Introduction to 2D

Going to start out with 2D operations, so we can manually write all of the code to develop a deeper understanding of the rendering techniques. Once we have experience with 2D, the transition to 3D will be straightforward.

Three types of affine transformations we'll focus on:

- Translation (moving)
- Scaling (sizing)
- Rotation

Affine Transformations:

- Preserves points, straight lines, & planes.
- Sets of parallel lines remain parallel, also preserves ratio of distances on a straight line.
- Angle and distances are not necessarily preserved.

$f : X \rightarrow Y$

$\vec{y} = A\vec{x} + \vec{b}$

where A is the "linear map" or transformation matrix

and $\vec{b}$ is the "translation"

Matrix Math Reminder

$$A=\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad B=\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

Addition

$$C=A+B=\begin{bmatrix} (a_{11}+b_{11}) & (a_{12}+b_{12}) \\ (a_{21}+b_{21}) & (a_{22}+b_{22}) \end{bmatrix}$$

Multiplication

- The number of columns in the 1st matrix must equal the number of rows in the 2nd matrix.
- The result will have the same number of rows as the 1st matrix and the same number of columns as the 2nd matrix.

$$C=A\cdot B=\begin{bmatrix} (a_{11}\cdot b_{11}+a_{12}\cdot b_{21}) & (a_{11}\cdot b_{12}+a_{12}\cdot b_{22}) \\ (a_{21}\cdot b_{11}+a_{22}\cdot b_{21}) & (a_{21}\cdot b_{12}+a_{22}\cdot b_{22}) \end{bmatrix}$$

General form...

$C_{ij}=\sum_{k=1}^{m} a_{ik}\cdot b_{kj}$ where m is the size of the matrix

Remember that multiplication is not commutative.

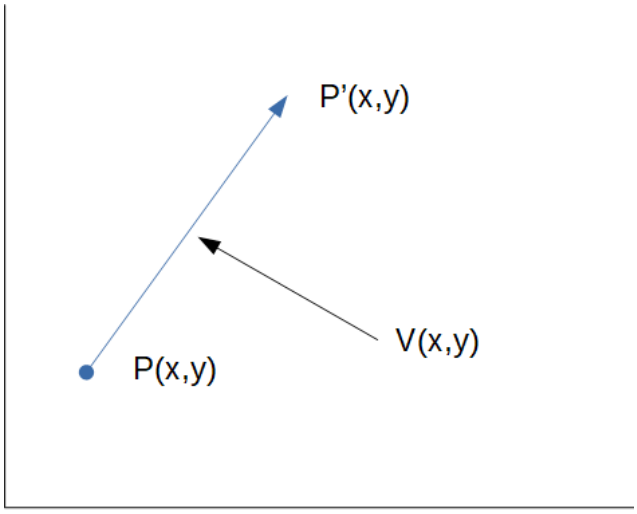
$A\cdot B\,!=B\cdot A$ The result might be equal, but isn't defined to be true

Also note that:

$$A\cdot B\cdot C=(A\cdot B)\cdot C=A\cdot(B\cdot C)$$

Translation

We translate each point, $P(x,y)$, by a vector $V(x,y)$, resulting in $P'(x,y)$



Building a series of equations...

$$V(x,y)=P'(x,y)-P(x,y)$$

$$dx=x'-x$$

$$dy=y'-y$$

$$P'(x)=P(x)+dx$$

$$P'(y)=P(y)+dy$$

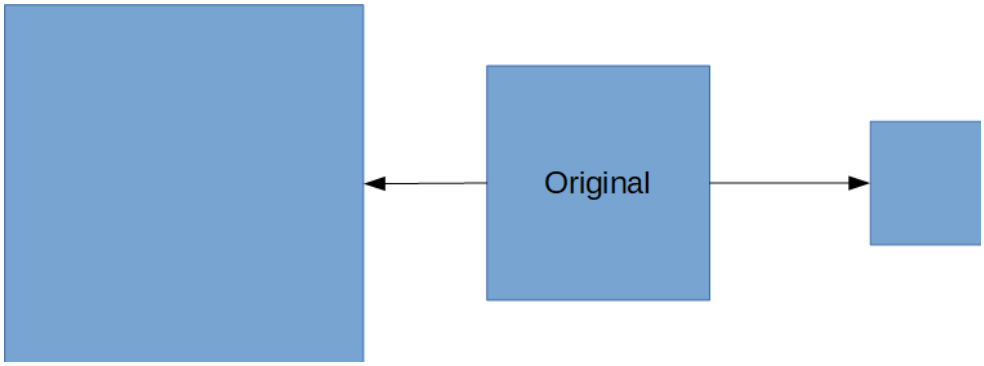
Putting it into matrix form…

$$\begin{bmatrix}x'\\y'\end{bmatrix}=\begin{bmatrix}x\\y\end{bmatrix}+\begin{bmatrix}dx\\dy\end{bmatrix} \quad \text{or} \quad \begin{bmatrix}x'\\y'\end{bmatrix}=\begin{bmatrix}dx\\dy\end{bmatrix}+\begin{bmatrix}x\\y\end{bmatrix}$$

$T=\begin{bmatrix}dx\\dy\end{bmatrix}$ giving us $P'=T+P$

Scaling

We scale each point, $P(x,y)$, by a scaling factor $S(x,y)$, resulting in $P'(x,y)$



Scaling is done on a point-by-point basis. Each point is scaled, about its center, which results in scaling of the whole object.

In matrix form

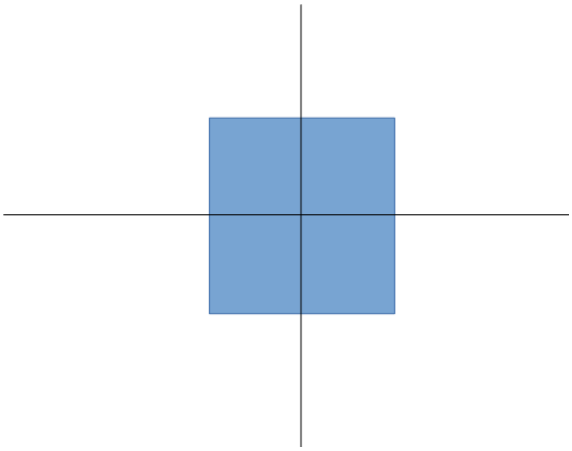
$$S=\begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \quad \text{...then...} \quad \begin{bmatrix} P'_x \\ P'_y \end{bmatrix}=\begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}\cdot\begin{bmatrix} P_x \\ P_y \end{bmatrix}$$

Scaling is performed relative to the origin.

if S(x) or S(y) > 1 ; distance from origin increased

if S(x) or S(y) < 1 ; distance to origin decreased

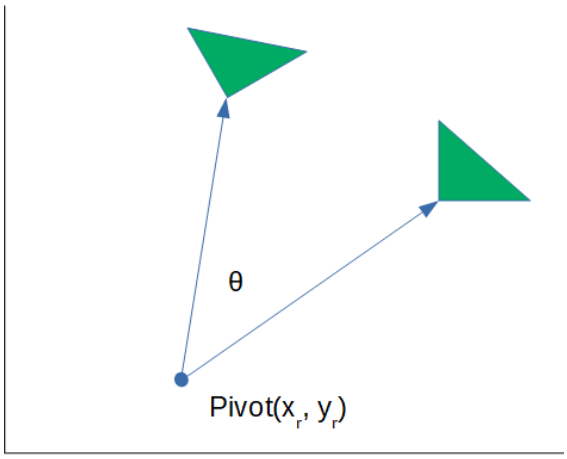
where S(x) and S(y) > 0



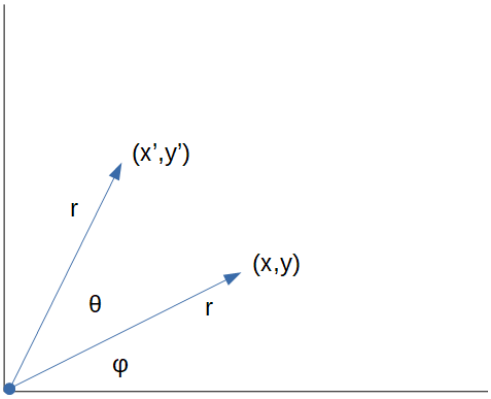
Therefore, each point must first be translated by the object center to the axis origin, scaled, then translated back to the object center.

Rotation

- Specify an angle of rotation theta
- Specify a point of rotation (pivot point)



Let's start by considering rotation about the origin:



$$x = r\cos(\Phi)$$

$$y = r\sin(\Phi)$$

$$x' = r\cos(\Phi+\Theta) = r\cos(\Phi)\cos(\Theta) - r\sin(\Phi)\sin(\Theta)$$

$$y' = r\sin(\Phi+\Theta) = r\cos(\Phi)\sin(\Theta) + r\sin(\Phi)\cos(\Theta)$$

(using trig identities for sin, cos sum of two angles)

Using substitution, we get…

$$x' = x\cos(\Theta) - y\sin(\Theta)$$

$$y' = x\sin(\Theta) + y\cos(\Theta)$$

Turning into matrix form…

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\Theta) & -\sin(\Theta) \\ \sin(\Theta) & \cos(\Theta) \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

$$P' = R \cdot P$$

General Rotation

$$x' = x_r + (x - x_r)\cos(\Theta) - (y - y_r)\sin(\Theta)$$

$$y' = y_r + (x - x_r)\sin(\Theta) + (y - y_r)\cos(\Theta)$$

Then create a matrix...

But we won't do it this way. Eventually we'll create a composite matrix.

General Rotation (again)

- Remember, rotation is about the origin.
- Just like translation and scaling, rotate point-by-point.
- To rotate an object, apply the same rotation to each point of the object.

Two approaches to handling rotation

Approach #1 : Global Coordinates

- Define all object points about some center (x,y)
- Translate each point by the center (negative center)
- Rotate the point
- Translate back by the center

Approach #2 : Local Coordinates

- Define all the object points about (0, 0)
- Apply rotation to each point
- When drawing, translate each point by the center first, then draw. This is called a *local to world coordinate transformation*. We'll come back and talk about this more when we work in 3D and use WebGL.

Homogeneous Coordinates

Consider that all three transformations are of the form:

$$P' = M_1 \cdot P + M_2$$

where:

Translation: $M_1 = I ; M_2 = Translation$

Scaling: $M_1 = S ; M2 = 0$

Rotation: $M_1 = R(\Theta) ; M_2 = 0$

Also consider a homogeneous triple:

Cartesian: (x, y)

Homogeneous: (x_h, y_h, h)

where

$$x = \frac{x_h}{h} \quad \text{and} \quad y = \frac{y_y}{h}$$

Therefore, *h* becomes a kind of scaling factor, therefore we choose *h = 1*

Given this, we can then make homogeneous matrices, getting rid of M_2.

Translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Scaling

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

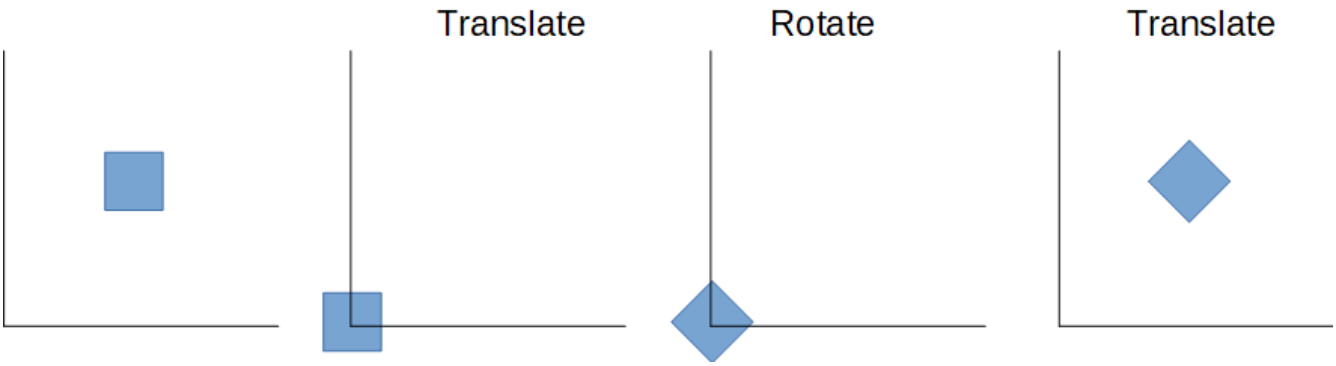
Rotation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\Theta) & -\sin(\Theta) & 0 \\ \sin(\Theta) & \cos(\Theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

What is so great about creating the homogeneous matrices? It is kind of interesting, but so what? This is what…

Composite Matrices – General Pivot Rotation

Remember that rotation requires three steps: Translate, Rotate, Translate.



This requires three separate matrix multiplication operations. What if we could do these all in a single matrix operation, wouldn't that be cool? We can, like this…

$P' = T_2 \cdot R \cdot T_1 \cdot P$ where T_1 is translation by negative center, T_2 is translation by positive center

$$M_c = T_2 \cdot R \cdot T_1$$

$$P' = M_c \cdot P$$

$$M_c = \begin{bmatrix} 1 & 0 & T_{x_2} \\ 0 & 1 & T_{y_2} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\Theta) & -\sin(\Theta) & 0 \\ \sin(\Theta) & \cos(\Theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & T_{x_1} \\ 0 & 1 & T_{y_1} \\ 0 & 0 & 1 \end{bmatrix}$$

note that $T_{x_2} = -T_{x_1}$ and $T_{y_2} = -T_{y_1}$

Using that, multiplying and simplifying, we get a 2D general pivot rotation matrix.

$$M_c = \begin{bmatrix} \cos(\Theta) & -\sin(\Theta) & T_{x_1}(\cos(\Theta)-1) - T_{y_1}\sin(\Theta) \\ \sin(\Theta) & \cos(\Theta) & T_{x_1}\sin(\Theta) + T_{y_1}(\cos(\Theta)-1) \\ 0 & 0 & 1 \end{bmatrix}$$

Now, rather than three matrix operations, can perform a single matrix operation and gain a lot of efficiency!

2D Reflection

X-Axis

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

Y-Axis

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

Origin

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

y = x

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

y = -x

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

About line: $y = mx + b$

1. Translate line to origin

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

2. Rotate line into the x-axis (rotate clockwise)

m = sin/cos

$$\sin(\Theta) = \frac{m}{\sqrt{(1+m^2)}} \qquad \cos(\Theta) = \frac{\sin(\Theta)}{m}$$

$$\begin{bmatrix} \cos(\Theta) & \sin(\Theta) & 0 \\ -\sin(\Theta) & \cos(\Theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3. Reflect about x-axis

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

4. Rotate back (rotate counterclockwise)

$$\begin{bmatrix} \cos(\Theta) & -\sin(\Theta) & 0 \\ \sin(\Theta) & \cos(\Theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

5. Translate back

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

$$P' = T_2 \cdot R_\Theta \cdot R_x \cdot R_{-\Theta} \cdot T_1 \cdot P$$