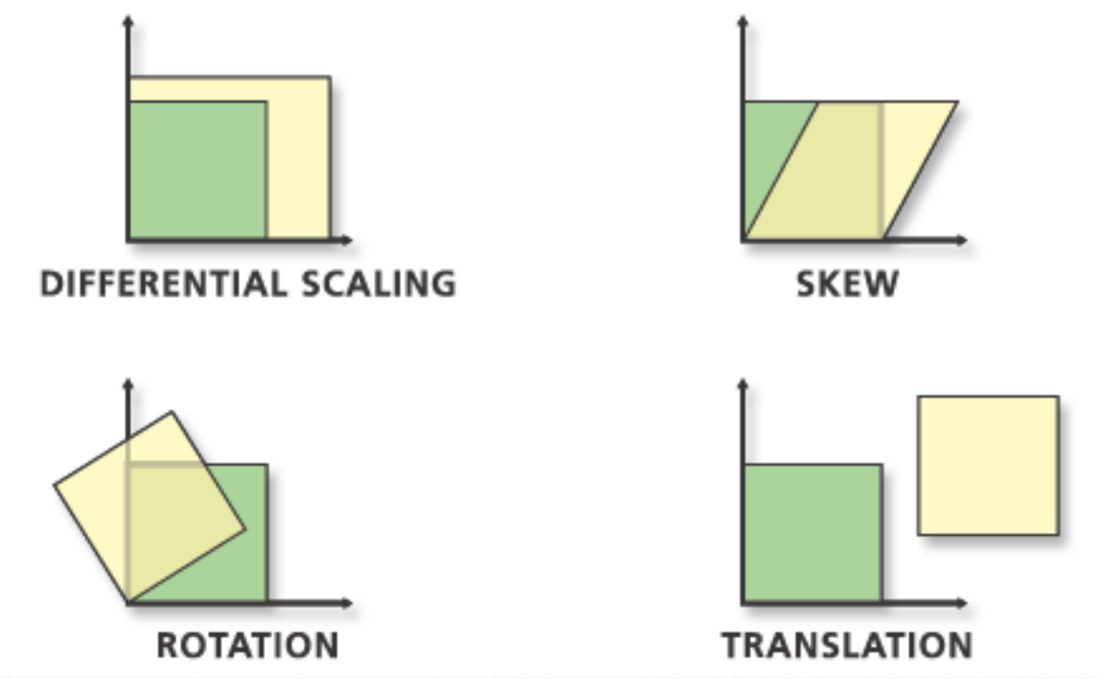
Dr. Sarah Abraham
University of Texas at Austin
Computer Science Department



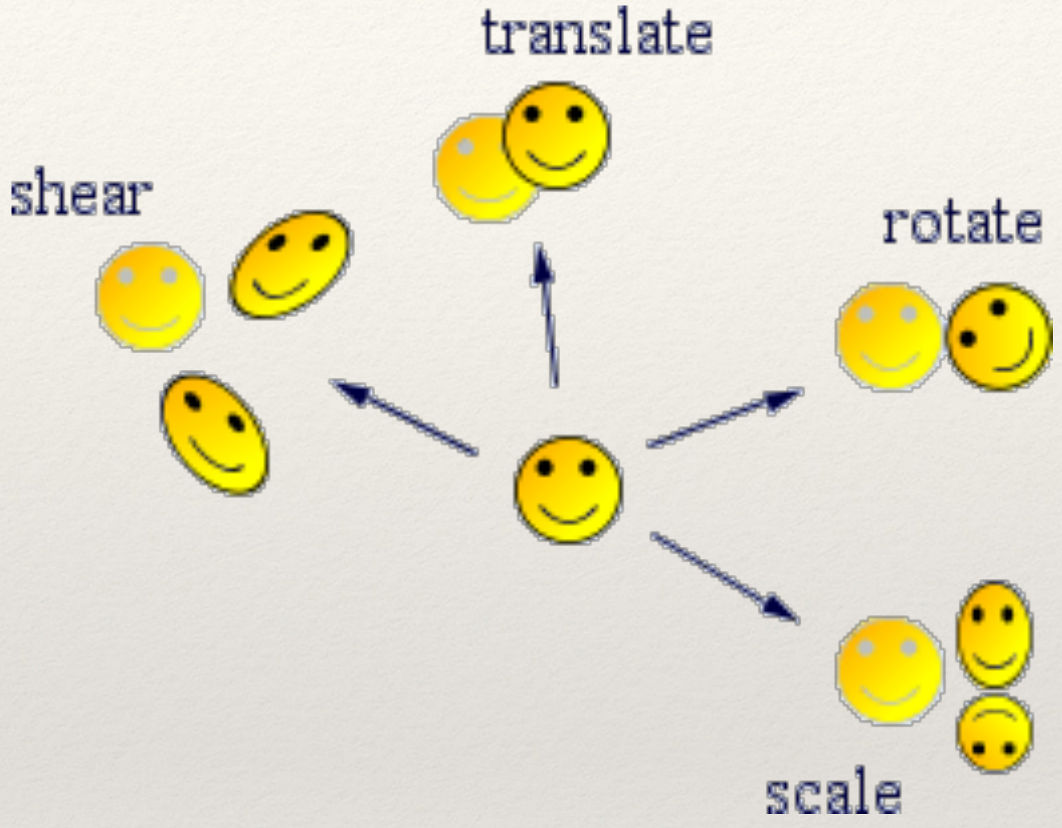
Transforms

Elements of Graphics
CS324e

Shapes and Hierarchies

- Shapes form complex structures via hierarchies
- Hierarchies make it easier to manipulate shapes and their structures
- Animation is a high-level form of this
- But how is this process done in practice?

Transformations



(RichDoc Framework)

- Foundation of rendering in computer graphics
- Allows for manipulation of objects within a scene

Point Representation

- Represent a single vertex point **p** as a vector:

$$\begin{bmatrix} x \\ y \end{bmatrix}$$

- Represent a 2-D transformation with matrix **M**:

$$\mathbf{M} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

- Multiply **p** by **M** to apply the transformation:

$$\mathbf{p}' = \mathbf{M}\mathbf{p}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Multiplication

- How do we multiply?

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$x' = ax + by$$
$$y' = cx + dy$$

- What if we multiply by the identity matrix?

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

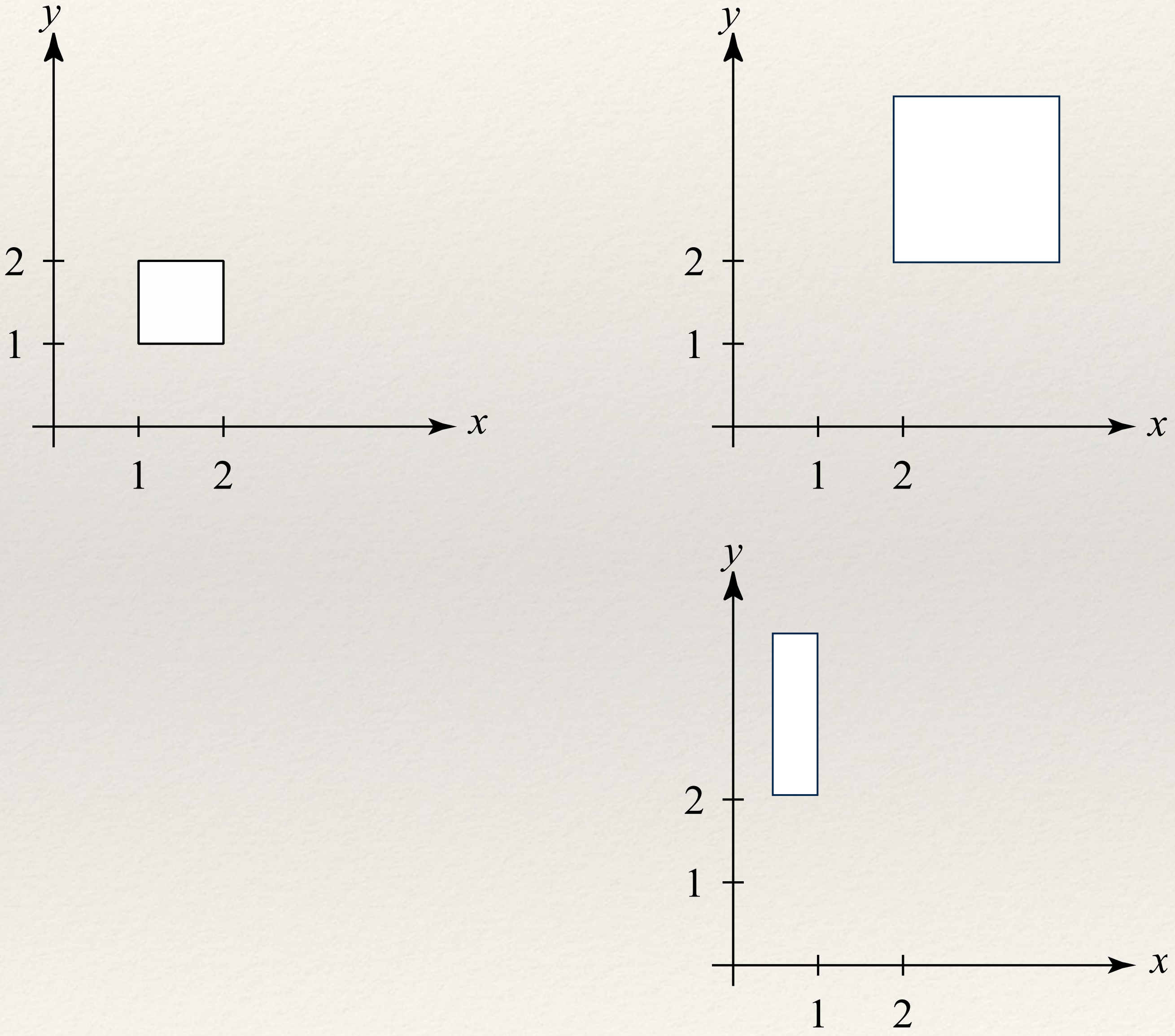
$$x' = ax$$
$$y' = dy$$

Scaling

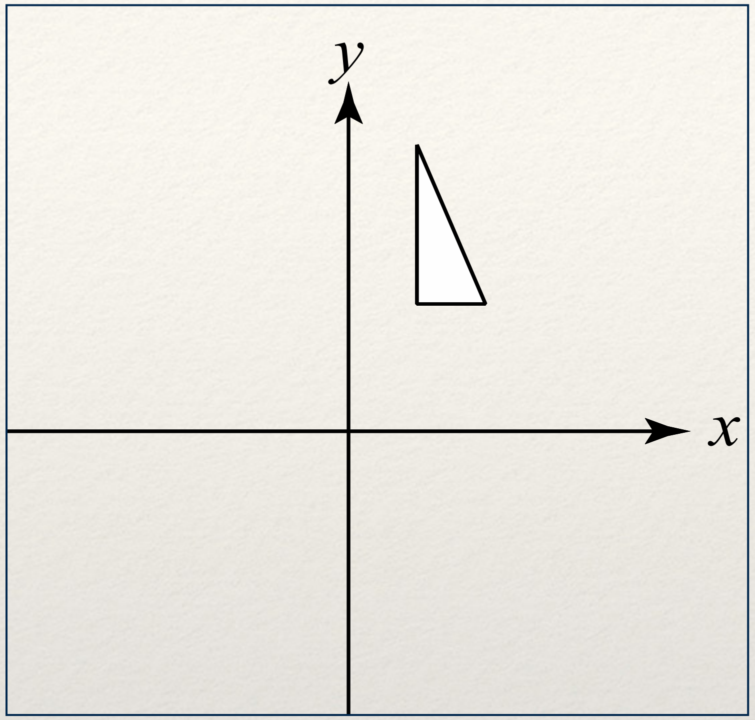
- What happens with these two matrices applied on this square?

$$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

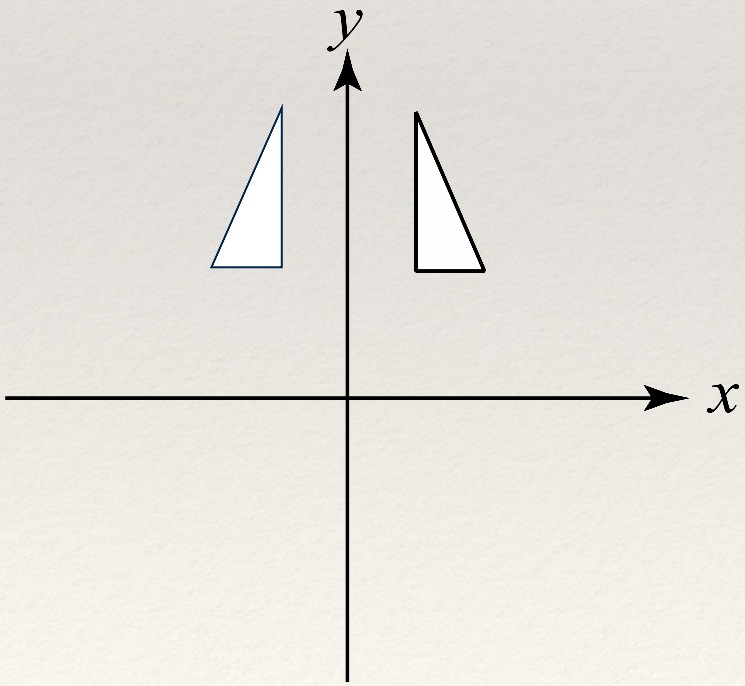
$$\begin{bmatrix} 1/2 & 0 \\ 0 & 2 \end{bmatrix}$$

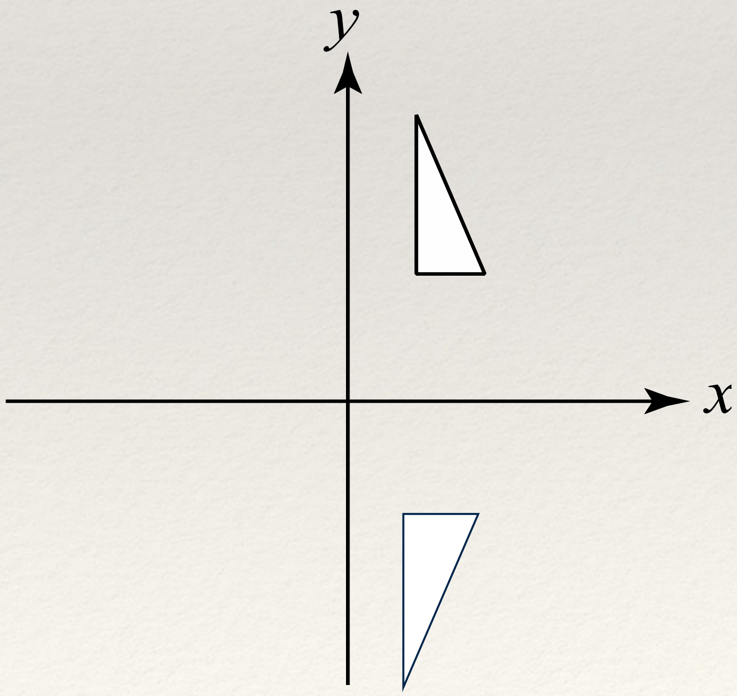


Reflection



$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$





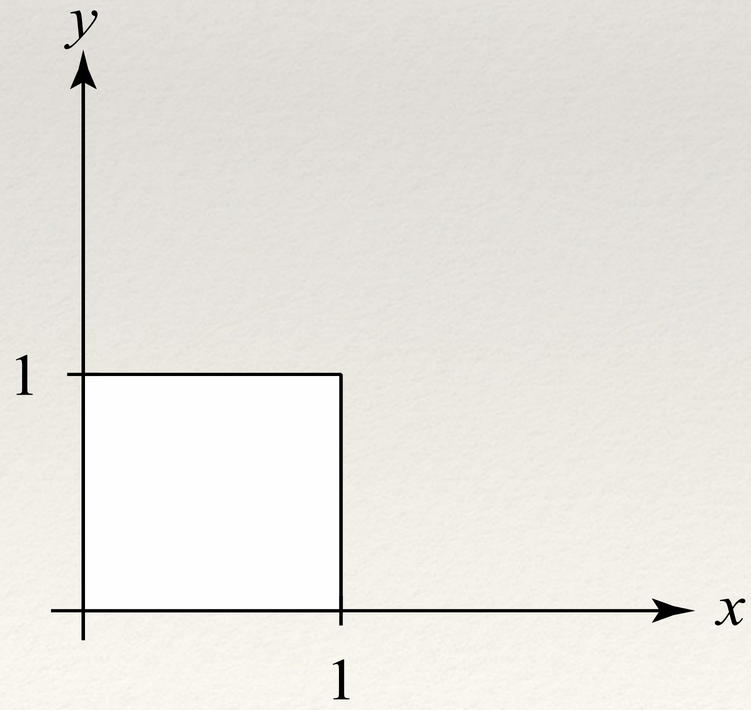
$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

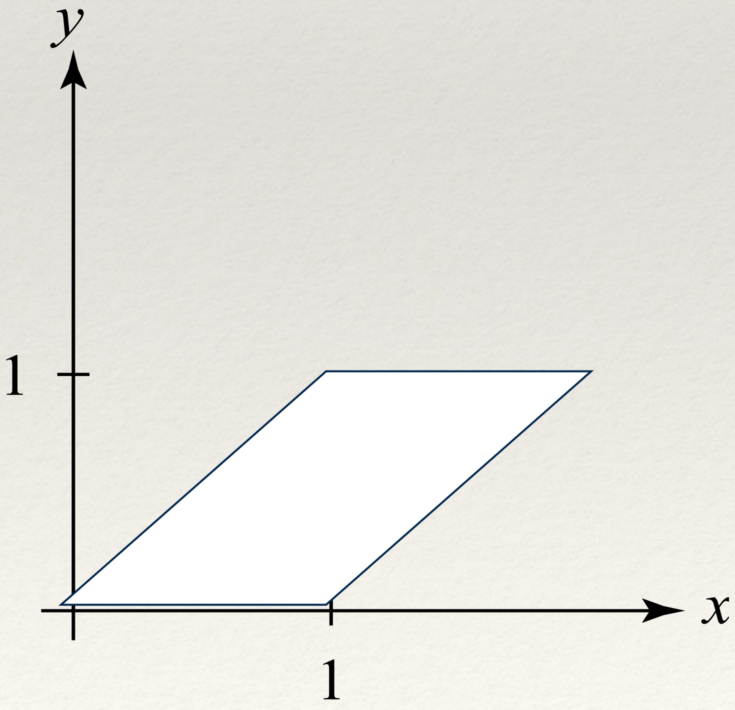
Shear

- What if we set **a** = **d** = 1 but then modify **b**?

$$\begin{bmatrix} 1 & b \\ 0 & 1 \end{bmatrix}$$

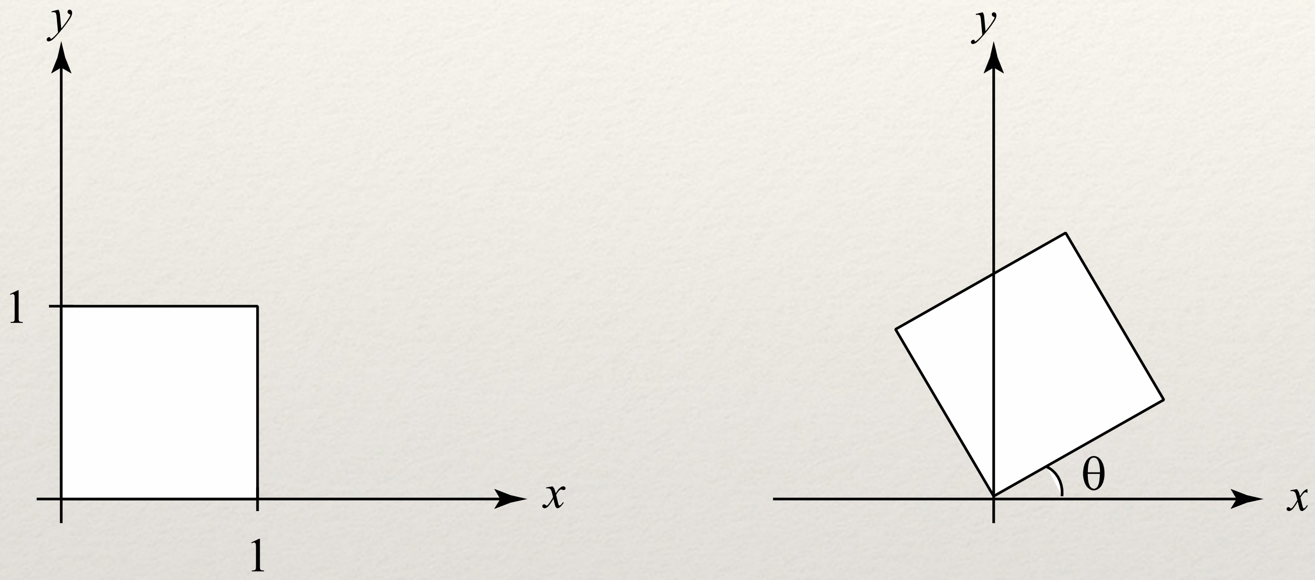
$$x' = x + by$$
$$y' = y$$





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

Rotation



$$\begin{bmatrix}1\\0\end{bmatrix} \rightarrow \begin{bmatrix}\cos(\theta)\\ \sin(\theta)\end{bmatrix}$$

$$\begin{bmatrix}0\\1\end{bmatrix} \rightarrow \begin{bmatrix}-\sin(\theta)\\ \cos(\theta)\end{bmatrix}$$

$$M_R = R(\theta) = \begin{bmatrix}\cos(\theta) & -\sin(\theta)\\ \sin(\theta) & \cos(\theta)\end{bmatrix}$$

Linear Transformations

- All of these transformations are considered linear transformations
    - Scaling
    - Reflection
    - Shearing
    - Rotation
- What’s missing?

Affine Transformations

- We want objects to move, or **translate**, through space
- Linear space (for linear transformations) has no notion of "position"
- Therefore affine space takes linear space and adds an "origin" point

Homogenous Coordinates

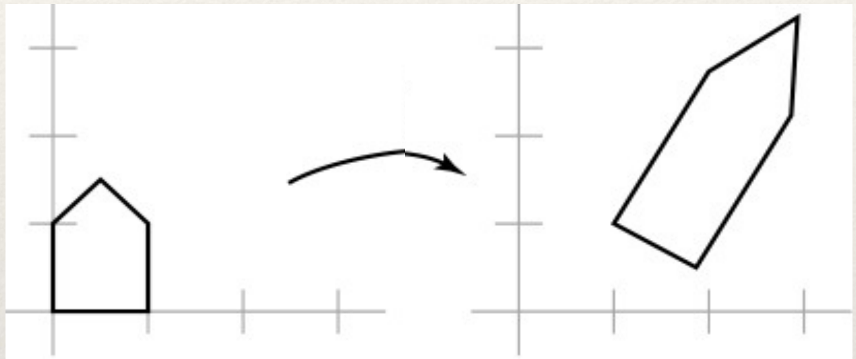
- Give every point a third component:

$$\mathbf{p}' = \mathbf{M}\mathbf{p}$$

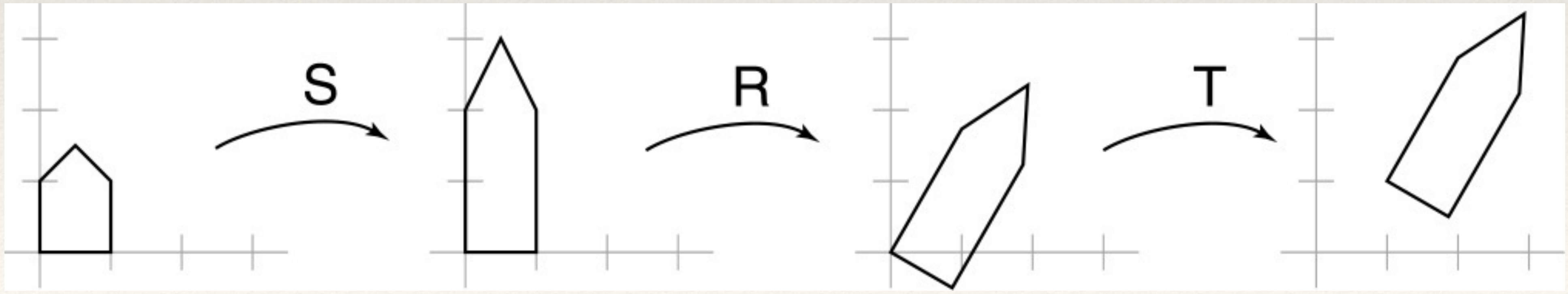
$$= \begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Series of Transformations

- We can combine a sequence of transformations into one matrix to transform the geometric instance:



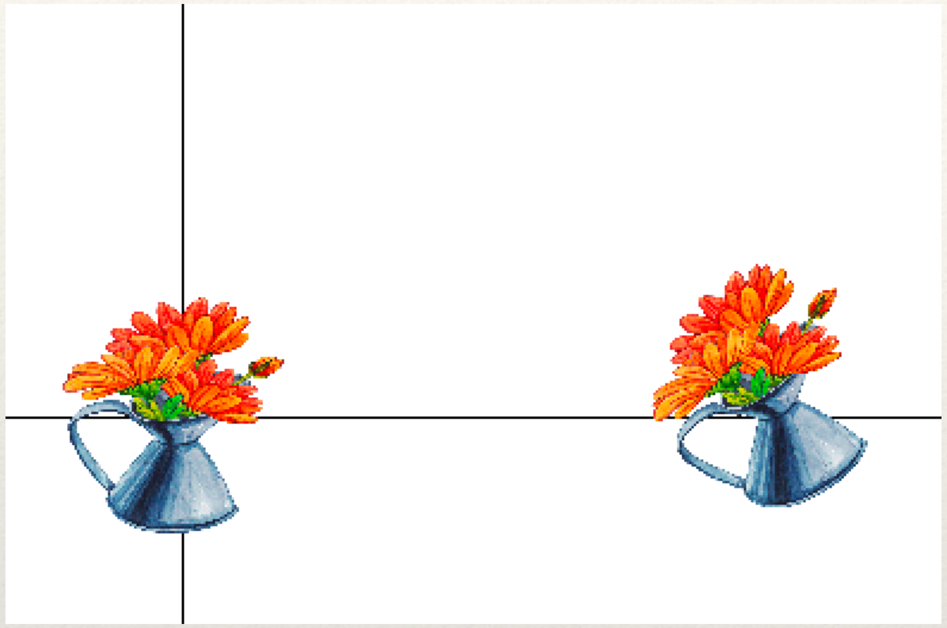
- But we can also think of this transformation as a series of simpler transformations:



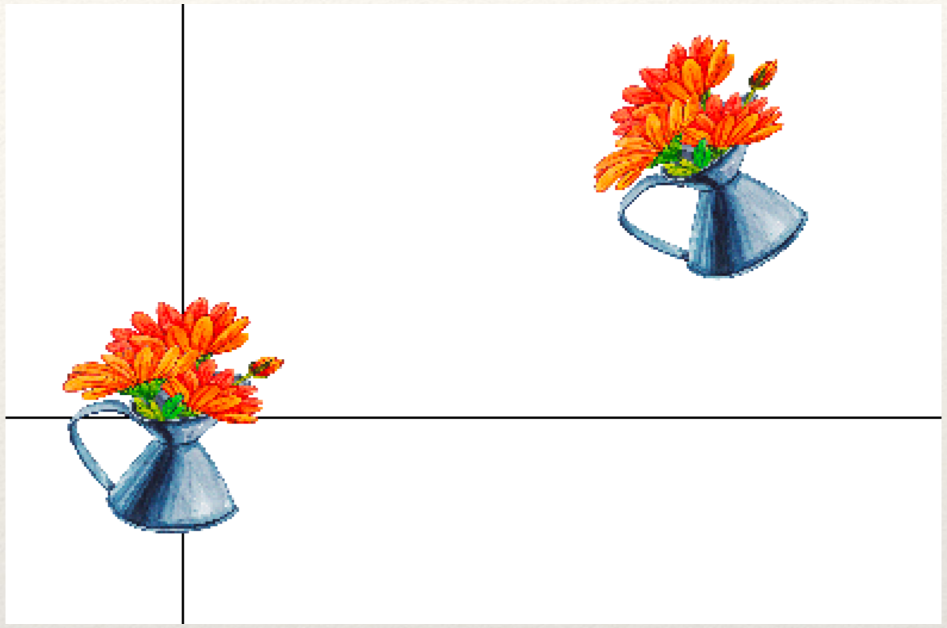
Transformation Order

- Transformation order matters!
- Mathematical reason: transformation matrices do not commute under matrix multiplication
- Intuitive reason: what happens when we rotate then translate versus translate then rotate?

Transformation Order



Translate -> Rotate



Rotate -> Translate

- To rotate an object around itself:
  - Scale -> Translate -> Rotate (applied to the canvas)
- To rotate around a specific point:
  - Scale -> Rotate -> Translate (applied to the canvas)

Transformations in Processing

- `translate()`, `rotate()` and `scale()`
- `translate(x, y)` moves the objects by an (x, y) offset
- `rotate(θ)` rotates the objects by Ө radians
- `scale(p)` scales the objects by p percent

PushMatrix() and PopMatrix()

- `pushMatrix()` records the current state of the transformation matrix
- popMatrix() returns the transformation matrix to the previously recorded state
- These functions allow us to manipulate objects at **different levels of hierarchy**
- pushes and pops can be nested

PushMatrix/PopMatrix Example

- Move Object2 **relative** to Object1

```
pushMatrix();
translate(x, y);
displayObject1();
translate(offsetX, offsetY);
displayObject2();
popMatrix();
```

Hands-on: Using Transformations

- Today's activities:
    1. Translate a sketch's screen origin to the center of the screen
    2. Draw a rectangle at the center of the screen
    3. Draw another rectangle after scaling then translating the sketch's screen origin
    4. Draw another rectangle after rotating then translating the sketch's screen origin
    5. Draw another rectangle after translate then rotating the sketch's screen origin
    6. Make a hierarchy where one objects moves relative to another