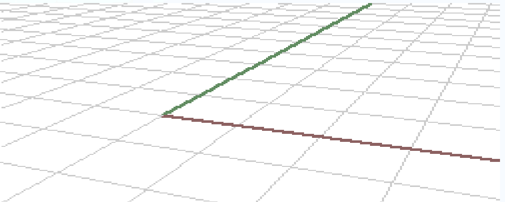
CSC 240 Computer Graphics
Video 7: Matrix Compositions

Nick Howe
Smith College

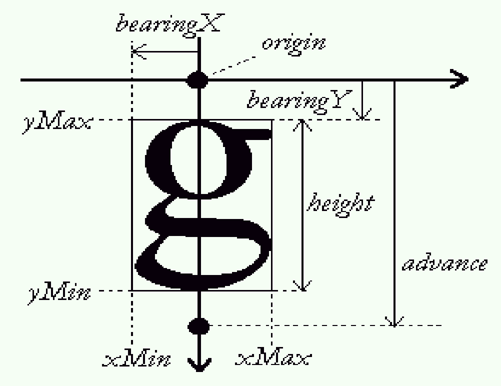
Coordinate Systems

Coordinate systems define positions in 2D space

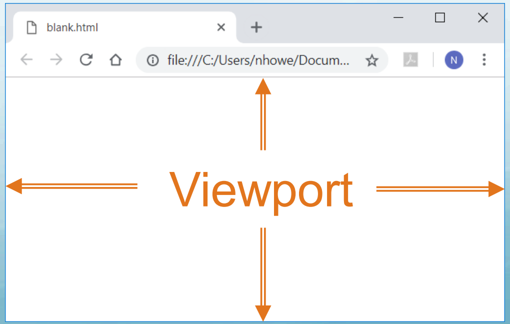
- **World coordinates** provide a global framework for all objects
- **Object coordinates** are chosen for their convenience in defining a given object
- **Viewport coordinates** define the pixels displayed on screen



P_{world}



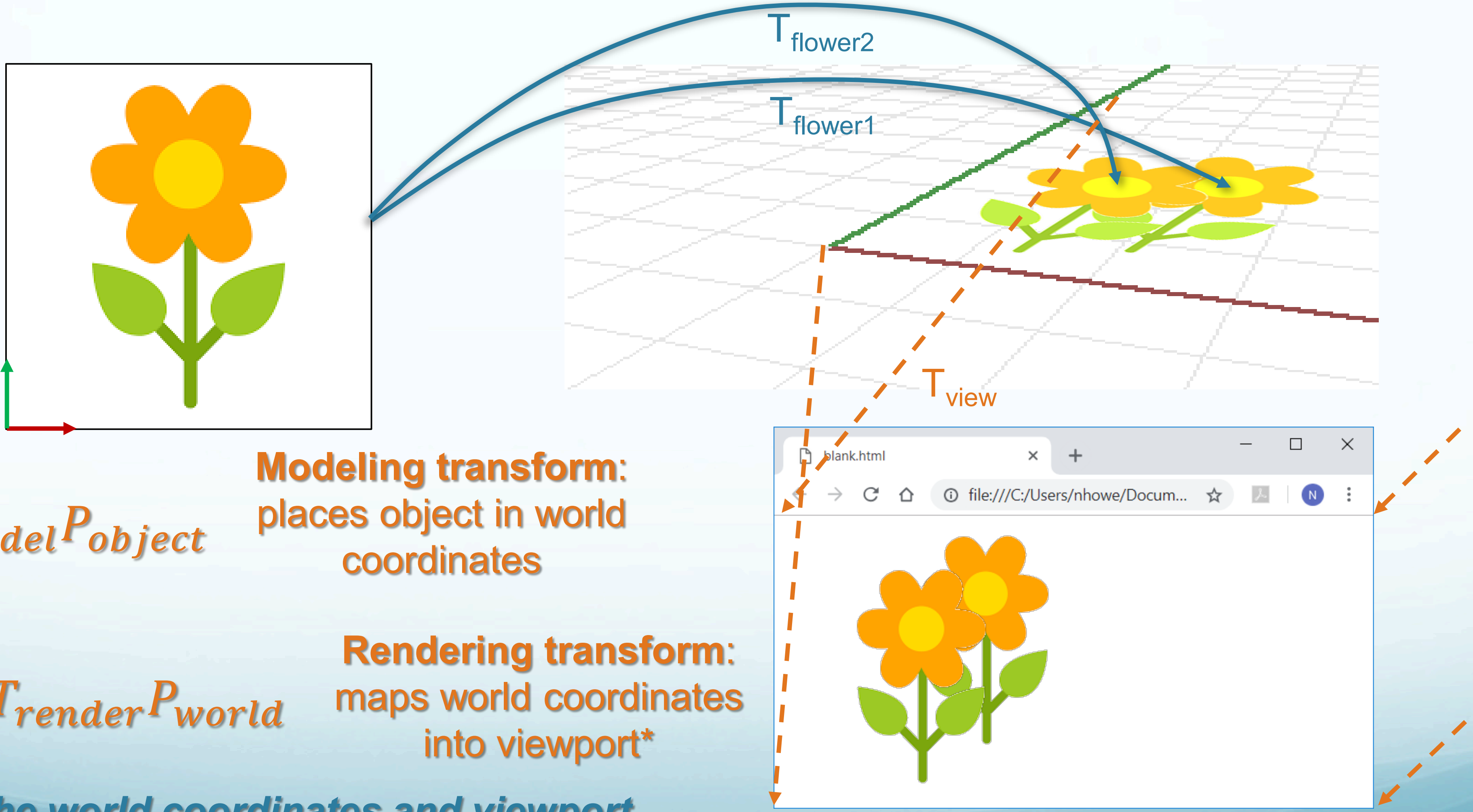
P_{object}



P_{view}

http://docs.mcneel.com/rhino/5/usersguide/en-us/html/ch-05_accuratemodeling.htm
https://bengarney.com/2011/12/07/fast-bitmap-fonts-in-flash/

Coordinate Systems



$$P_{world} = T_{model} P_{object}$$

Modeling transform: places object in world coordinates

$$P_{view} = T_{render} P_{world}$$

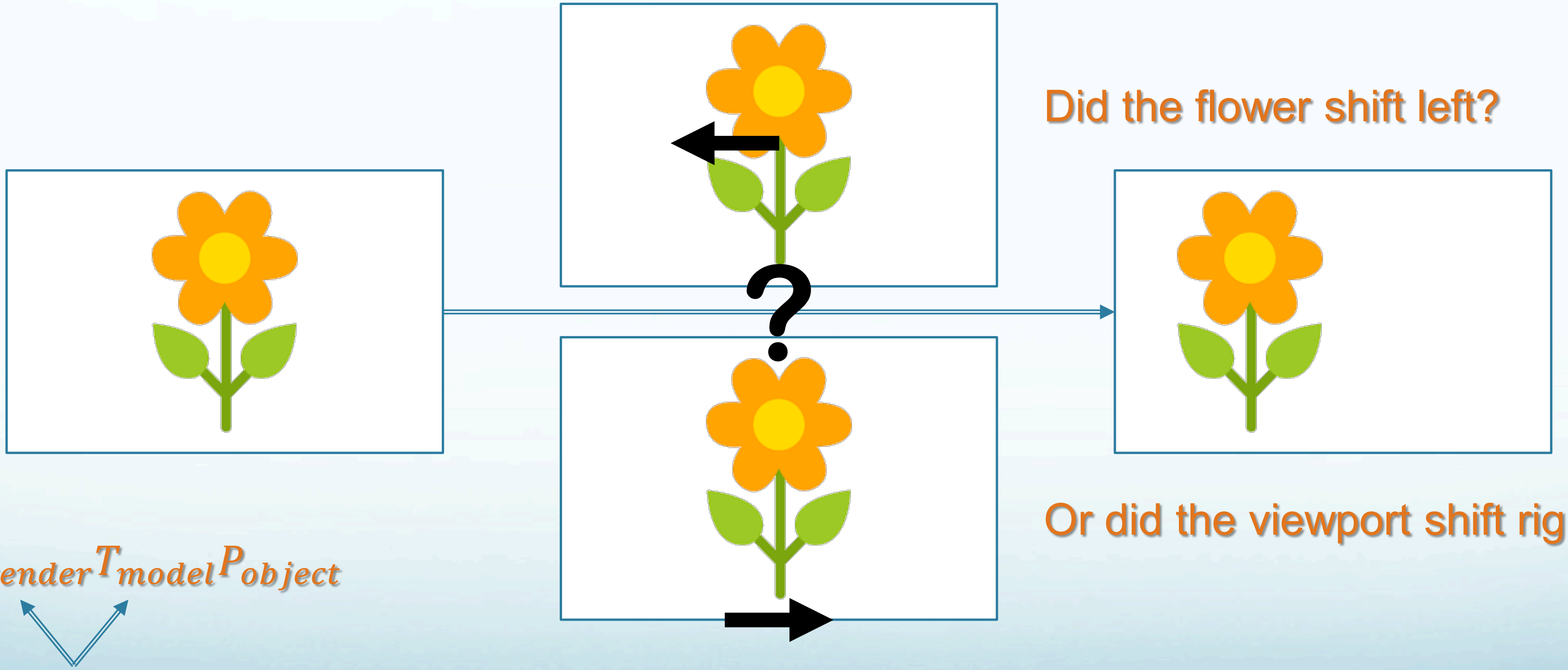
Rendering transform: maps world coordinates into viewport*

Note: in 2D, the world coordinates and viewport coordinates are often the same. In 3D they will differ.

https://www.stockio.com/free-icon/nature-icons-orange-flower

Coordinate Systems

Fundamental ambiguity: is shift in appearance caused by change to modeling transform or to viewport transform?



Did the flower shift left?

Or did the viewport shift right?

$$P_{view} = T_{render} T_{model} P_{object}$$

Which is responsible?

Disagreement on this question is the reason
Mac and Windows have different scroll behavior!

Questions

PAUSE NOW & ANSWER

1. If you see the flower as moving to the left, which transformation is responsible?

   The modeling transform

2. If you see the viewport as moving to the right, which transformation is responsible?

   The rendering transform

3. What circumstance would make it possible to resolve the dispute?

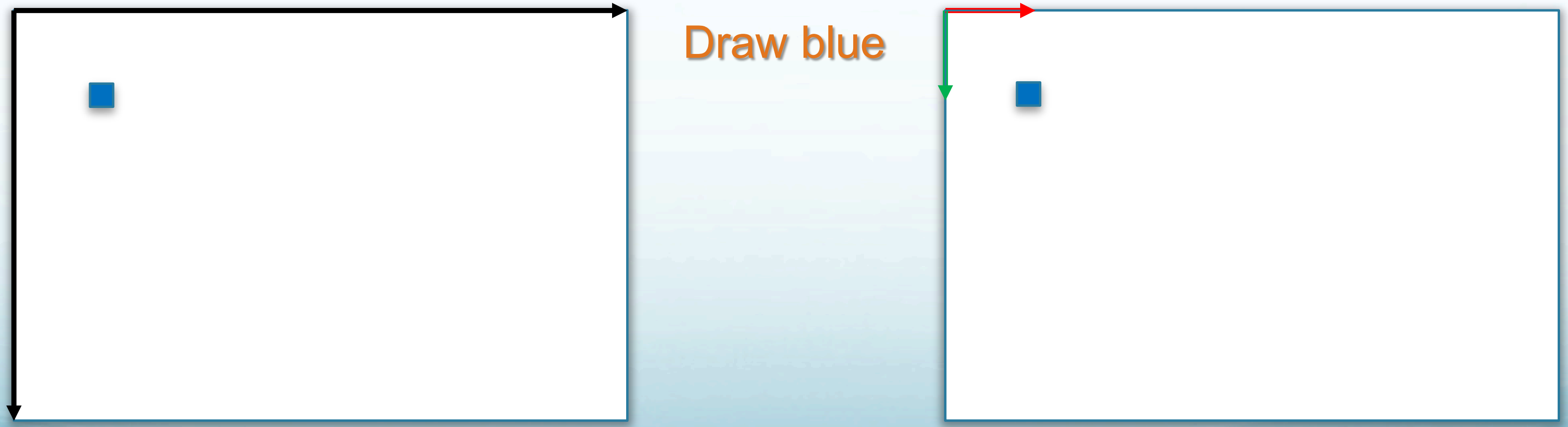
   If there were another object in the scene with fixed world coordinates, then we could tell which type of motion was responsible.

Canvas Graphics

HTML5 2D graphics (what we're using):

- Supports one current transform, applied to new drawing
- Alter transform by composing additional primitives on top
- Use inverse transform to remove last effect, or revert to saved

Two views of what is happening



Draw blue

Drawing axes

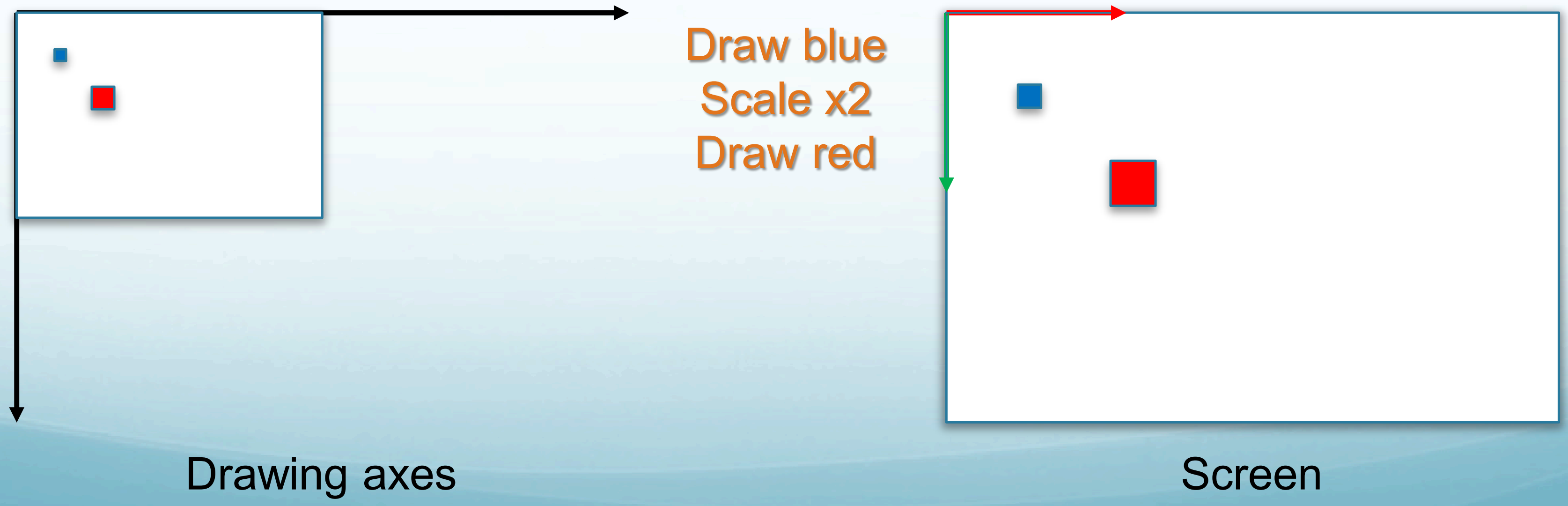
Screen

Canvas Graphics

HTML5 2D graphics (what we're using):

- Supports one current transform, applied to new drawing
- Alter transform by composing additional primitives on top
- Use inverse transform to remove last effect, or revert to saved

Two views of what is happening

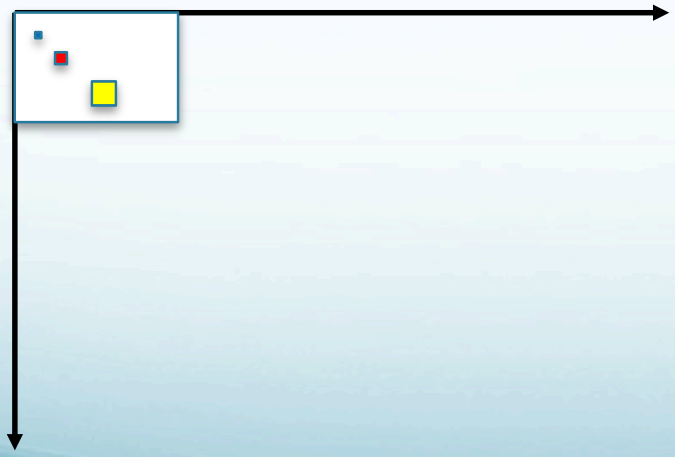


Canvas Graphics

HTML5 2D graphics (what we're using):

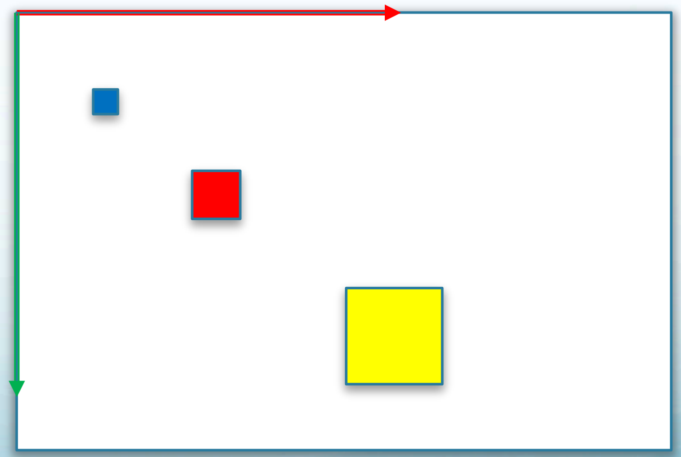
- Supports one current transform, applied to new drawing
- Alter transform by composing additional primitives on top
- Use inverse transform to remove last effect, or revert to saved

Two views of what is happening



Drawing axes

Draw blue
Scale x2
Draw red
Scale x2
Draw yellow



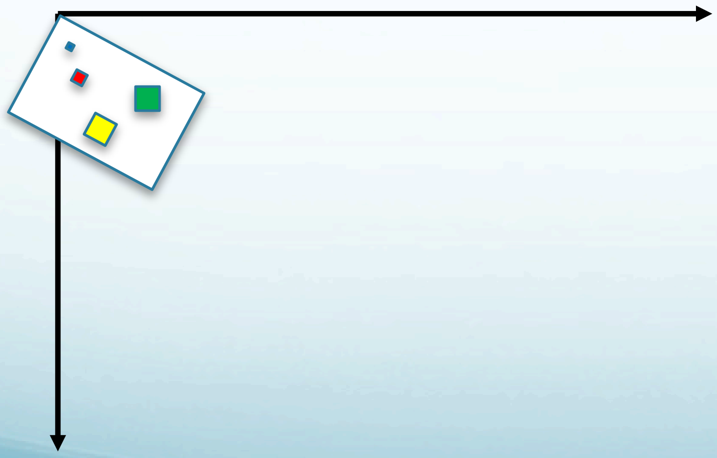
Screen

Canvas Graphics

HTML5 2D graphics (what we're using):

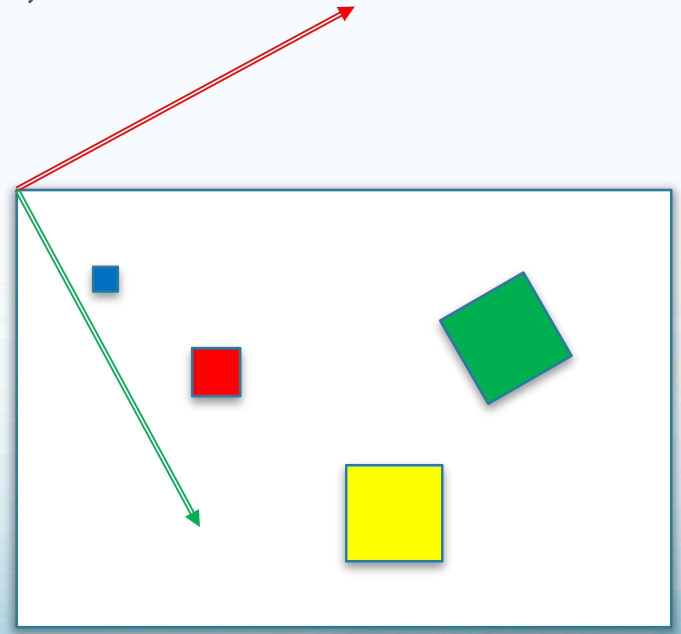
- Supports one current transform, applied to new drawing
- Alter transform by composing additional primitives on top
- Use inverse transform to remove last effect, or revert to saved

Two views of what is happening



Drawing axes

Draw blue
Scale x2
Draw red
Scale x2
Draw yellow
Rotate -30
Draw green



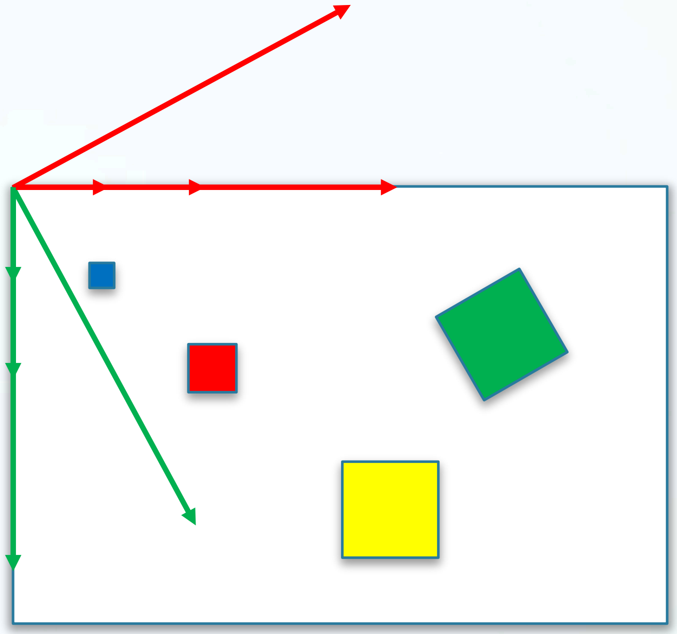
Screen

Personally, I prefer the screen view!

Specifics

Let's revisit my example with attention to the transform:

Command sequence:	Transform matrix:	Square in viewport:
`drawSquare("blue");`	I	$\begin{bmatrix} 80 & 120 & 120 & 80 \\ 80 & 80 & 120 & 120 \end{bmatrix}$
`graphics.scale(2,2);`	S	
`drawSquare("red");`	S	$\begin{bmatrix} 160 & 240 & 240 & 160 \\ 160 & 160 & 240 & 240 \end{bmatrix}$
`graphics.scale(2,2);`	SS	
`drawSquare("yellow");`	SS	$\begin{bmatrix} 320 & 480 & 480 & 320 \\ 320 & 320 & 480 & 480 \end{bmatrix}$
`graphics.rotate(-0.5);`	SSR	
`drawSquare("green");`	SSR	$\begin{bmatrix} 434 & 575 & 651 & 511 \\ 127 & 51 & 191 & 268 \end{bmatrix}$



Visualization

$$P = \begin{bmatrix} 80 & 120 & 120 & 80 \\ 80 & 80 & 120 & 120 \end{bmatrix}$$

$$S = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \quad SS = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}$$

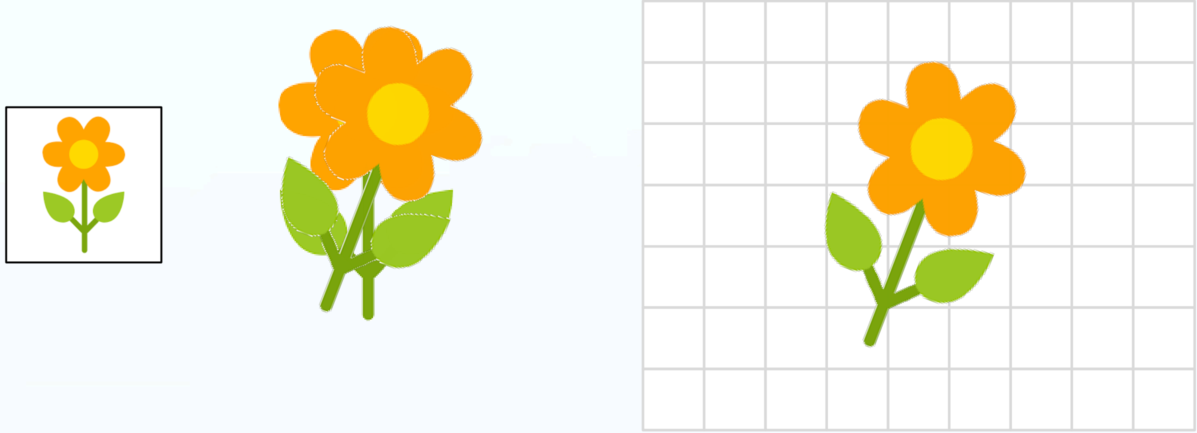
$$R = \begin{bmatrix} 0.88 & 0.48 \\ -0.48 & 0.88 \end{bmatrix}$$

$$SSR = \begin{bmatrix} 3.51 & 1.92 \\ -1.92 & 3.51 \end{bmatrix}$$

Modeling with Transforms

Typical work flow:

1. Build object
2. Add to scene using scale, rotate, translate



Note that in code, the applications must be applied in the opposite order (translate first, rotate second, scale third)

$T = I$	$P' = I \cdot P$	Initial state
$T = I \cdot T$	$P' = I \cdot T \cdot P$	Apply translate
$T = I \cdot T \cdot R$	$P' = I \cdot T \cdot R \cdot P$	Apply rotate
$T = I \cdot T \cdot R \cdot S$	$P' = I \cdot T \cdot R \cdot S \cdot P$	Apply scale

Usually best to restore previous transform when done!

Animation

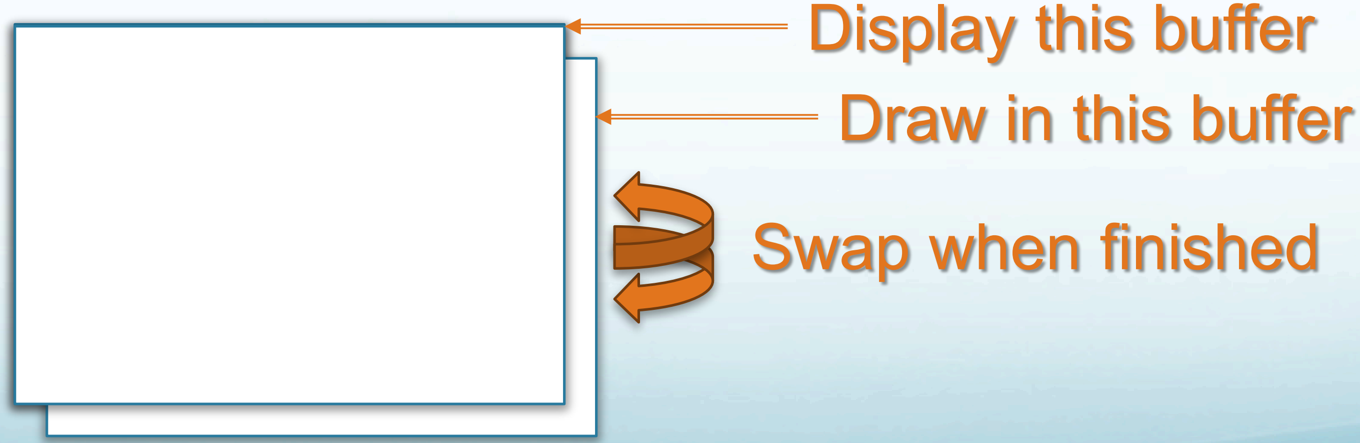
We can use transforms to do simple animation.

- Initialization: Define object appearance, initial model transform
- Infinite loop:
  1. Erase screen
  2. Draw object(s) using current transform(s)
  3. Update transform for next iteration

https://www.deviantart.com/flutterluv/art/A-Cow-Jumped-Over-The-Moon-622575145

Buffering

- For animation, need to draw and redraw graphics
- Complex renderings take time to produce
- Don't want user to see drawing process
- Solution: **double buffering**



Hierarchical Modeling

- Complex objects can be built up of subparts
- Overall object has one modeling transform
  - Subparts apply their own transform on top of the parent’s
    - If they have subparts, they can apply yet another, etc.
  - After adding each subpart, revert to the parent transform
- When animating, a change to the transform of the overall object modifies all the subparts as well

DEMO

https://www.canstockphoto.com/kicking-mannequin-4280471.html
https://www.canstockphoto.com/wooden-mannequin-lying-43108622.html

Questions

PAUSE NOW & ANSWER

A hierarchical model is used for a steam locomotive, with the boiler as the root and the hierarchy shown

1. What is the full transformation applied to driver wheel #1?

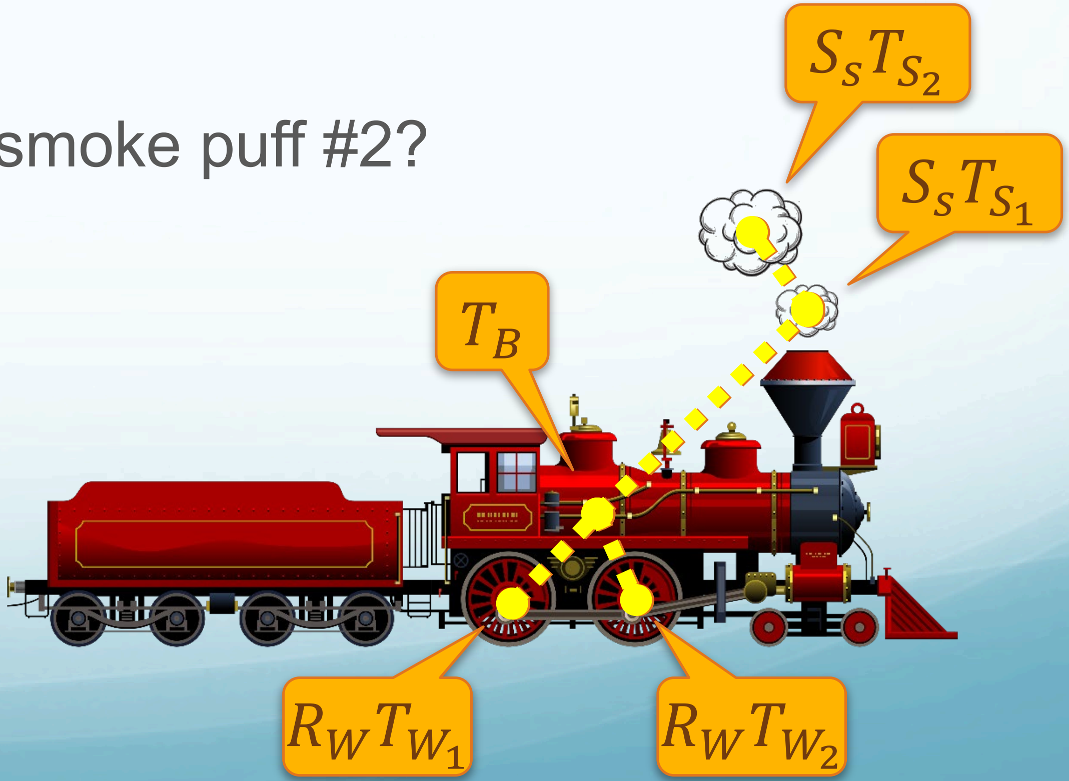
   $R_W T_{W_1} T_B$

2. What is the full transformation applied to smoke puff #2?

   $S_s T_{S_2} S_s T_{S_1} T_B$

3. Why can the smoke puffs use the same scale transformation but need different translations?

   They grow by the same amount, but the movements are different.



HTML5 2D Graphics

Object creation:

- `fillRect(x,y,w,h)`
- `strokeRect(x,y,w,h)`
- `clearRect(x,y,w,h)`
- `fillText(str,x,y)`
- `strokeText(str,x,y)`

Transforms:

$$\begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix}$$

- `scale(sx,sy)`
- `rotate(theta)`
- `translate(dx,dy)`
- `transform(a,b,c,d,e,f)`

(Apply to current transform)

- `setTransform(a,b,c,d,e,f)`

(Replace current transform)

- `save()`
- `restore()`

When working on a subpart,
save the parent transform
and restore it later

Animations

- Need something here

Review

After watching this video, you should be able to…

- List three types of coordinate systems used in graphics
- Describe the transforms used to relate the different coordinate systems
- Recognize that the same effect can be achieved through either transform
- Know how the current transform affects objects added to a scene
- Manipulate the current transform to achieve desired effects
- Understand that transforms accumulate through object part hierarchies

Music: https://www.bensound.com