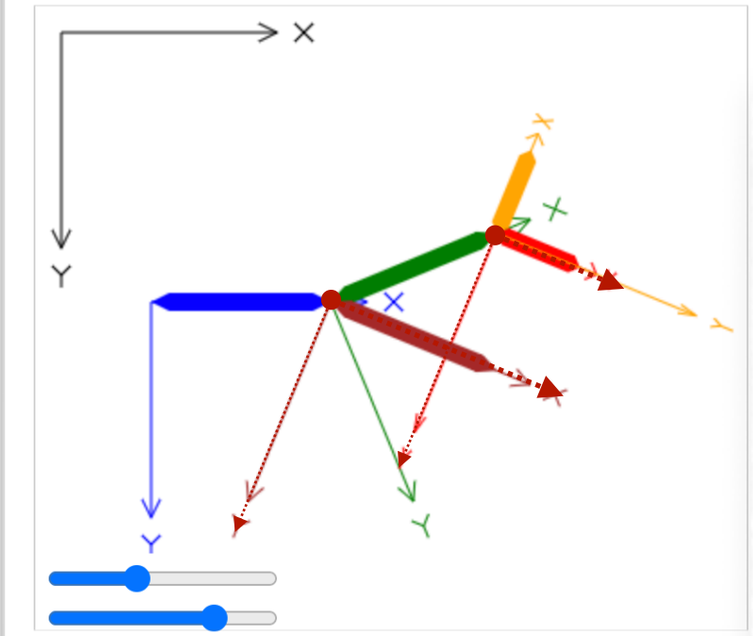
JavaScript

```
[...]
  // 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

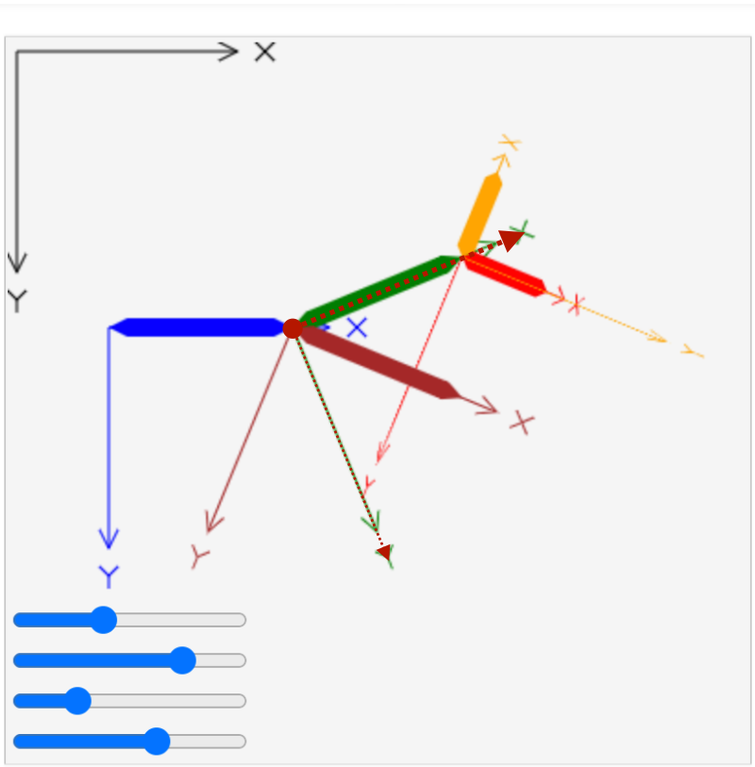
Week4/Demo0



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

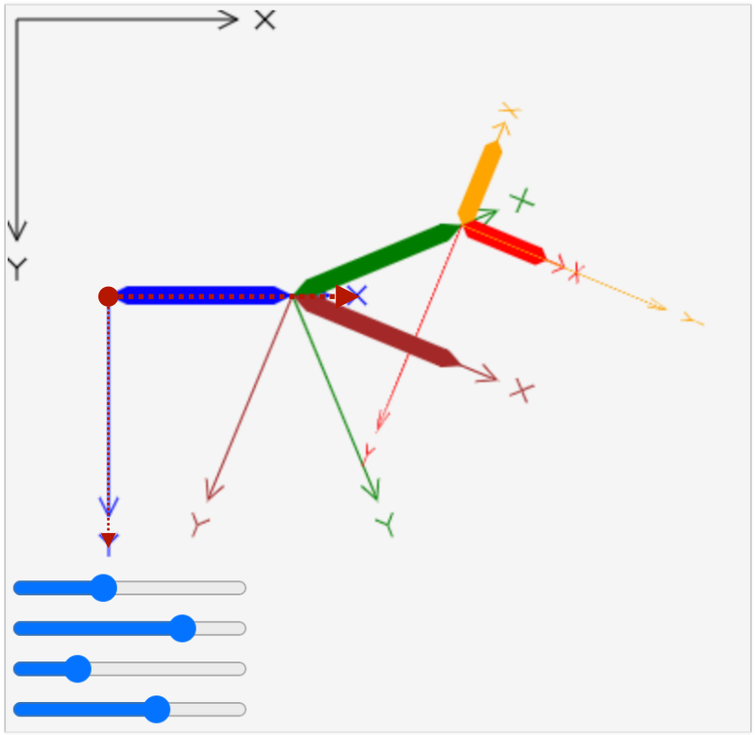
canvas-to-blue

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

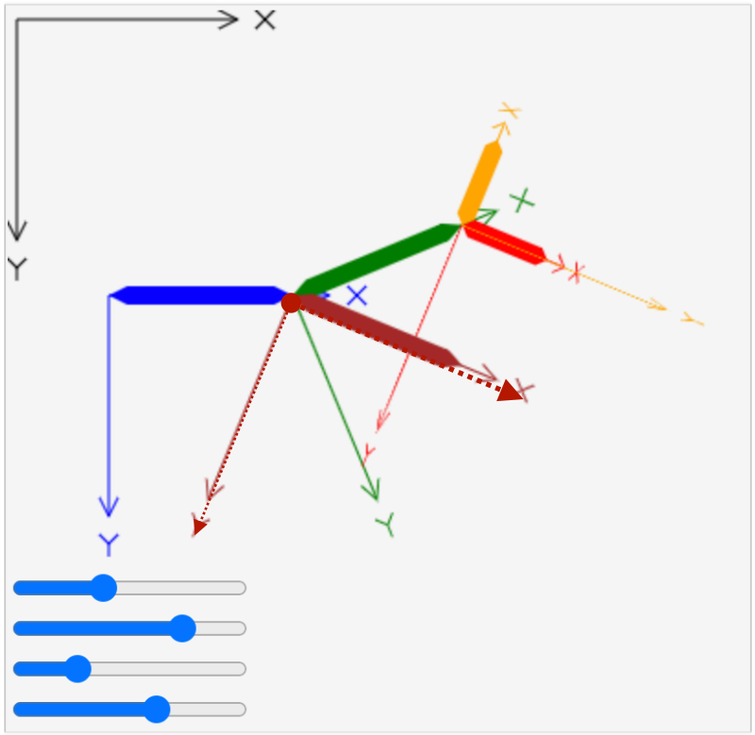
canvas-to-blue

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

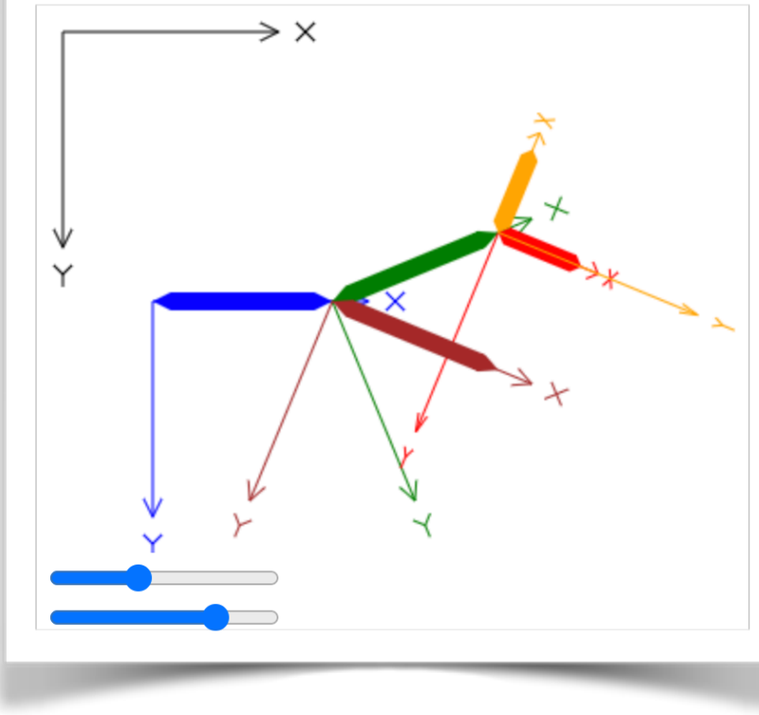
$$\begin{pmatrix} a[0] & a[3] & a[6] \\ a[1] & a[4] & a[7] \\ a[2] & a[5] & a[8] \end{pmatrix}$$

- 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



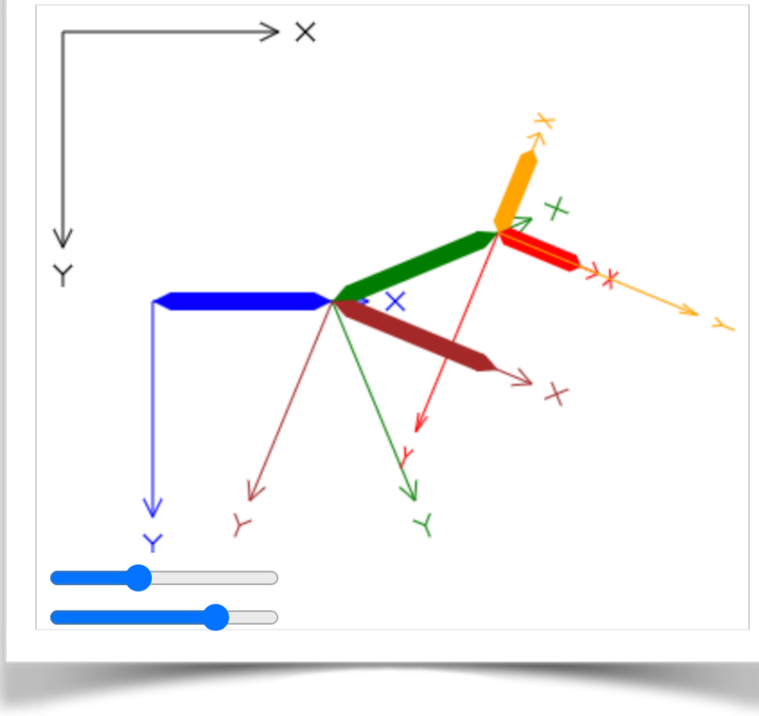
demo.js

```
[...]

    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

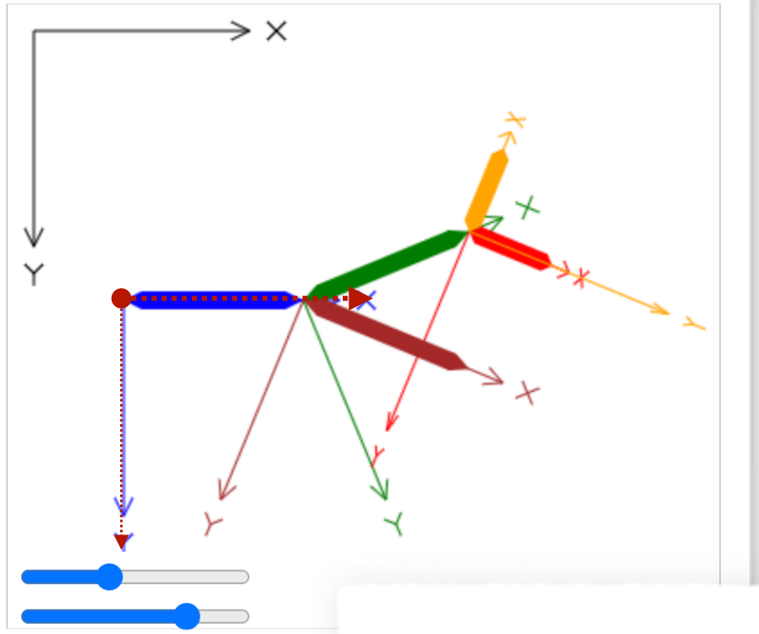
Week4/Demo1

- 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

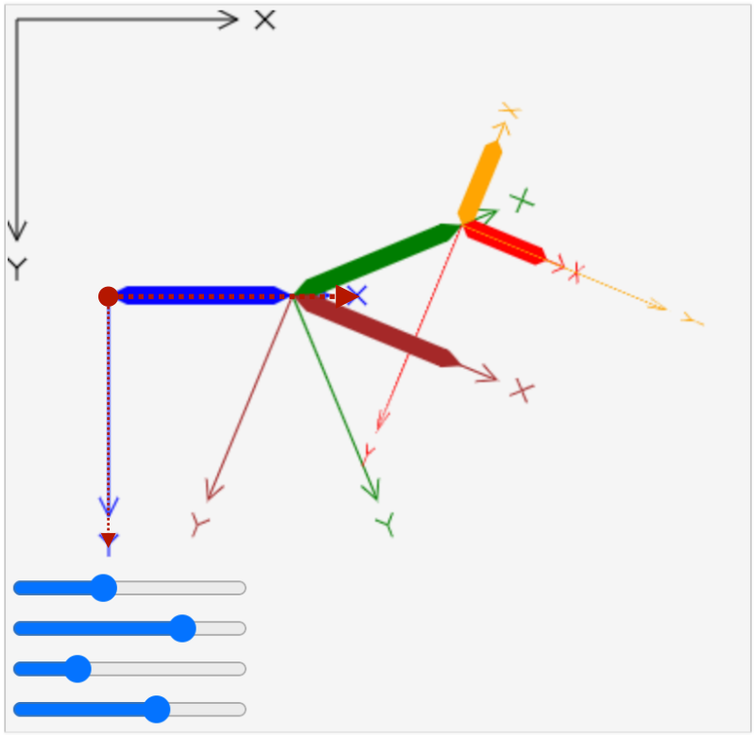
Week4/Demo1



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

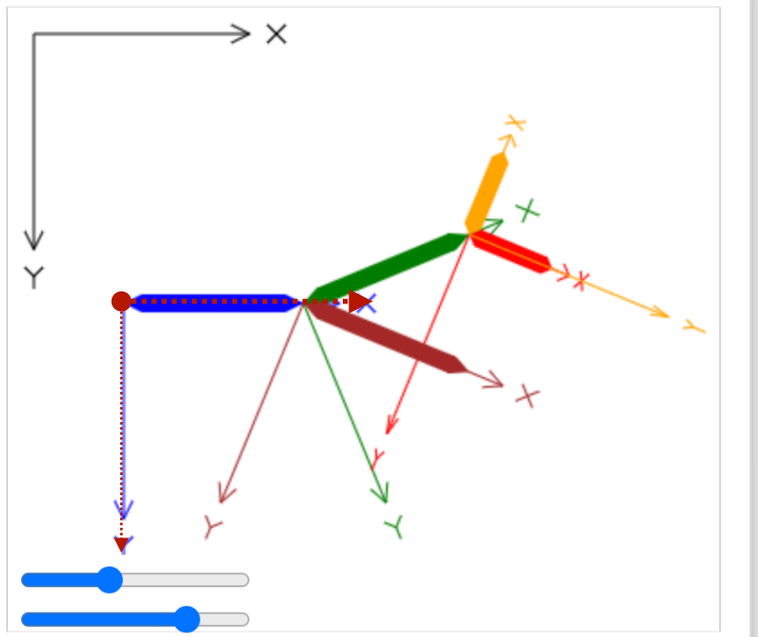
JavaScript

```
[...]
  // 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

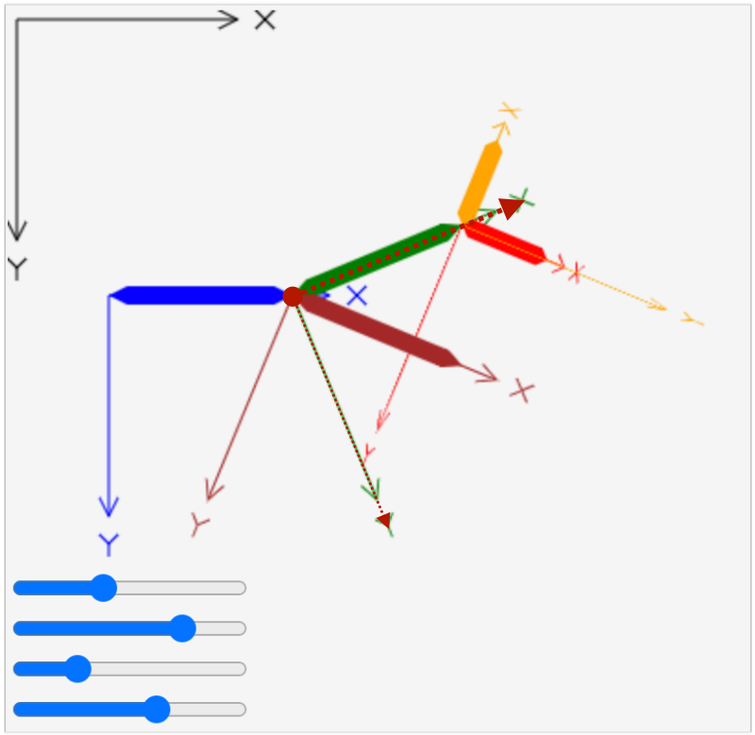
Week4/Demo1



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

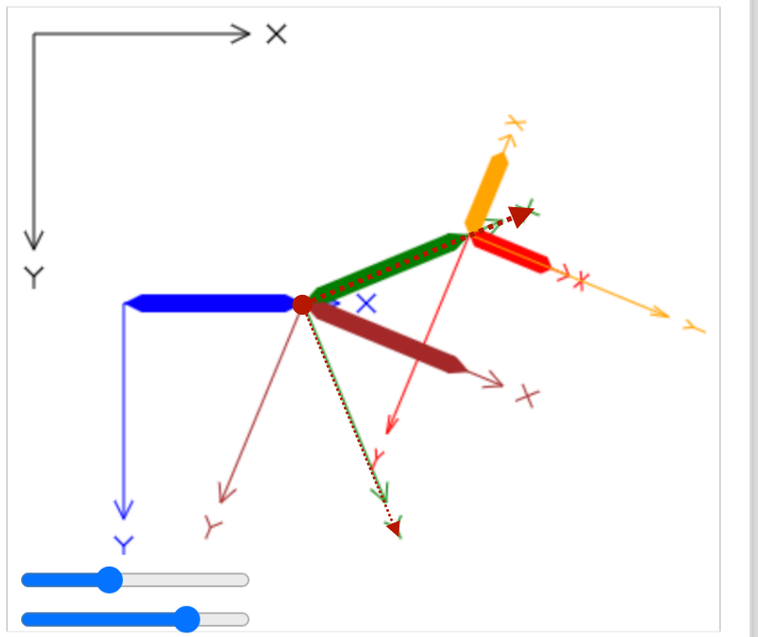
JavaScript

```
[...]
  // 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

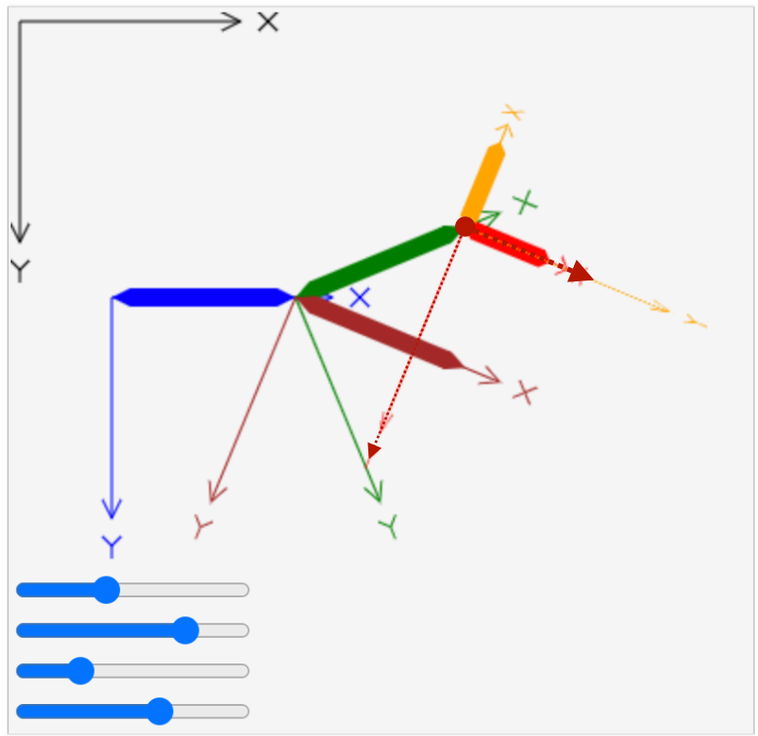
Week4/Demo1



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

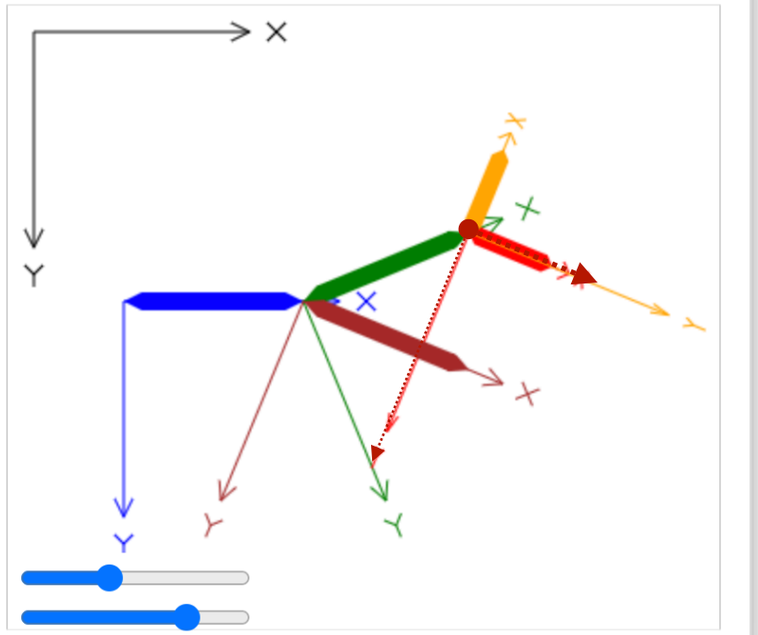
JavaScript

```
[...]
  // 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

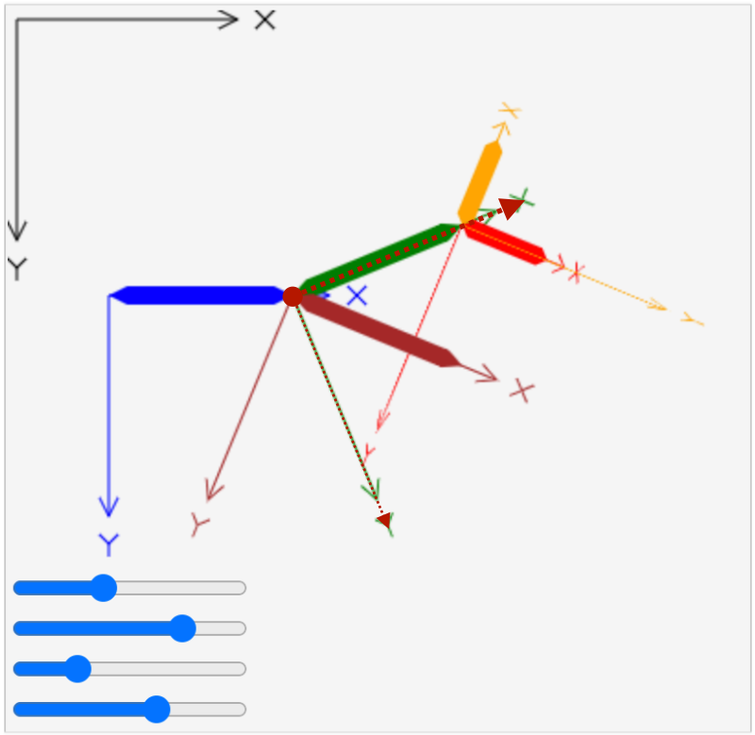
Week4/Demo1



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

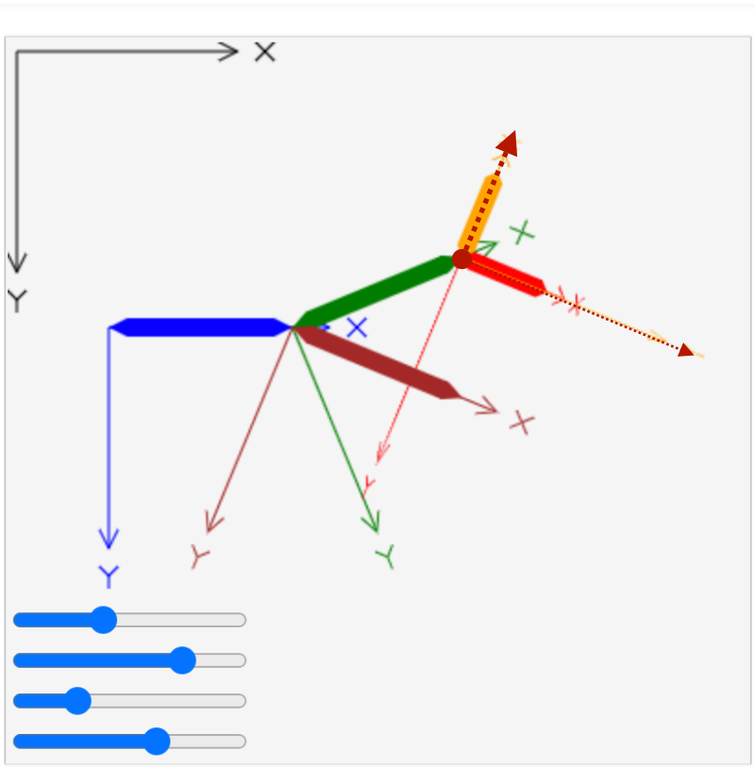
canvas-to-blue

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

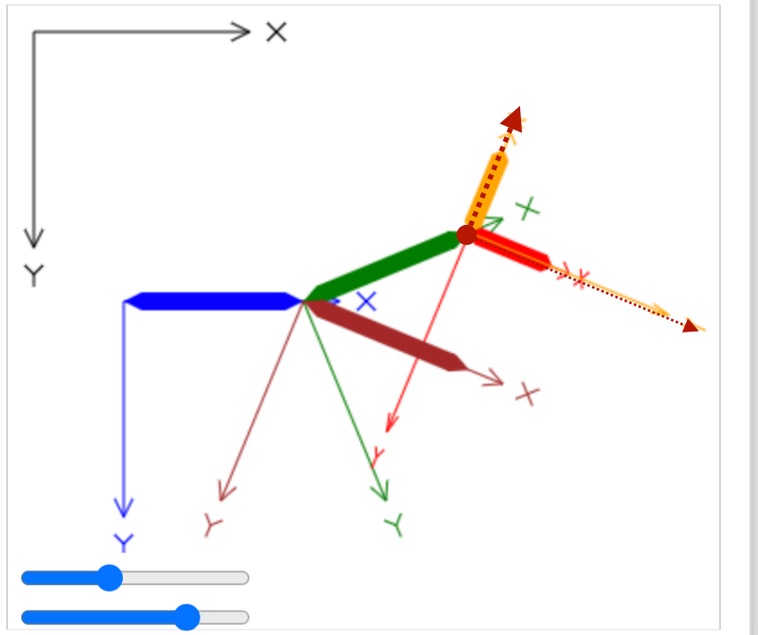
JavaScript

```
[...]
  // 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

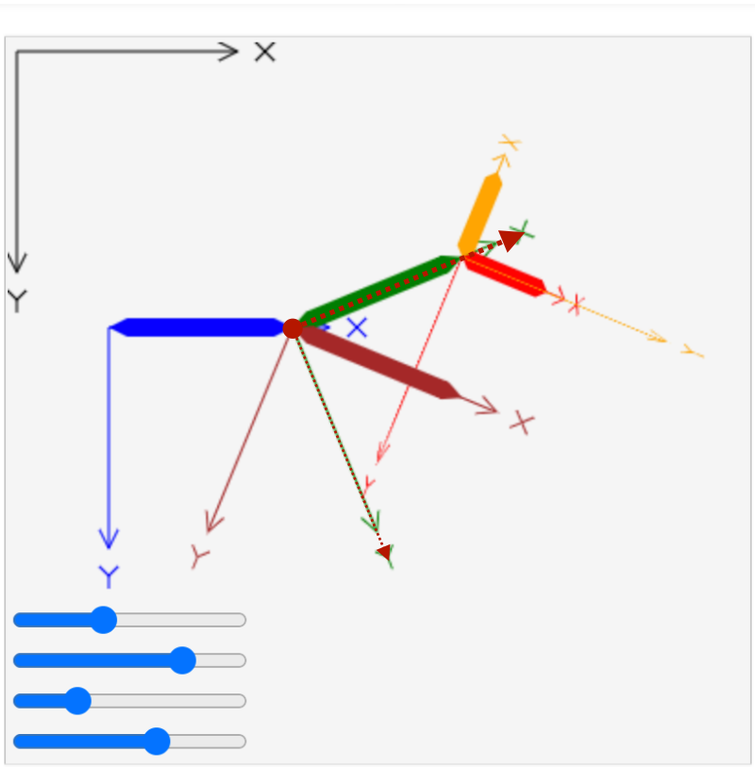
Week4/Demo1



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

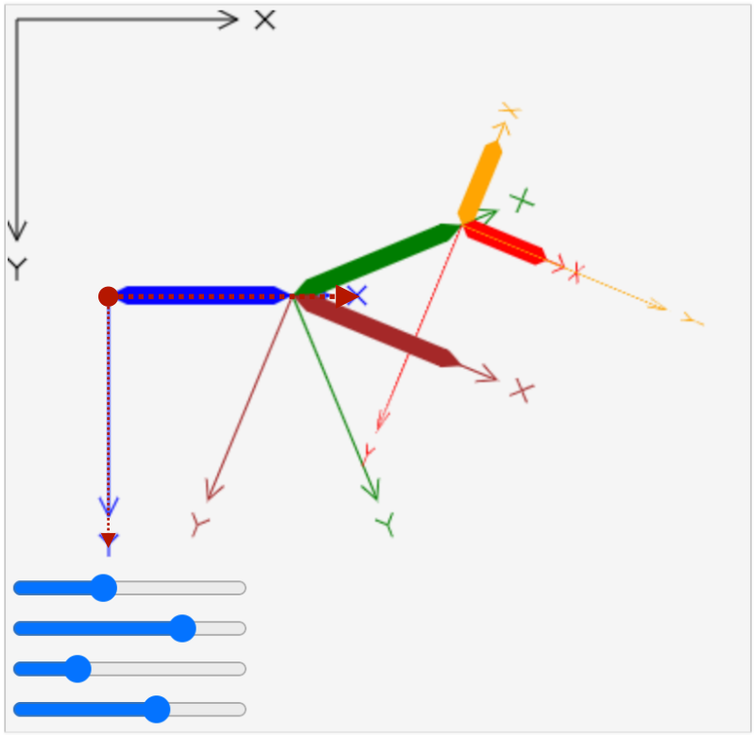
canvas-to-blue

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

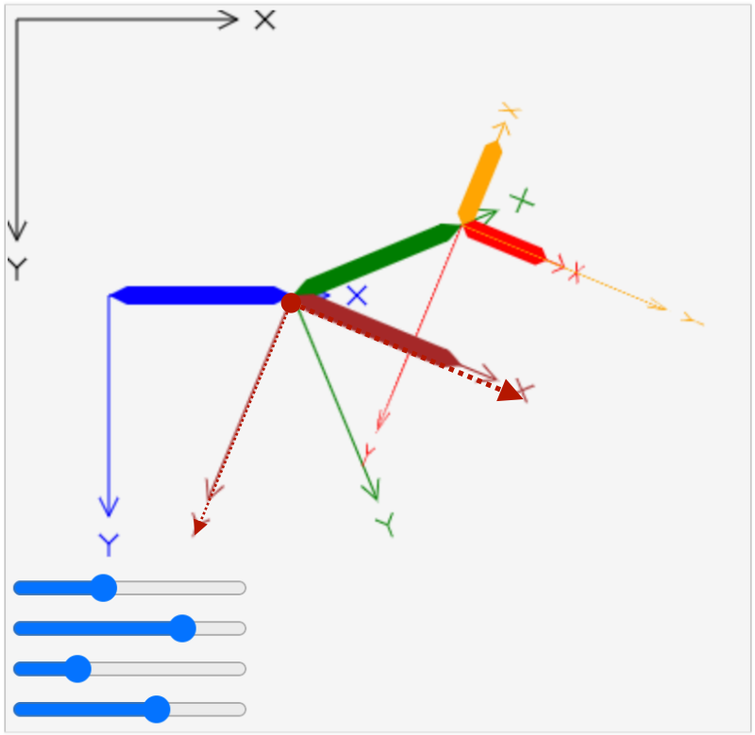
canvas-to-blue

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

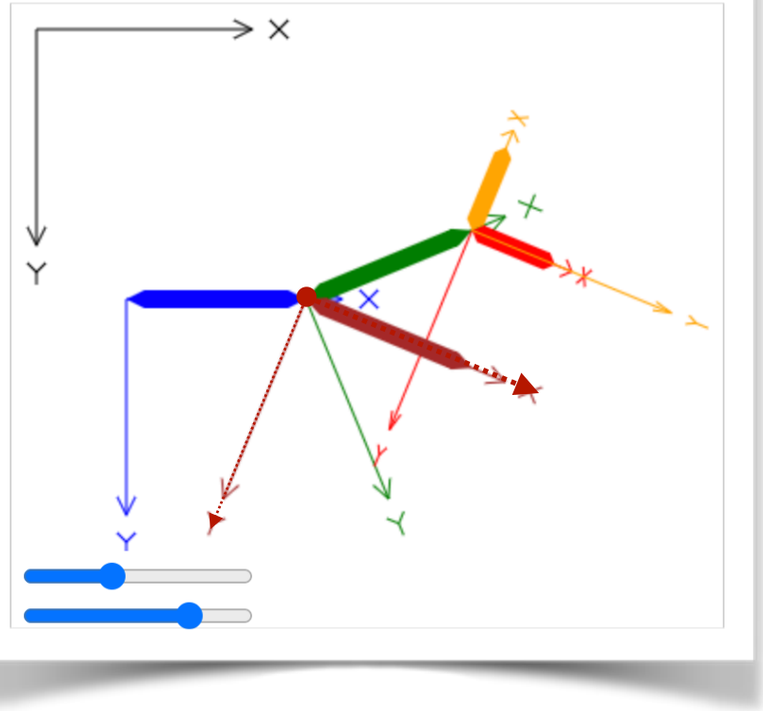
JavaScript

```
[...]
  // 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"

[...]
```

Practical implementation

Week4/Demo0b

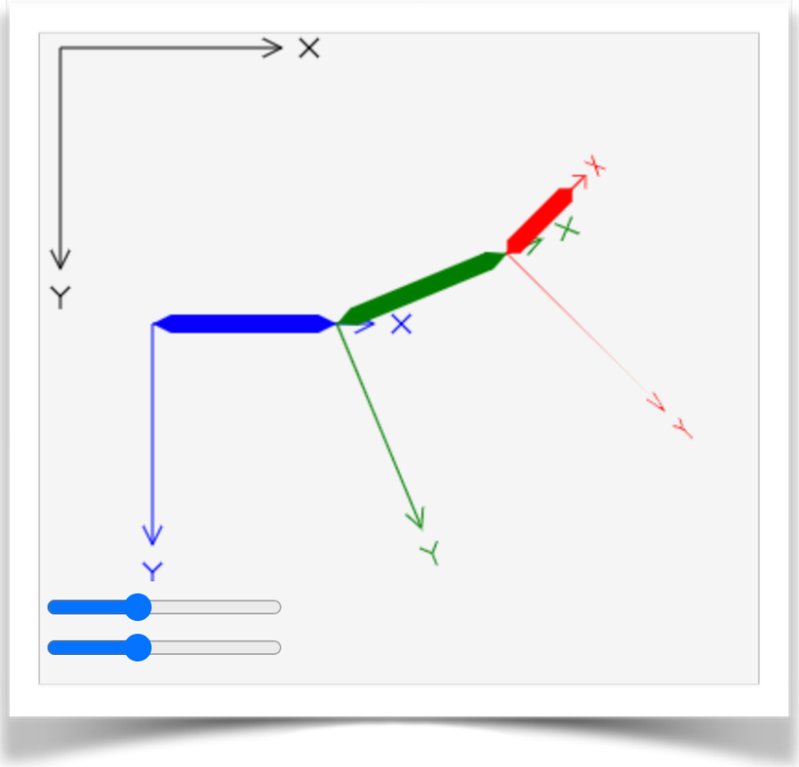
jsbin.com/qutelof

- TWGL (https://twgljs.org/) is an alternative library to use. It is much broader than glmatrix in its features (not necessarily a good thing …)
- Major point of caution! It uses *transposed* notation for matrices and vectors

$$\mathbf{x}' = \mathbf{x}\mathbf{T}$$

How are transforms combined?

jsbin.com/hozeyar



Combining transforms via multiplication works with arbitrarily long 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/qutelof

Week4/Demo0b

- TWGL (https://twgljs.org/) is an alternative library to use. It is much broader than glmatrix in its features (not necessarily a good thing …)
- Major point of caution! It uses *transposed* notation for matrices and vectors

$$\mathbf{x}' = \mathbf{x}\mathbf{T}$$

Practical implementation

jsbin.com/qutelof

Week4/Demo0b

- TWGL (https://twgljs.org/) is an alternative library to use. It is much broader than glmatrix in its features (not necessarily a good thing …)
- Major point of caution! It uses *transposed* notation for matrices and vectors

$$\mathbf{x}' = \mathbf{x}\mathbf{T}$$

$$\mathbf{T}_{combined} = \mathbf{T}_1\mathbf{T}_2\cdots\mathbf{T}_k$$

Practical implementation

jsbin.com/leropag

Week4/Demo0a

Intuition: A transform chain in glmatrix

$$\mathbf{x}' = \mathbf{T}_n \mathbf{T}_{n-1} \cdots \mathbf{T}_2 \mathbf{T}_1 \mathbf{x}$$

$$\mathbf{x}' = \mathbf{T}_{\text{combined}} \mathbf{x} \quad \text{where} \quad \mathbf{T}_{\text{combined}} = \mathbf{T}_n \mathbf{T}_{n-1} \cdots \mathbf{T}_2 \mathbf{T}_1 \mathbf{x}$$

Practical implementation

jsbin.com/leropag

Week4/Demo0a

Intuition: A transform chain in glmatrix

$$\mathbf{x}' = \mathbf{T}_n \mathbf{T}_{n-1} \cdots \mathbf{T}_2 \mathbf{T}_1 \mathbf{x}$$

$$\mathbf{x}' = \mathbf{T}_{\text{combined}} \mathbf{x} \quad \text{where} \quad \mathbf{T}_{\text{combined}} = \mathbf{T}_n \mathbf{T}_{n-1} \cdots \mathbf{T}_2 \mathbf{T}_1 \mathbf{x}$$

Transpose everything …

$$\mathbf{x}'^T = \mathbf{x}^T \mathbf{T}_1^T \mathbf{T}_2^T \cdots \mathbf{T}_{n-1}^T \mathbf{T}_1^T$$

Practical implementation

jsbin.com/leropag

Week4/Demo0a

Intuition: A transform chain in glmatrix

$$\mathbf{x}' = \mathbf{T}_n \mathbf{T}_{n-1} \cdots \mathbf{T}_2 \mathbf{T}_1 \mathbf{x}$$

$$\mathbf{x}' = \mathbf{T}_{\text{combined}} \mathbf{x} \quad \text{where} \quad \mathbf{T}_{\text{combined}} = \mathbf{T}_n \mathbf{T}_{n-1} \cdots \mathbf{T}_2 \mathbf{T}_1 \mathbf{x}$$

Transpose everything …

$$\mathbf{x}'^T = \mathbf{x}^T \mathbf{T}_1^T \mathbf{T}_2^T \cdots \mathbf{T}_{n-1}^T \mathbf{T}_1^T$$

Rename transposes of matrices and vectors …

$$\mathbf{y}' = \mathbf{y} \mathbf{M}_1 \mathbf{M}_2 \cdots \mathbf{M}_{n-1} \mathbf{M}_1$$

Practical implementation

jsbin.com/leropag

Week4/Demo0a

Intuition: A transform chain in glmatrix

$$\mathbf{x}' = \mathbf{T}_n \mathbf{T}_{n-1} \cdots \mathbf{T}_2 \mathbf{T}_1 \mathbf{x}$$

$$\mathbf{x}' = \mathbf{T}_{\text{combined}} \mathbf{x} \quad \text{where} \quad \mathbf{T}_{\text{combined}} = \mathbf{T}_n \mathbf{T}_{n-1} \cdots \mathbf{T}_2 \mathbf{T}_1 \mathbf{x}$$

Transpose everything …

$$\mathbf{x}'^T = \mathbf{x}^T \mathbf{T}_1^T \mathbf{T}_2^T \cdots \mathbf{T}_{n-1}^T \mathbf{T}_1^T$$

Rename transposes of matrices and vectors …

$$\mathbf{y}' = \mathbf{y} \mathbf{M}_1 \mathbf{M}_2 \cdots \mathbf{M}_{n-1} \mathbf{M}_1$$

… and combined transforms are formed as follows (vectors treated as row matrices)

$$\mathbf{y}' = \mathbf{y} \mathbf{M}_{\text{combined}} \quad \text{where} \quad \mathbf{M}_{\text{combined}} = \mathbf{M}_1 \mathbf{M}_2 \cdots \mathbf{M}_{n-1} \mathbf{M}_1$$

Practical implementation

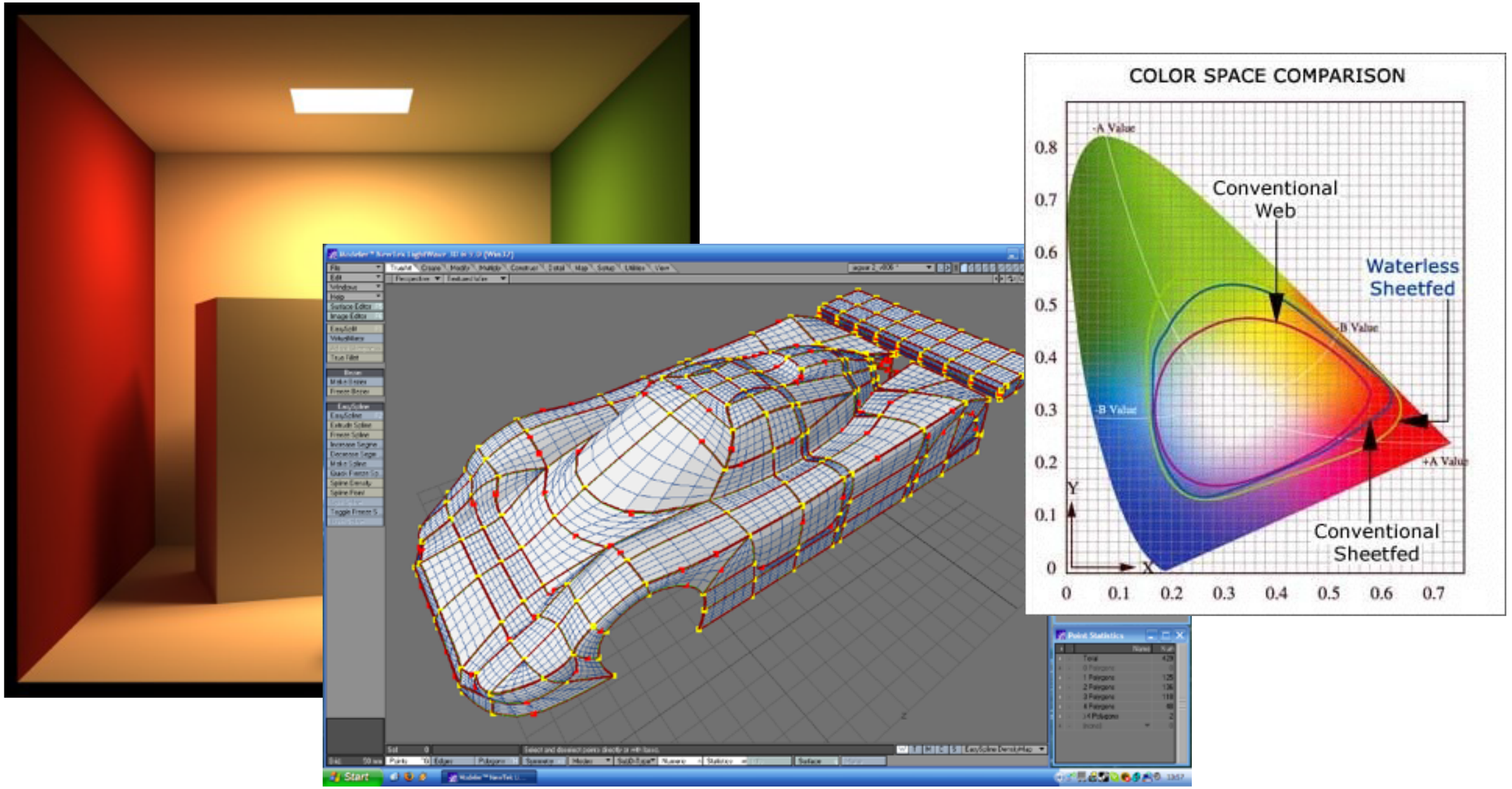
jsbin.com/qutelof

Week4/Demo0b

- TWGL (https://twgljs.org/) is an alternative library to use. It is much broader than glmatrix in its features (not necessarily a good thing …)
- Major point of caution! It uses *transposed* notation for matrices and vectors

$$\mathbf{x}' = \mathbf{x}\mathbf{T}$$

$$\mathbf{T}_{combined} = \mathbf{T}_1\mathbf{T}_2\cdots\mathbf{T}_k$$



Lecture 7 : Practical use of 2D transforms in code (in algebraic form, via glMatrix, contrasting to Canvas)

Thursday September 30th 2021