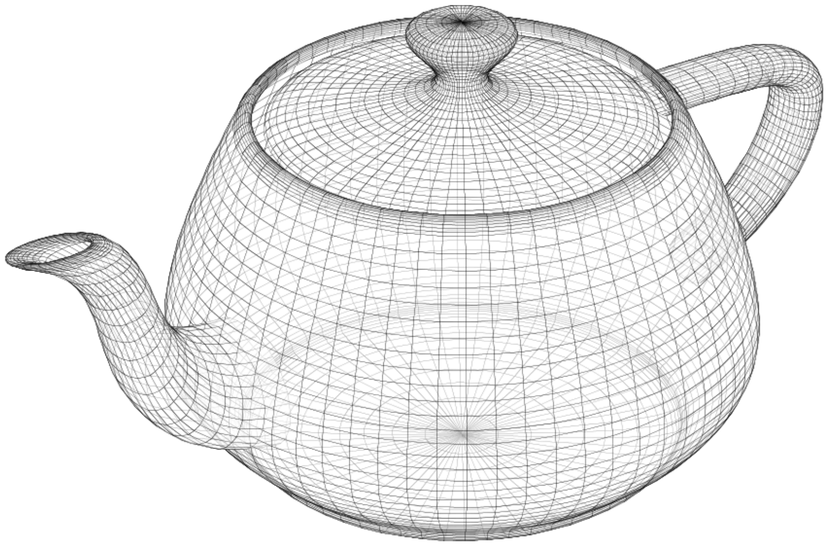
Affine Transformations

Scale and Translation



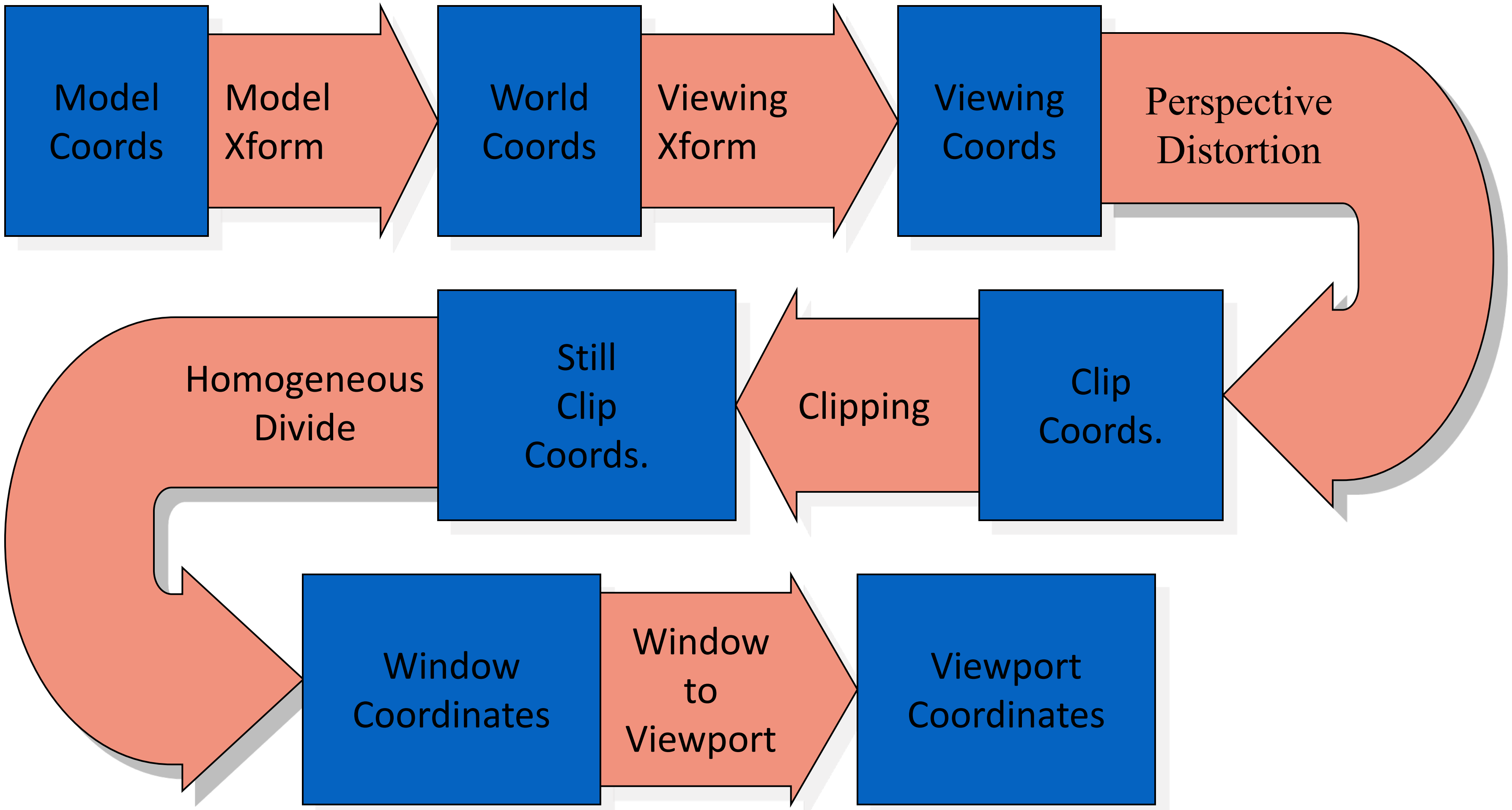
CS 418: Interactive Computer Graphics

Professor Eric Shaffer

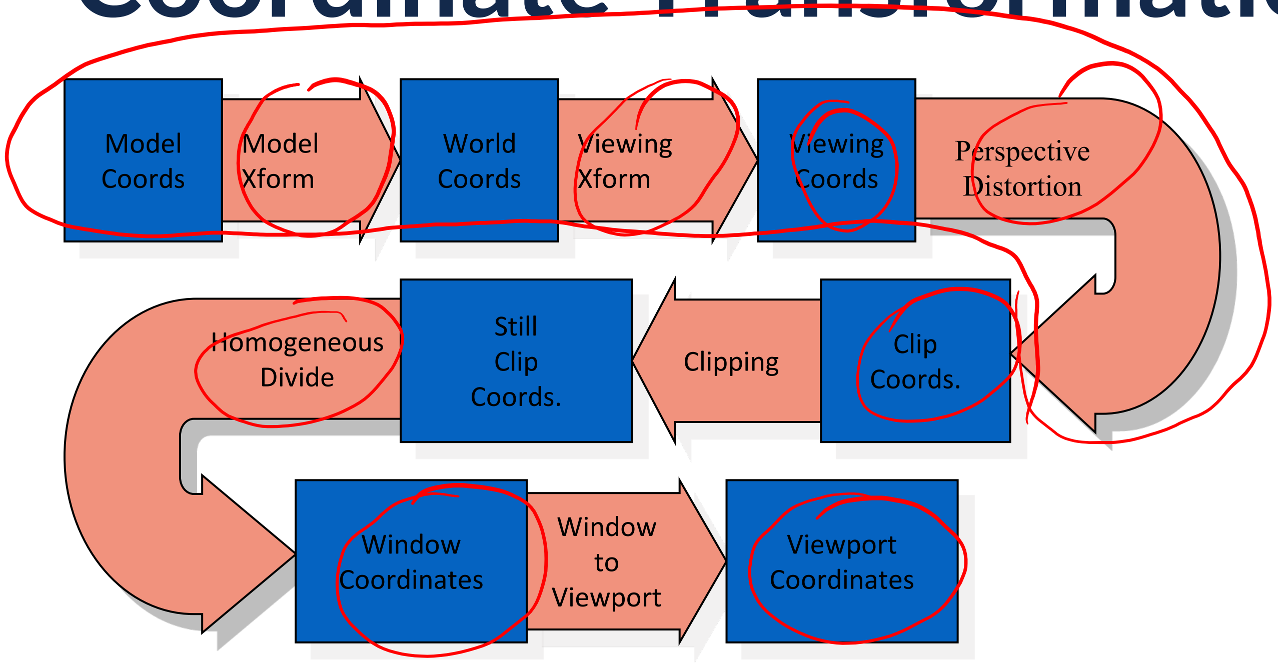
Slides courtesy of Professor John Hart

ILLINOIS

Rendering Pipeline: Coordinate Transformations



Rendering Pipeline: Coordinate Transformations



Not everyone uses the same terminology… in Unity, viewport coordinates are in the range [-1,1]

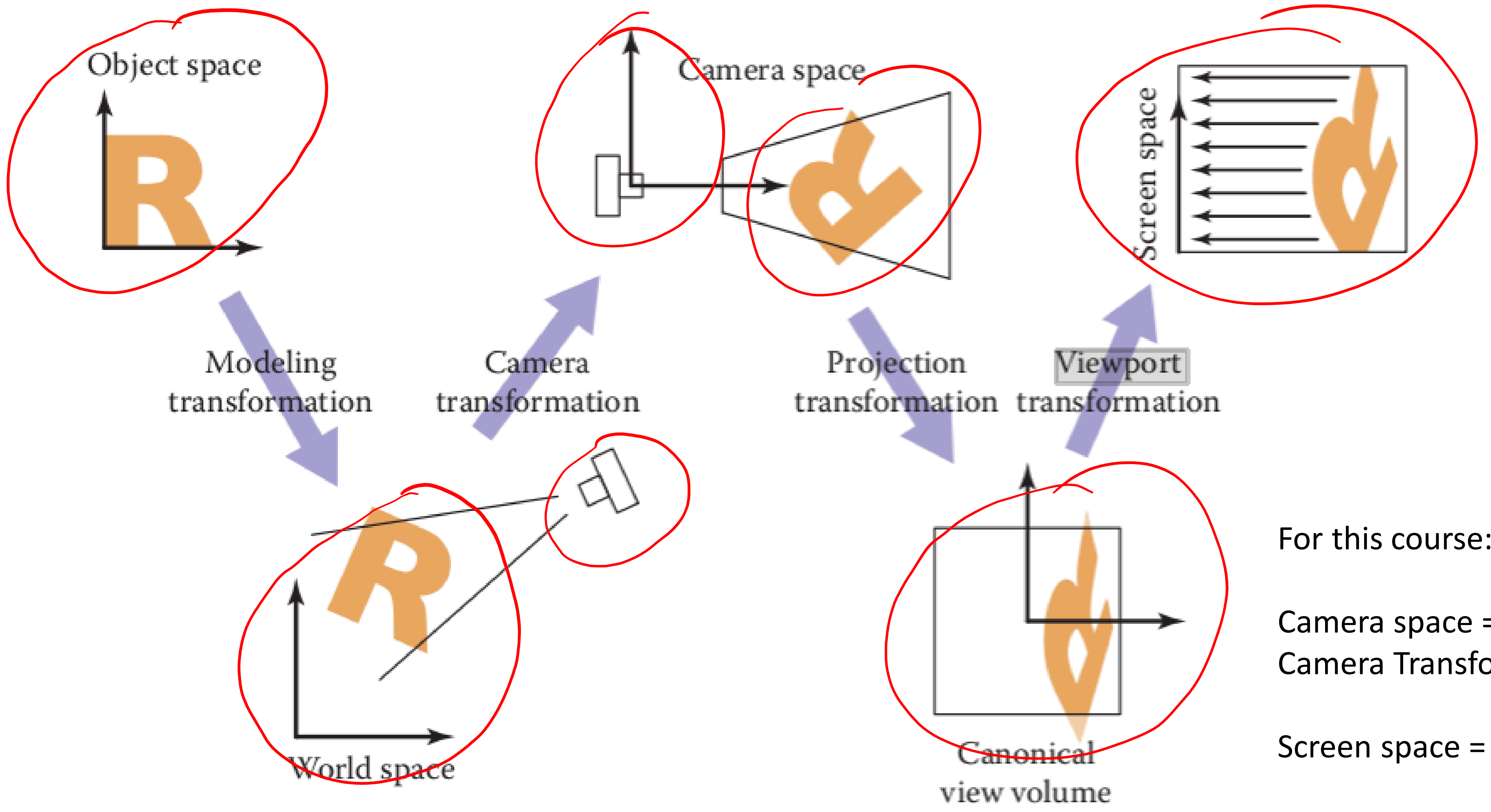
we'll use what you see listed here

Window Coordinates: 2D and in range [-1,1]
Viewport Coordinates: pixel coordinates

What stages that you see here happen in the vertex shader?

Most of the transformations described here are accomplished by matrix-vector multiplications…so we will review some linear algebra

Another way of looking at it…



From Fundamentals of Computer Graphics by Marschner and Shirley

For this course:

Camera space = View space
Camera Transformation = View Transformation

Screen space = Viewport coordinates

A Brief Sampling of Useful Math

Definition of GEOMETRY

plural **geometries**

1 **a** : a branch of mathematics that deals with the measurement, properties, and relationships of points, lines, angles, surfaces, and solids; *broadly* : the study of properties of given elements that remain invariant under specified transformations

We will look at three basic geometric elements

Scalars: Encode a magnitude
Vectors: Encode a magnitude and direction
Points: Encode a position in space

Operations on Vectors

- Scalar-vector multiplication $u = \alpha v$
- Vector-vector addition: $w = u + v$
  - Allows expressions such as $v = u + 2w - 3r$

- Vectors lack position

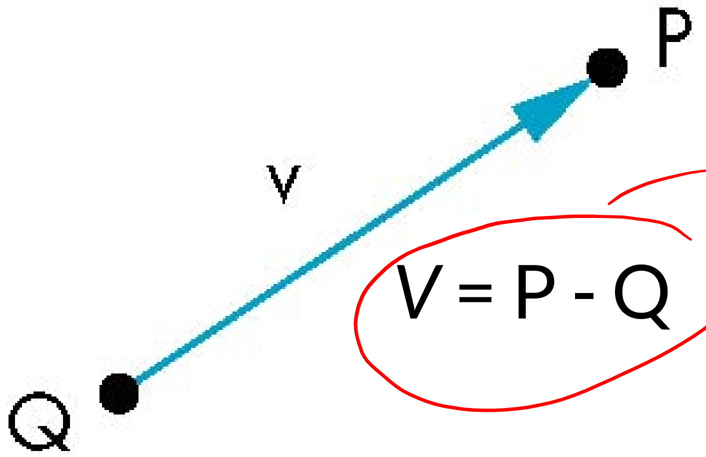
…need points to make things interesting

Points

Location in space

Operations allowed between points and vectors

- Point-point subtraction yields a vector
- Equivalent to point-vector addition



$$P = V + Q$$

$$V = P - Q$$

Linear Transformations

A transformation f is linear if

$$f(a\mathbf{u} + b\mathbf{v}) = af(\mathbf{u}) + bf(\mathbf{v})$$

for vectors **u** and **v** and scalars a and b.

In other words: doesn't matter if we add the vectors and then apply the map, or apply the map and then add the vectors…same for scaling

Linear transformations have a geometric interpretation and can be applied to points or vectors.

In two-dimensional space R^2 linear transformations are described by 2 × 2 real matrices.

- rotation by 90 degrees counterclockwise:

$$\mathbf{A} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

- rotation by angle θ counterclockwise:

$$\mathbf{A} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

- reflection against the x axis:

$$\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

- reflection against the y axis:

$$\mathbf{A} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$

- scaling by 2 in all directions:

$$\mathbf{A} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

- horizontal shear mapping:

$$\mathbf{A} = \begin{pmatrix} 1 & m \\ 0 & 1 \end{pmatrix}$$

- squeeze mapping:

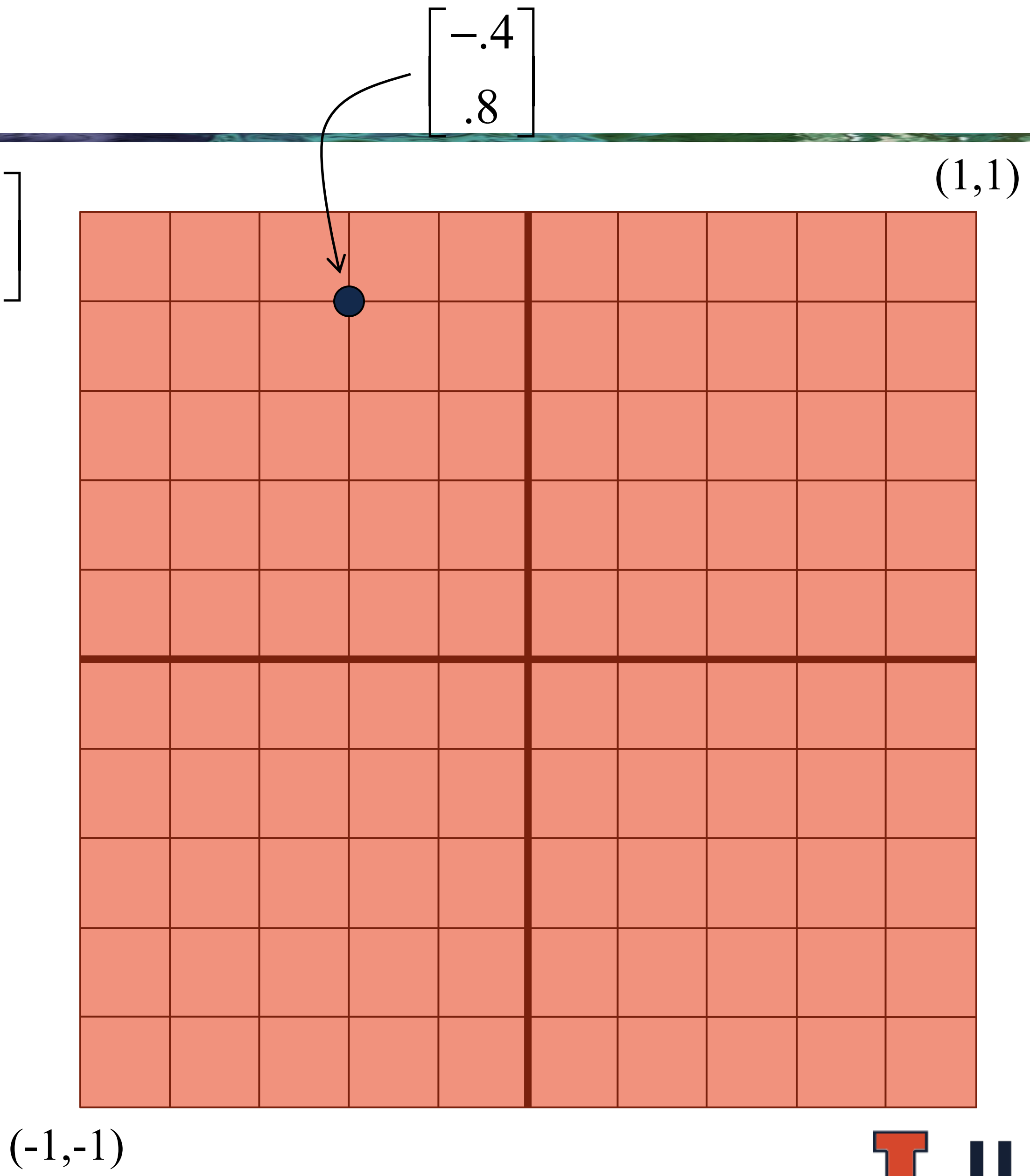
$$\mathbf{A} = \begin{pmatrix} k & 0 \\ 0 & 1/k \end{pmatrix}$$

- projection onto the y axis:

$$\mathbf{A} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}.$$

2-D Points

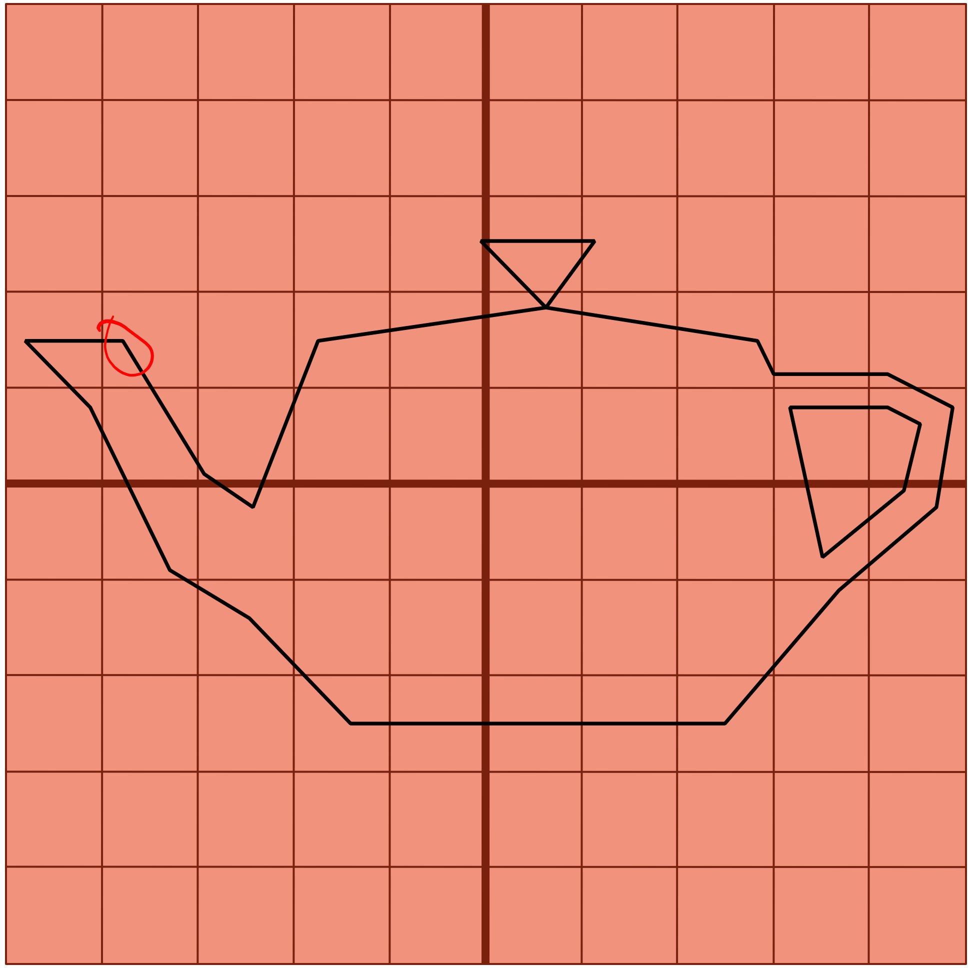
- Represents points and vertices as column vectors: $\begin{bmatrix} x \\ y \end{bmatrix}$



2-D Points

- Represents points and vertices as column vectors: $\begin{bmatrix} x \\ y \end{bmatrix}$
- Transform polygonal object by transforming its vertices

(1,1)



(-1,-1)

2-D Points

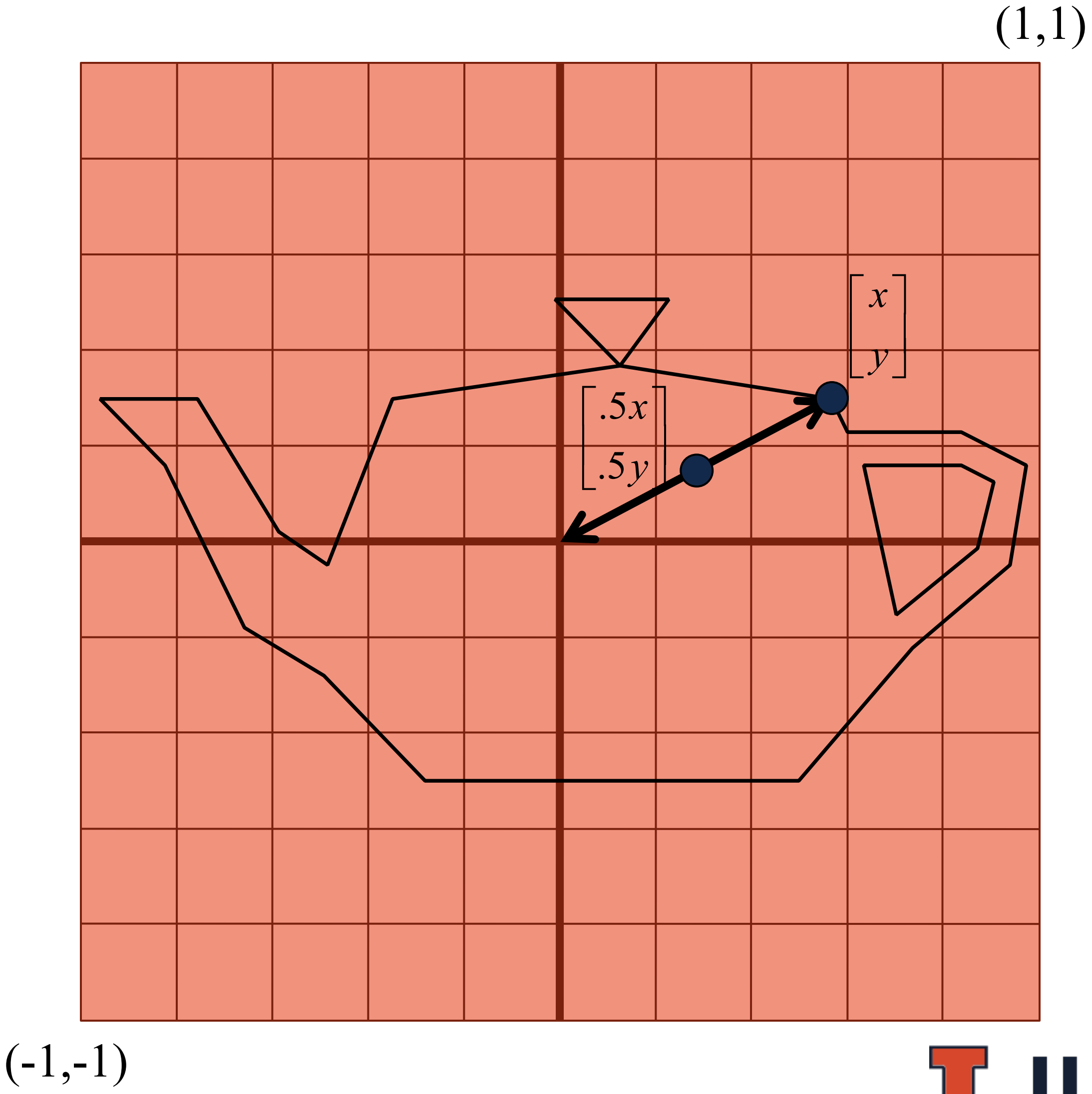
$$\begin{bmatrix} .5 & 0 \\ 0 & .5 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}$$

- Represents points and vertices as column vectors: $\begin{bmatrix} x \\ y \end{bmatrix}$

- Transform polygonal object by transforming its vertices

- Scale by matrix multiplication

$$\begin{bmatrix} a & 0 \\ 0 & a \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax \\ ay \end{bmatrix}$$



2-D Points

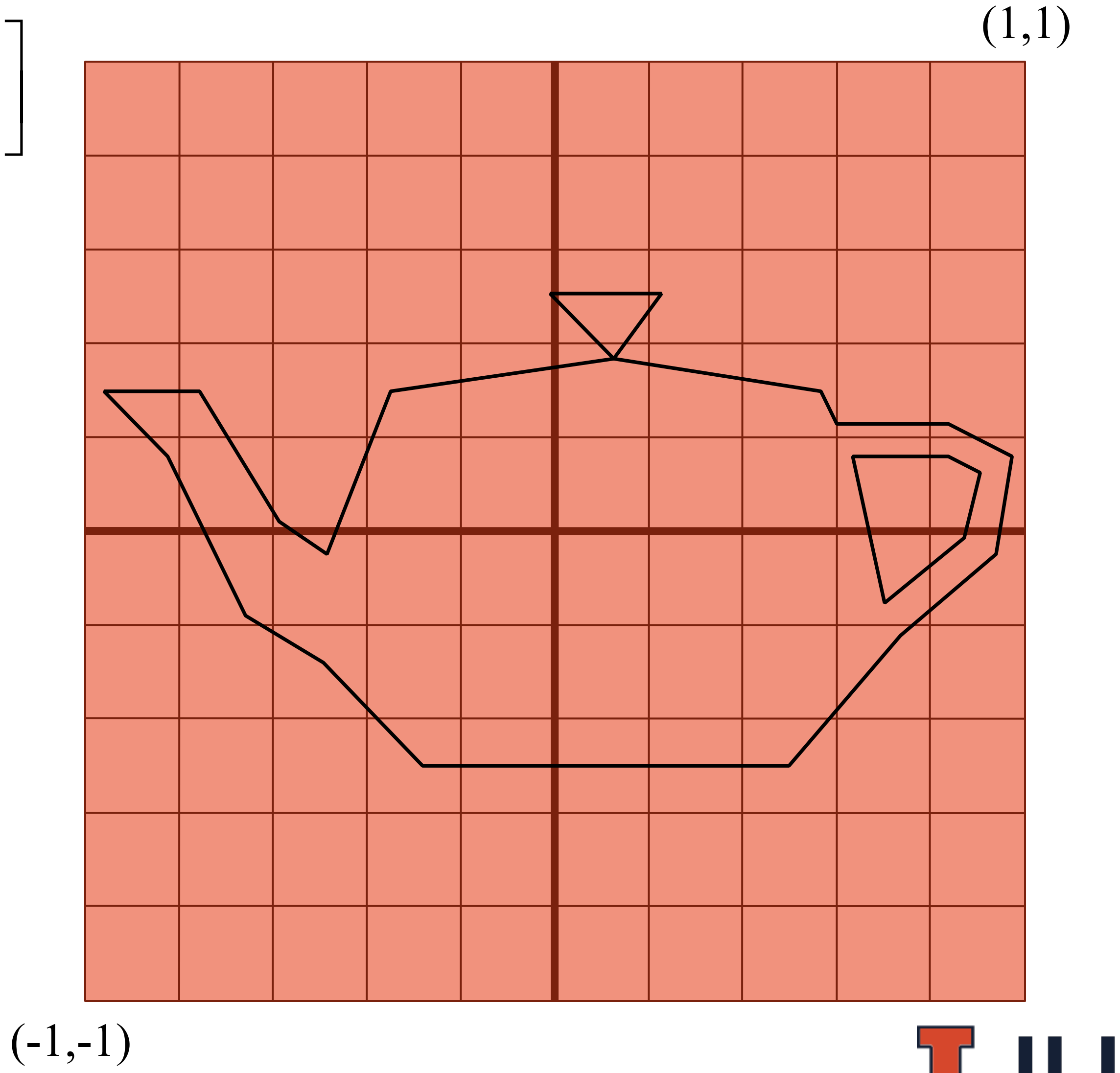
$$\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ -.4 \end{bmatrix}$$

- Represents points and vertices as column vectors: $\begin{bmatrix} x \\ y \end{bmatrix}$
- Transform polygonal object by transforming its vertices
- Scale by matrix multiplication

$$\begin{bmatrix} a & 0 \\ 0 & a \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax \\ ay \end{bmatrix}$$

- Translation via vector sum

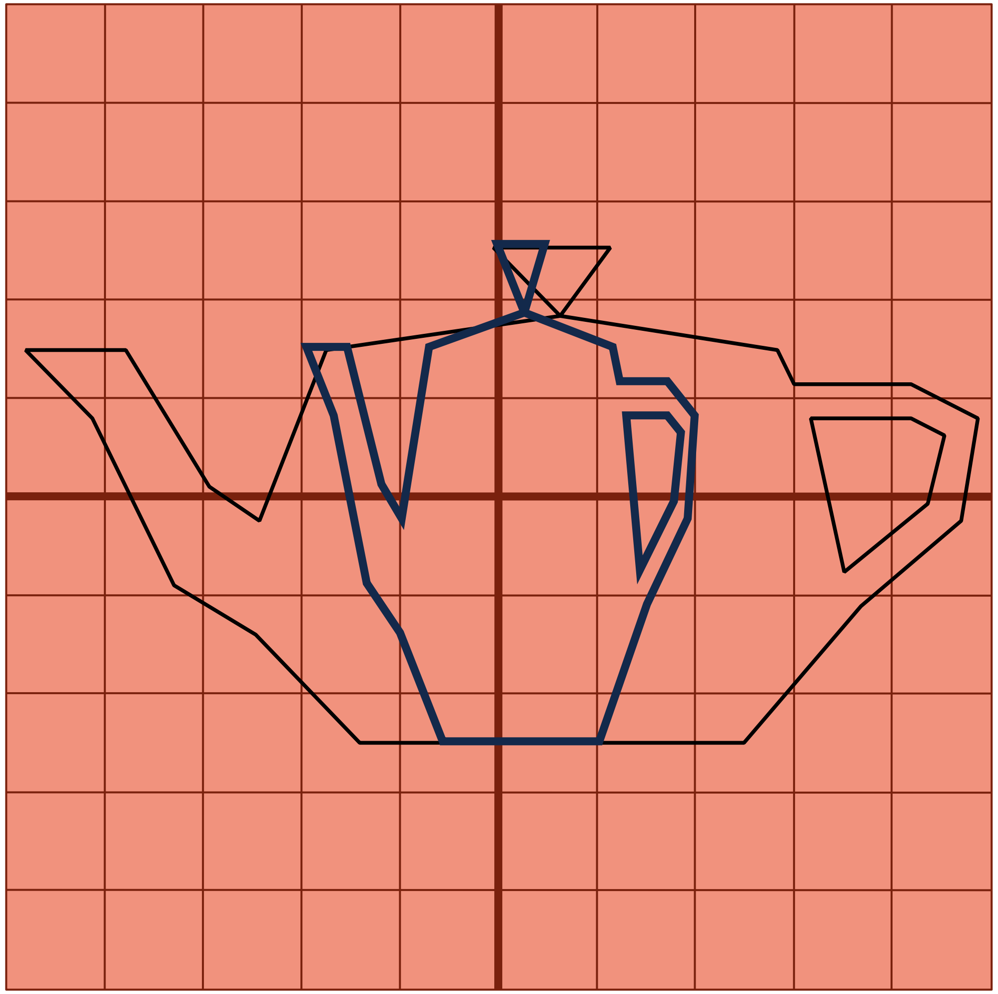
$$\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} x+a \\ y+b \end{bmatrix}$$



Squash & Stretch

Classic animation technique

Scale one coordinate by matrix multiplication



2-D Points

$$\begin{bmatrix} .5 & 0 \\ 0 & .5 \end{bmatrix}\left(\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ -.4 \end{bmatrix}\right)$$

$$\begin{bmatrix} .5 & 0 \\ 0 & .5 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ -.4 \end{bmatrix}$$

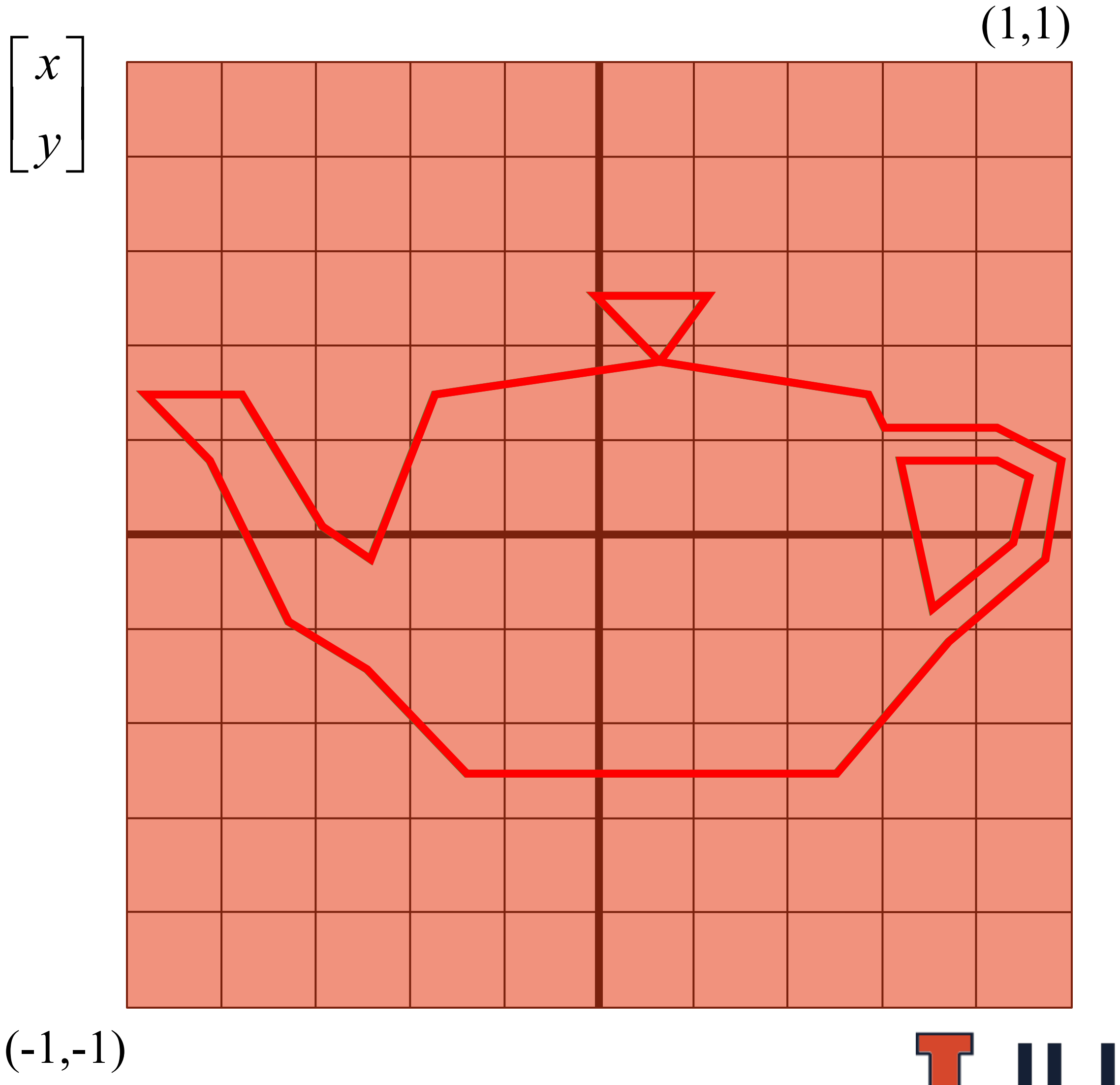
- Represent points and vertices as column vectors: $\begin{bmatrix} x \\ y \end{bmatrix}$
- Transform polygonal object by transforming its vertices
- Scale by matrix multiplication

$$\begin{bmatrix} a & 0 \\ 0 & a \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax \\ ay \end{bmatrix}$$

- Translation via vector sum

$$\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} x+a \\ y+b \end{bmatrix}$$

- Order is important
  - Translate then scale
  - Scale then translate



ILLINOIS

Homogeneous Coordinates

$$\begin{bmatrix} .5 & 0 \\ 0 & .5 \end{bmatrix}\left(\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ -.4 \end{bmatrix}\right)$$

$$\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} x+a \\ y+b \end{bmatrix}$$

- Translation by vector sum is cumbersome
- Add a extra coordinate
  - Called the homogeneous coordinate
  - For now, set to one
- Translation now expressed as a matrix
- Now we can compose scales and translations into a single matrix by matrix multiplication

$$\begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x+a \\ y+b \\ 1 \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -.4 \\ 0 & 0 & 1 \end{bmatrix}}_{T}\underbrace{\begin{bmatrix} .5 & 0 & 0 \\ 0 & .5 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{S} = \begin{bmatrix} .5 & 0 & 0 \\ 0 & .5 & -.4 \\ 0 & 0 & 1 \end{bmatrix}$$

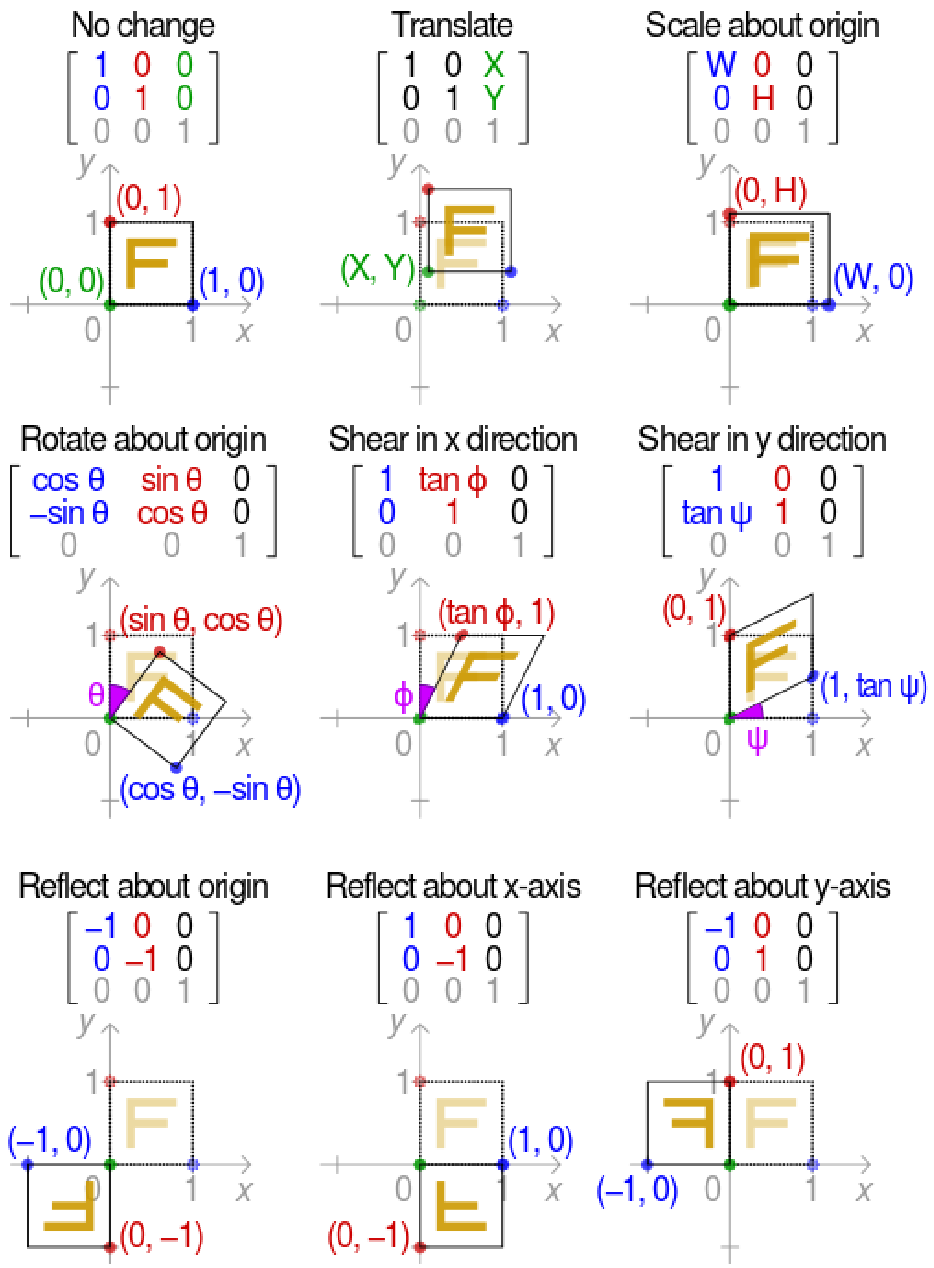
ILLINOIS

Affine Transformations

An affine transformation is the sum of a linear transformation and a constant vector…

Linear transformations preserve the origin

Translations map the origin to a new position



Order Dependence

$$\begin{bmatrix} .5 & 0 \\ 0 & .5 \end{bmatrix}\left(\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ -.4 \end{bmatrix}\right)$$

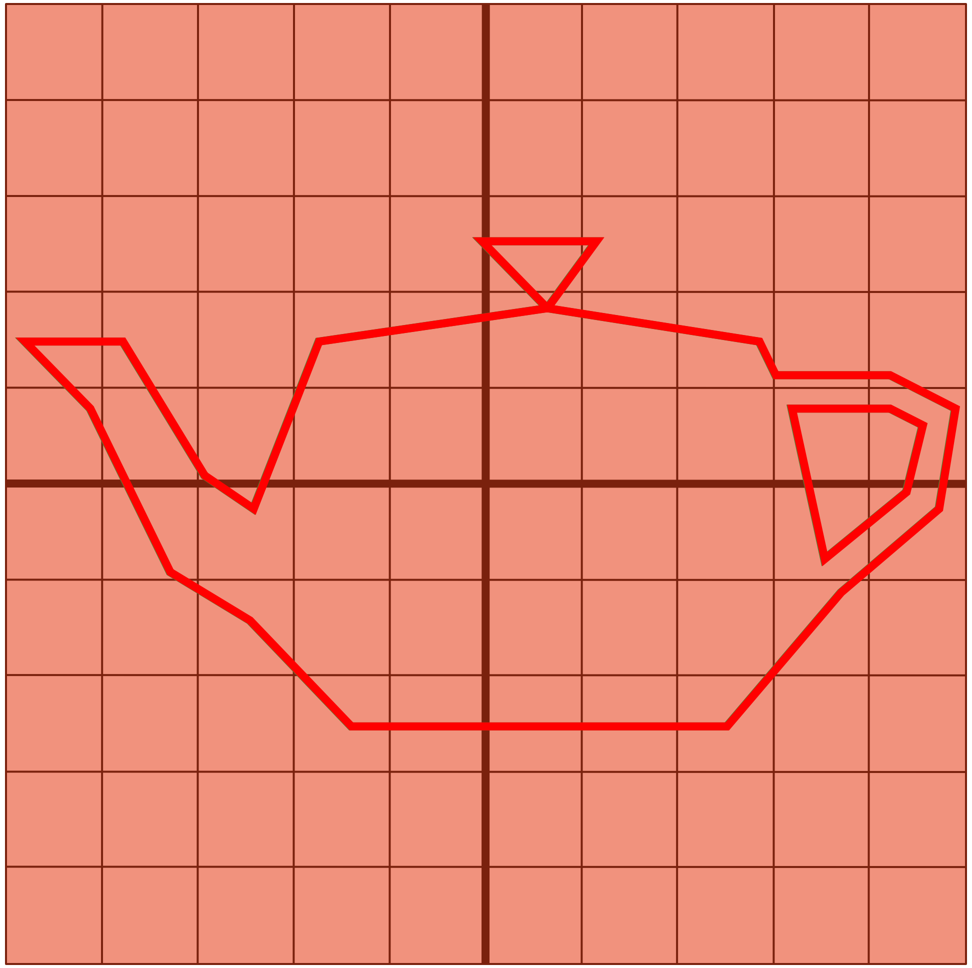
$$\begin{bmatrix} .5 & 0 & 0 \\ 0 & .5 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -.4 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} .5 & 0 & 0 \\ 0 & .5 & -.2 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} .5 & 0 \\ 0 & .5 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ -.4 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -.4 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} .5 & 0 & 0 \\ 0 & .5 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} .5 & 0 & 0 \\ 0 & .5 & -.4 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



(-1,-1)

Window-to-Viewport

- First translate lower-left corner to (0,0)

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

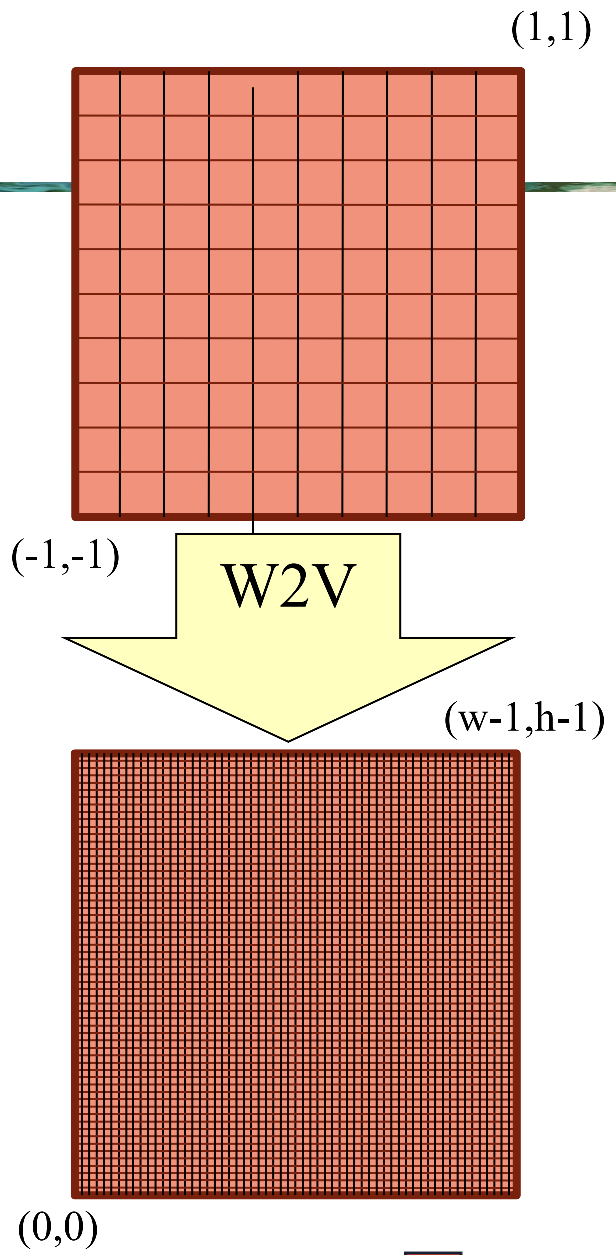
- Then scale upper-right corner from (2,2) to (w-1,h-1)

$$\begin{bmatrix} \frac{w-1}{2} & 0 & 0 \\ 0 & \frac{h-1}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- To get

$$\begin{bmatrix} \frac{w-1}{2} & 0 & \frac{w-1}{2} \\ 0 & \frac{h-1}{2} & \frac{h-1}{2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

This is the transformation WebGL and DirectX10+ use…pixel centers are at offsets of 0.5 from integer values



ILLINOIS

Window-to-Viewport

- First translate lower-left corner to (0,0)

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

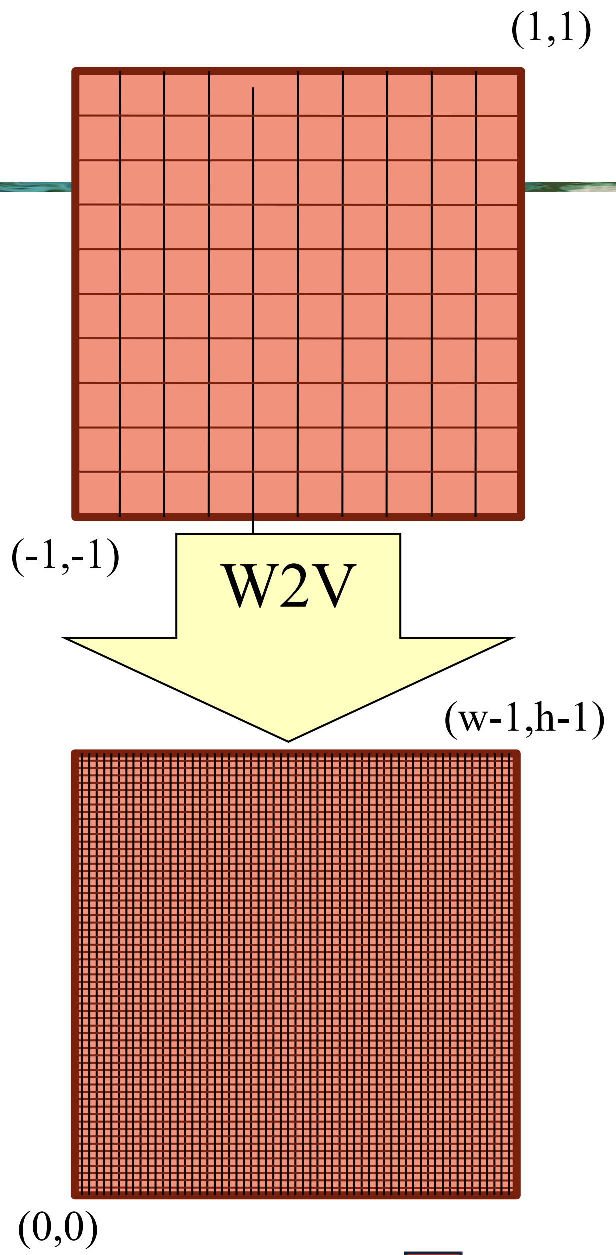
- Then scale upper-right corner from (2,2) to (w-1,h-1)

$$\begin{bmatrix} \frac{w-1}{2} & 0 & 0 \\ 0 & \frac{h-1}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- To get

$$\begin{bmatrix} \frac{w-1}{2} & 0 & \frac{w-1}{2} \\ 0 & \frac{h-1}{2} & \frac{h-1}{2} \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

This is the transformation WebGL and DirectX10+ use…pixel centers are at offsets of 0.5 from integer values



ILLINOIS