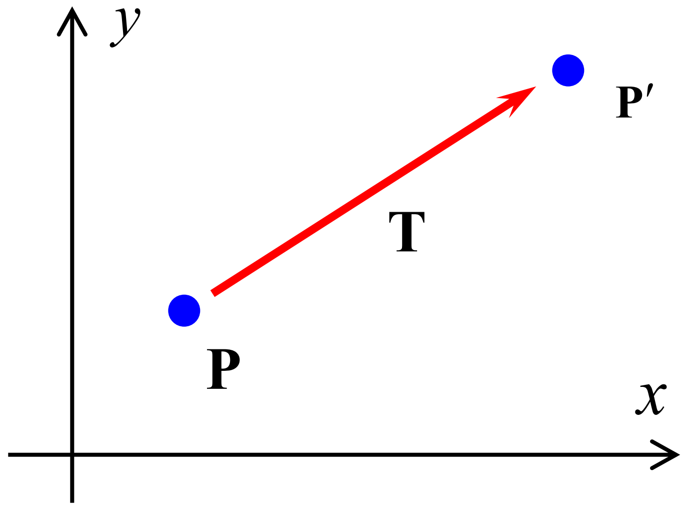
Geometric Transformations

Unit-:4

2D Translation

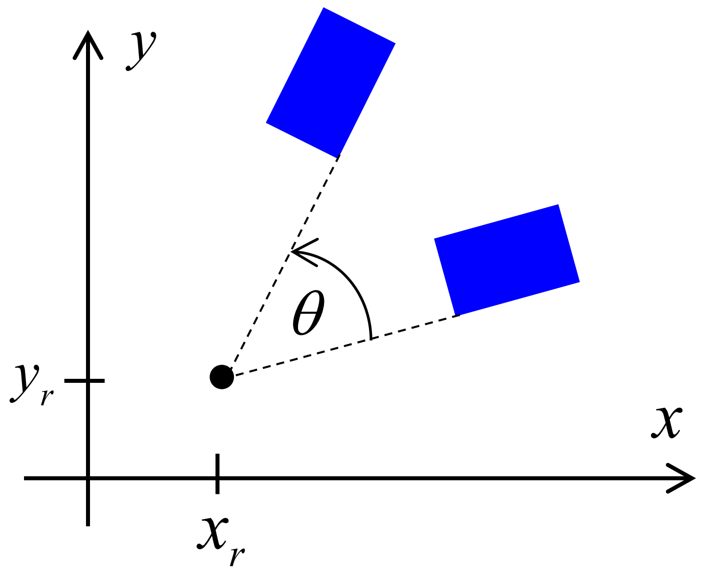


$$x' = x + t_x, \quad y' = y + t_y$$

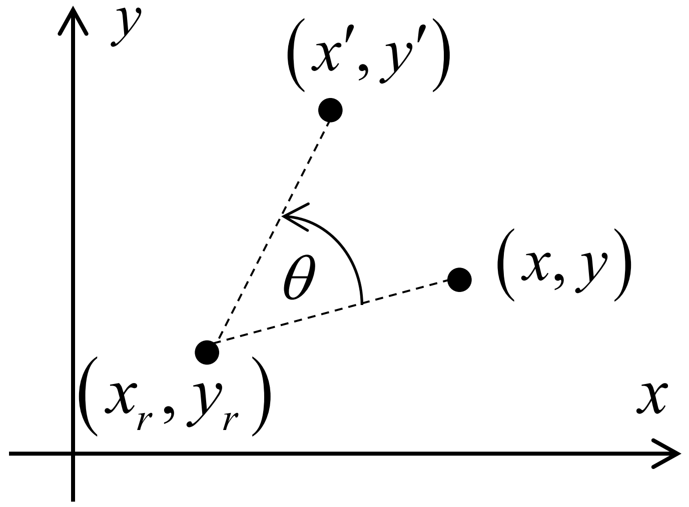
$$\mathbf{P} = \begin{bmatrix} x \\ y \end{bmatrix}, \quad \mathbf{P}' = \begin{bmatrix} x' \\ y' \end{bmatrix}, \quad \mathbf{T} = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$\mathbf{P}' = \mathbf{P} + \mathbf{T}$$

2D Rotation



Rotation in angle θ about a pivot (rotation) point (x_r, y_r).



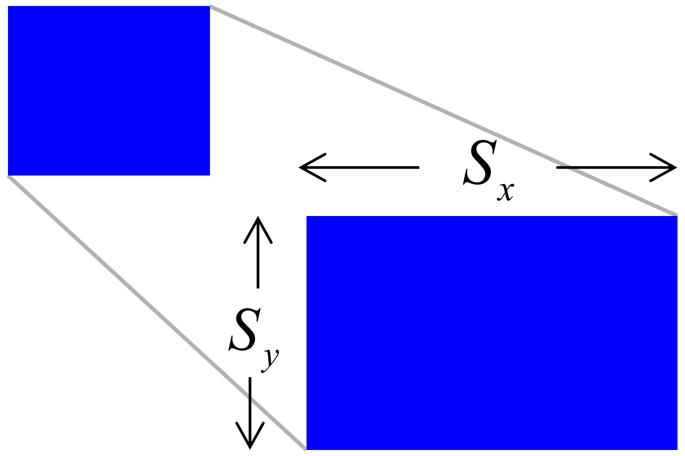
$$x' = x_r + (x - x_r)\cos\theta - (y - y_r)\sin\theta$$

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

$$\mathbf{P}' = \mathbf{P}_r + \mathbf{R}\cdot(\mathbf{P} - \mathbf{P}_r)$$

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

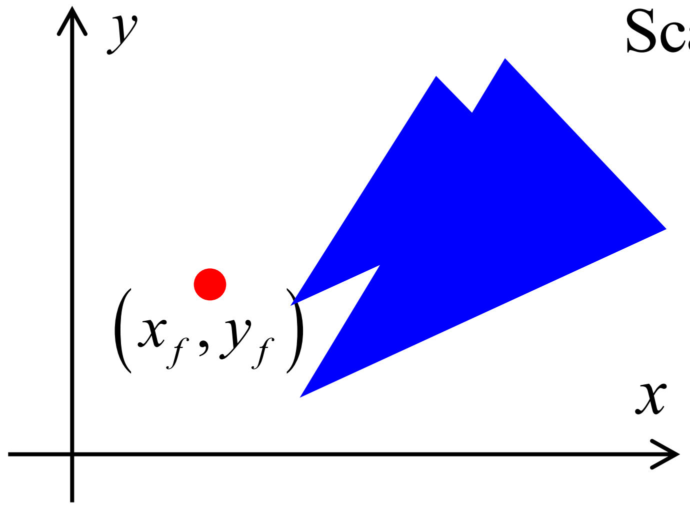
2D Scaling



$$x' = x \cdot s_x, \quad y' = y \cdot s_y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\mathbf{P}' = \mathbf{S} \cdot \mathbf{P}$$



Scaling about a fixed point (x_f, y_f)

$$x' = x \cdot s_x + x_f (1 - s_x)$$

$$y' = y \cdot s_y + y_f (1 - s_y)$$

$$\mathbf{P}' = \mathbf{P} \cdot \mathbf{S} + \mathbf{P}_f \cdot (\mathbf{1} - \mathbf{S})$$

Homogeneous Coordinates

Rotate and then displace a point $\mathbf{P}$: $\mathbf{P}' = \mathbf{M}_1 \cdot \mathbf{P} + \mathbf{M}_2$

$\mathbf{M}_1$: 2×2 rotation matrix. $\mathbf{M}_2$: 2×1 displacement vector.

Displacement is unfortunately a non linear operation.

Make displacement linear with **Homoheneous Coordinates**.

$(x, y) \Rightarrow (x, y, 1)$. Transformations turn into 3×3 matrices.

Very big advantage. All transformations are concatenated by matrix multiplication.

2D Translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \mathbf{P}' = \mathbf{T}\left(t_x, t_y\right) \cdot \mathbf{P}$$

2D Rotation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \mathbf{P}' = \mathbf{R}\left(\theta\right) \cdot \mathbf{P}$$

2D Scaling

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \mathbf{P}' = \mathbf{S}\left(S_x, S_y\right) \cdot \mathbf{P}$$

Inverse transformations:

$$\mathbf{T}^{-1} = \begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{bmatrix}, \ \mathbf{R}^{-1} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \ \mathbf{S}^{-1} = \begin{bmatrix} 1/S_x & 0 & 0 \\ 0 & 1/S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Composite transformations: $\mathbf{P}' = \mathbf{M}_2\left(\mathbf{M}_1 \cdot \mathbf{P}\right) = \left(\mathbf{M}_2 \cdot \mathbf{M}_1\right) \cdot \mathbf{P} = \mathbf{M} \cdot \mathbf{P}$

$$\mathbf{P}' = \mathbf{T}\left(t_{2x}, t_{2y}\right)\left\{\mathbf{T}\left(t_{1x}, t_{1y}\right) \cdot \mathbf{P}\right\} = \left\{\mathbf{T}\left(t_{2x}, t_{2y}\right) \cdot \mathbf{T}\left(t_{1x}, t_{1y}\right)\right\} \cdot \mathbf{P}$$

Composite translations: $\begin{bmatrix} 1 & 0 & t_{2x} \\ 0 & 1 & t_{2y} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & t_{1x} \\ 0 & 1 & t_{1y} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{1x} + t_{2x} \\ 0 & 1 & t_{1y} + t_{2y} \\ 0 & 0 & 1 \end{bmatrix}$

$$\mathbf{T}\left(t_{2x}, t_{2y}\right) \cdot \mathbf{T}\left(t_{1x}, t_{1y}\right) = \mathbf{T}\left(t_{1x} + t_{2x}, t_{1y} + t_{2y}\right)$$

$$\mathbf{P}' = \mathbf{R}(\theta_2)\left\{\mathbf{R}(\theta_1)\cdot\mathbf{P}\right\} = \left\{\mathbf{R}(\theta_2)\cdot\mathbf{R}(\theta_1)\right\}\cdot\mathbf{P}$$

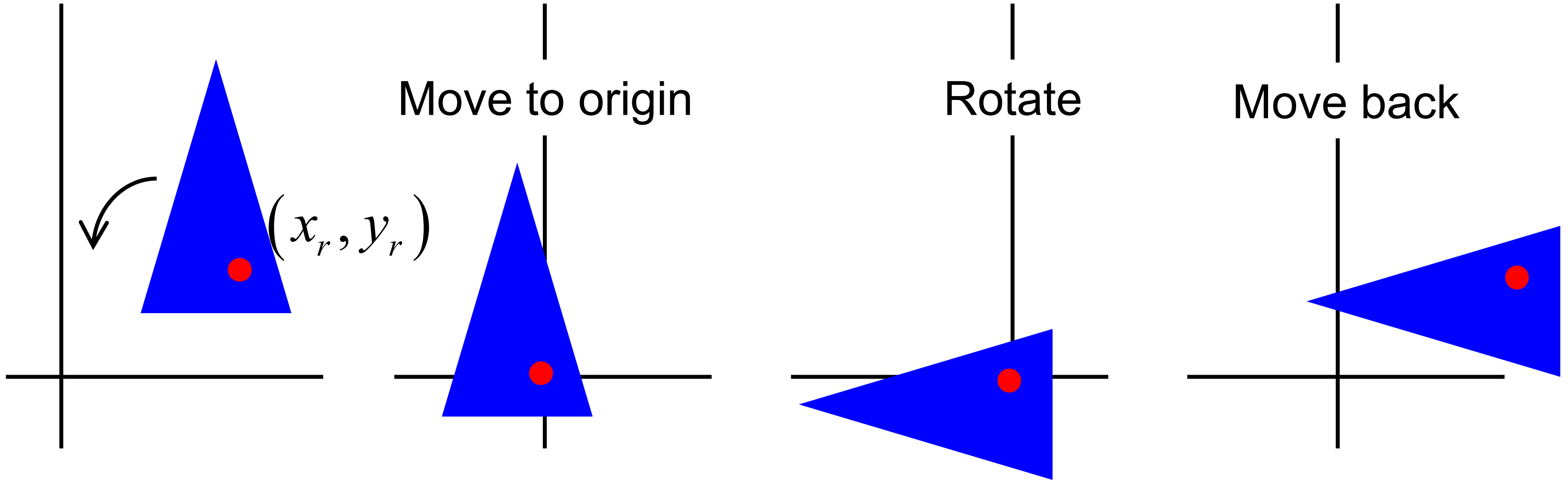
Composite Rotations: $$\mathbf{R}(\theta_2)\cdot\mathbf{R}(\theta_1) = \mathbf{R}(\theta_1+\theta_2)$$

$$\mathbf{P}' = \mathbf{R}(\theta_1+\theta_2)\cdot\mathbf{P}$$

$$\begin{bmatrix} S_{2x} & 0 & 0 \\ 0 & S_{2y} & 0 \\ 0 & 0 & 1 \end{bmatrix}\cdot\begin{bmatrix} S_{1x} & 0 & 0 \\ 0 & S_{1y} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} S_{1x}\cdot S_{2x} & 0 & 0 \\ 0 & S_{1y}\cdot S_{2y} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Composite Scaling: $$\mathbf{S}(S_{2x},S_{2y})\cdot\mathbf{S}(S_{1x},S_{1y}) = \mathbf{S}(S_{1x}\cdot S_{2x},S_{1y}\cdot S_{2y})$$

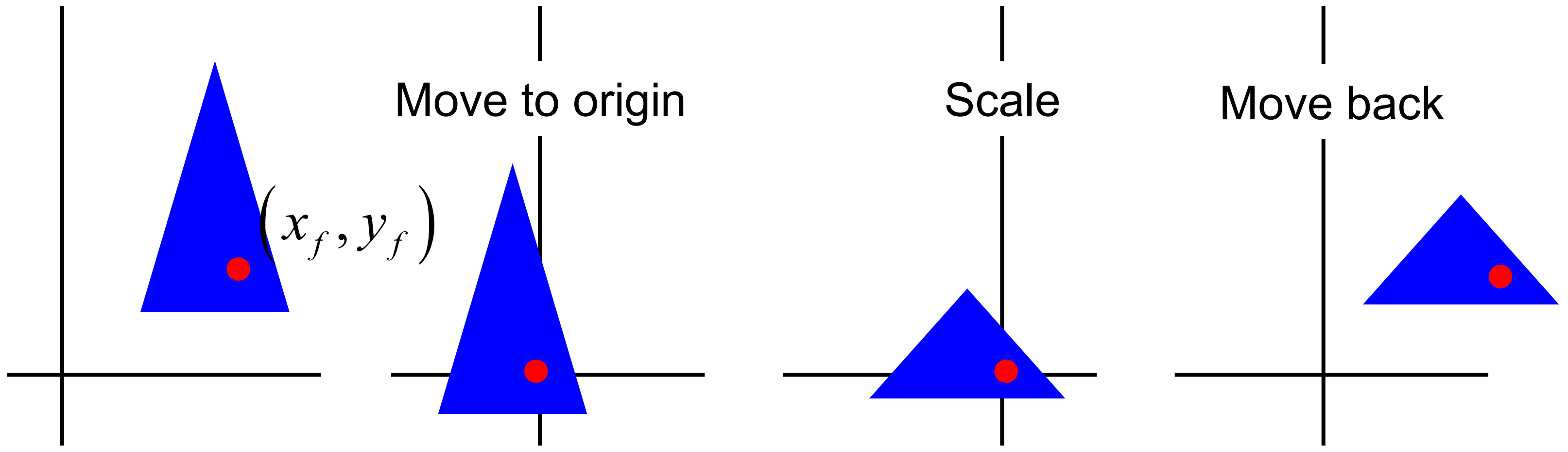
General 2D Rotation



$$\begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix} =$$

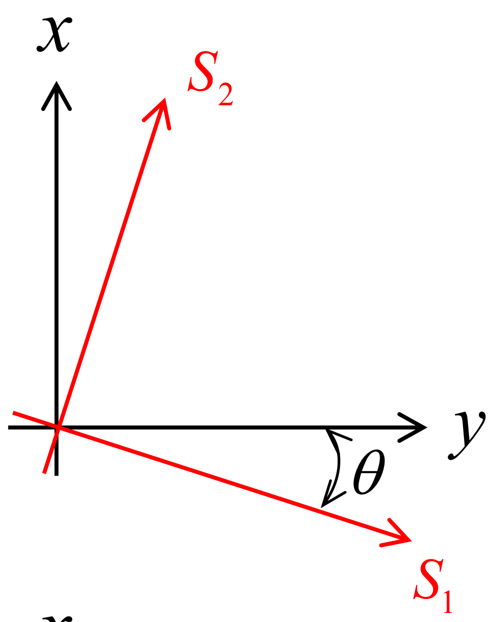
$$\begin{bmatrix} \cos\theta & -\sin\theta & x_r(1-\cos\theta)+y_r\sin\theta \\ \sin\theta & \cos\theta & y_r(1-\cos\theta)-x_r\sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

General 2D Scaling



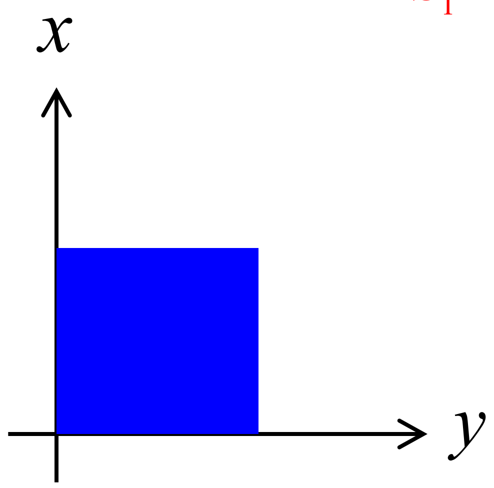
$$\begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & x_f(1-S_x) \\ 0 & S_y & y_f(1-S_y) \\ 0 & 0 & 1 \end{bmatrix}$$

2D Directional Scaling



$$\mathbf{R}^{-1}(\theta)\cdot\mathbf{S}(S_1,S_2)\cdot\mathbf{R}(\theta)=$$

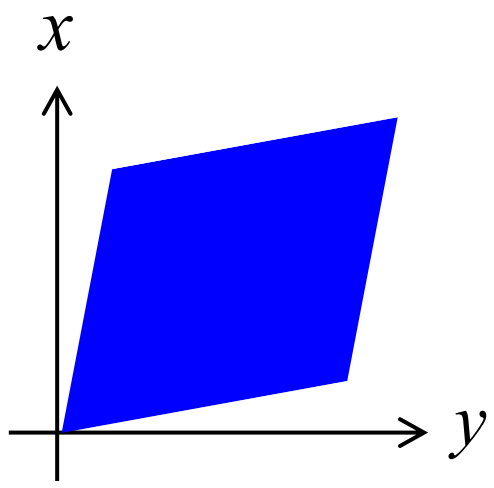
$$\begin{bmatrix} S_1\cos^2\theta+S_2\sin^2\theta & (S_2-S_1)\cos\theta\sin\theta & 0 \\ (S_2-S_1)\cos\theta\sin\theta & S_1\sin^2\theta+S_2\cos^2\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



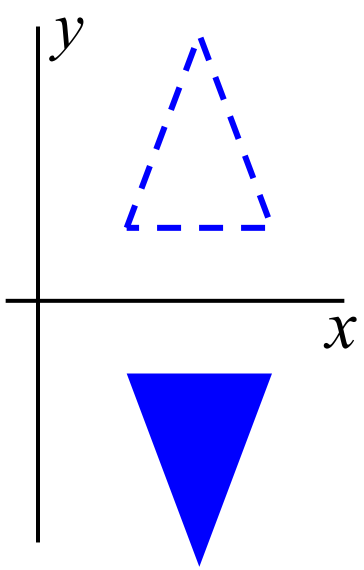
$S_1 = 1$

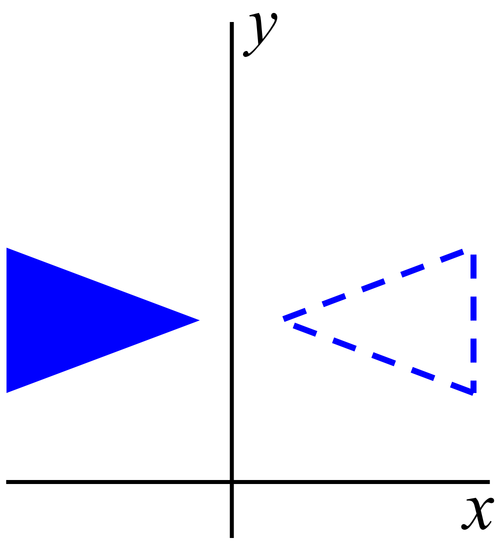
$S_2 = 2$

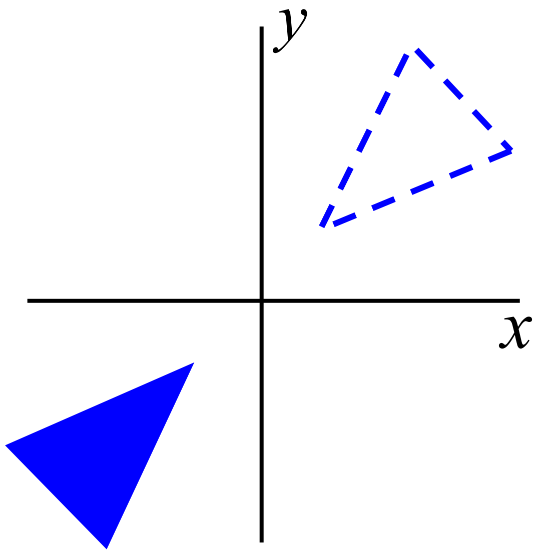
$\theta = 45^{\mathrm{O}}$



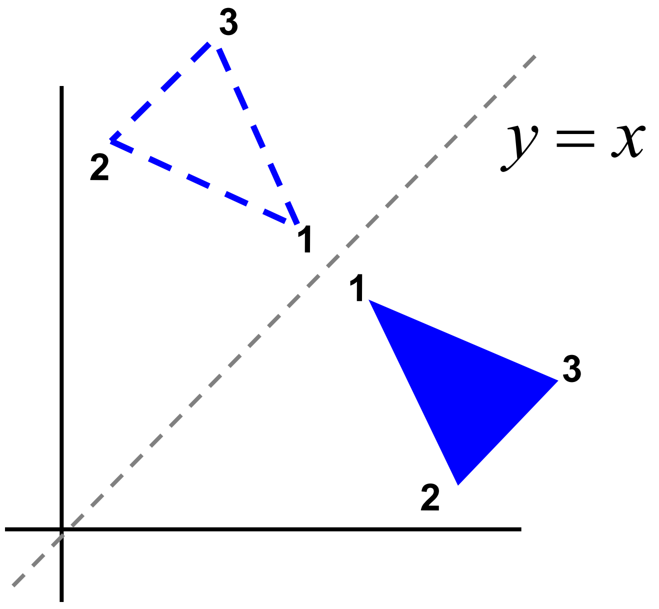
2D Reflections

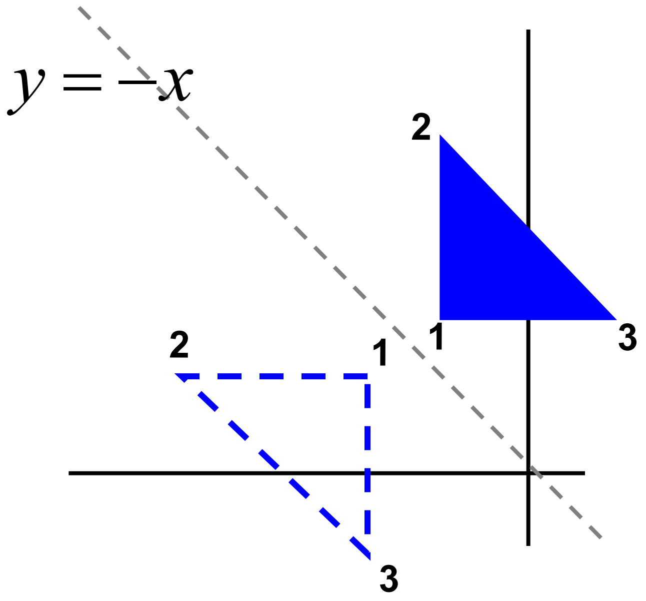






$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



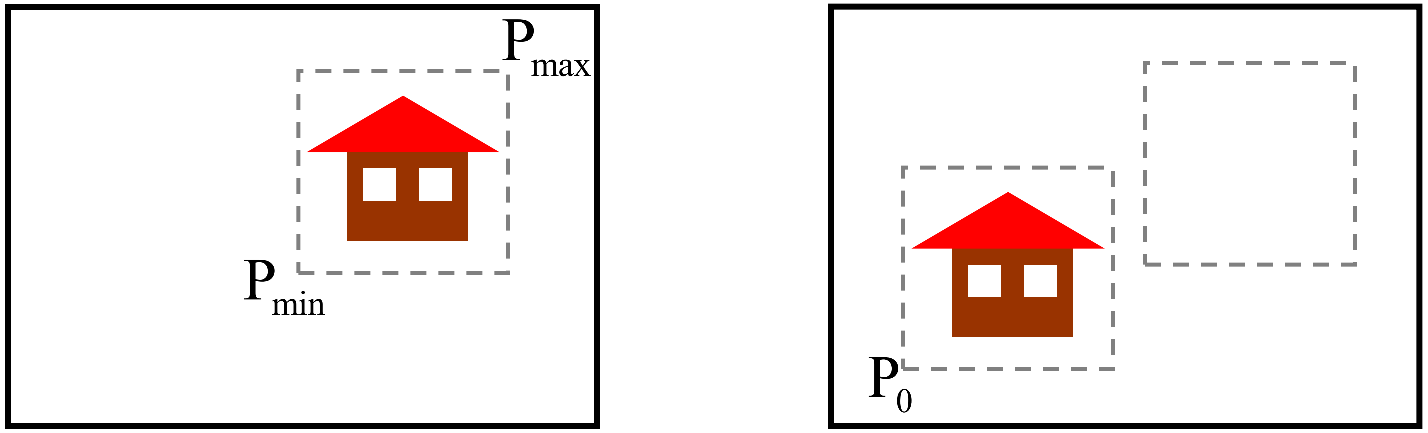


$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

Geometric Transformations by Rasterization

- The transformed shape needs to be filled.
  - A whole scan-line filling is usually in order.
- However, simple transformations can save new filling by manipulating blocks in the frame buffer.



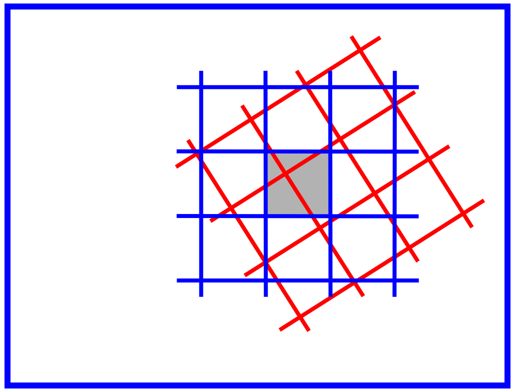
Translation:

Move block of pixels of frame buffer into new destination.

90° counterclockwise rotation

$$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\\10&11&12\end{bmatrix} \longleftrightarrow \begin{bmatrix}3&6&9&12\\2&5&8&11\\1&4&7&10\end{bmatrix} \qquad \begin{bmatrix}12&11&10\\9&8&7\\6&5&4\\3&2&1\end{bmatrix}$$

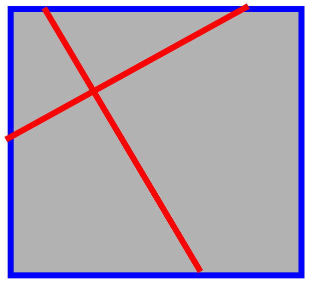
180° rotation



Rotated pixel block

Destination pixel array

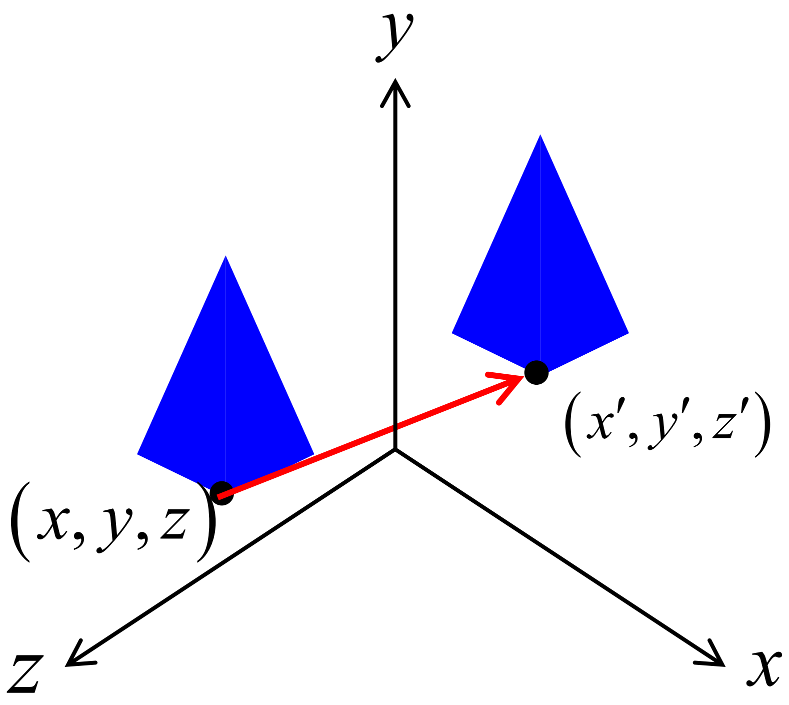
RGB of destination pixel can be determined by averaging rotated ones (as antialiasing)



3D Transformations

Very similar to 2D. Using 4x4 matrices rather than 3x3.

Translation



$$x' = x + t_x$$

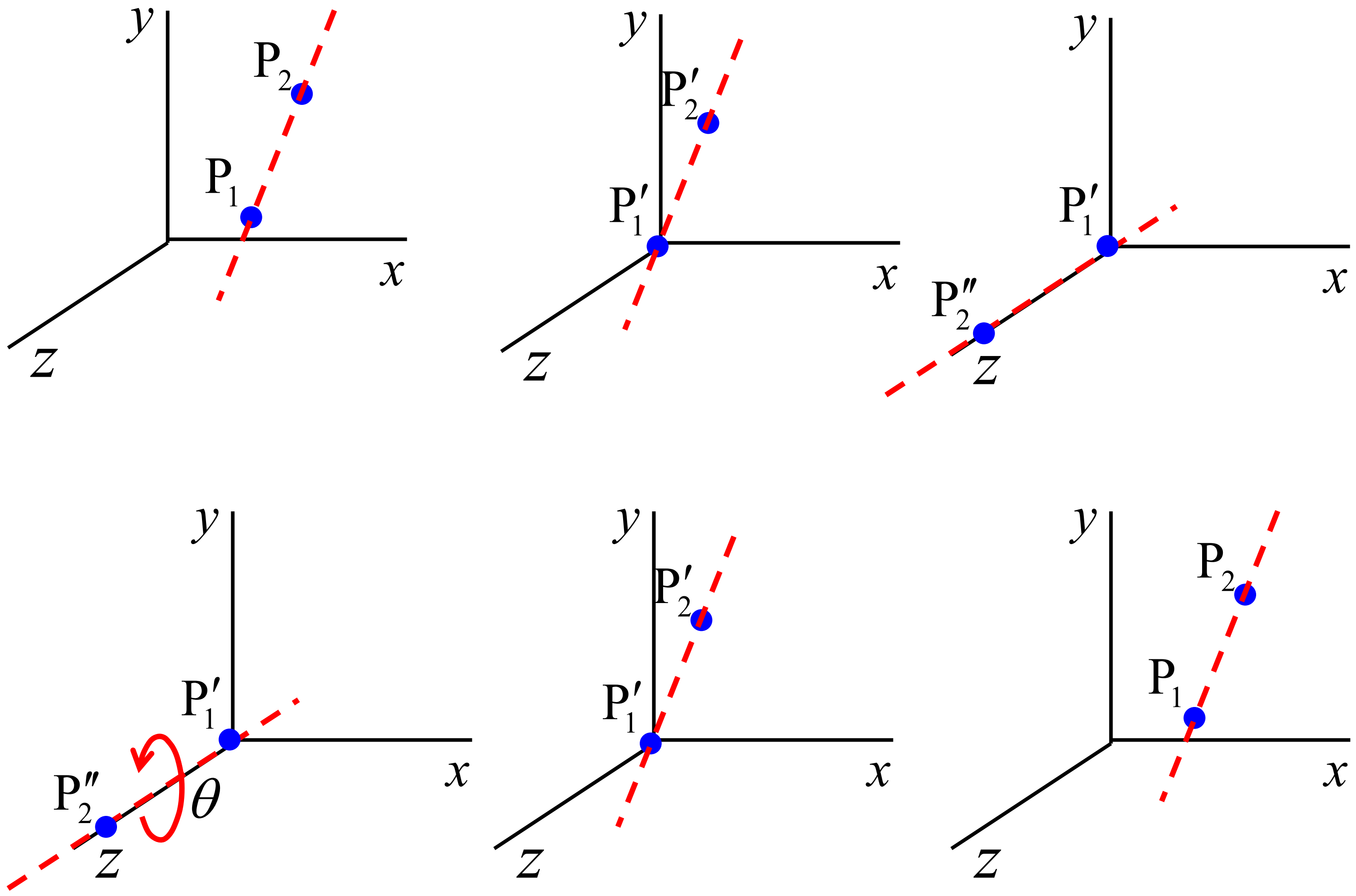
$$y' = y + t_y$$

$$z' = z + t_z$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

General 3D Rotation

1. Translate the object such that rotation axis passes through the origin.
2. Rotate the object such that rotation axis coincides with one of Cartesian axes.
3. Perform specified rotation about the Cartesian axis.
4. Apply inverse rotation to return rotation axis to original direction.
5. Apply inverse translation to return rotation axis to original position.



y
P₂
P₁
x
z
y
P′₂
P′₁
x
z
y
P′₁
P″₂
x
z
y
P′₁
P″₂
θ
x
z
y
P′₂
P′₁
x
z
y
P₂
P₁
x
z

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & -x_1 \\ 0 & 1 & 0 & -y_1 \\ 0 & 0 & 1 & -z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The vector from $\mathbf{P}_1$ to $\mathbf{P}_2$ is:

$$\mathbf{V} = \mathbf{P}_2 - \mathbf{P}_1 = \left(x_2 - x_1, y_2 - y_1, z_2 - z_1\right)$$

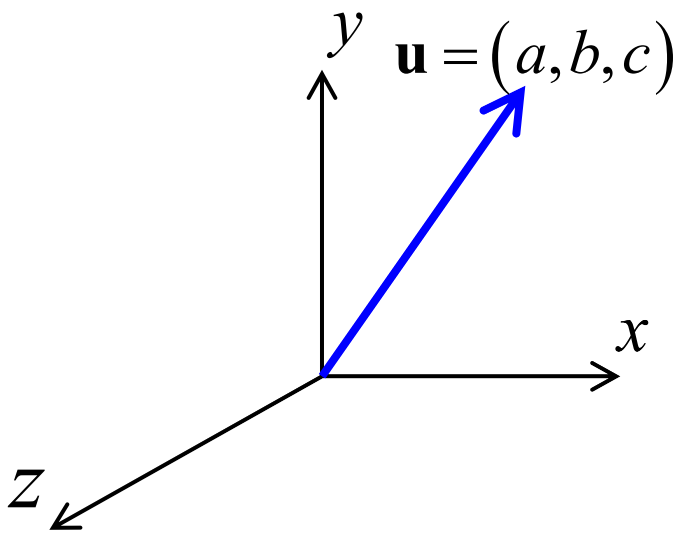
Unit rotation vector: $\mathbf{u} = \mathbf{V}/|\,\mathbf{V}\,| = \left(a, b, c\right)$

$$a = \left(x_2 - x_1\right)/|\,\mathbf{V}\,|$$

$$b = \left(y_2 - y_1\right)/|\,\mathbf{V}\,|$$

$$c = \left(z_2 - z_1\right)/|\,\mathbf{V}\,|$$

$$\sqrt{a^2 + b^2 + c^2} = 1$$



Rotating **u** to coincide with z axis

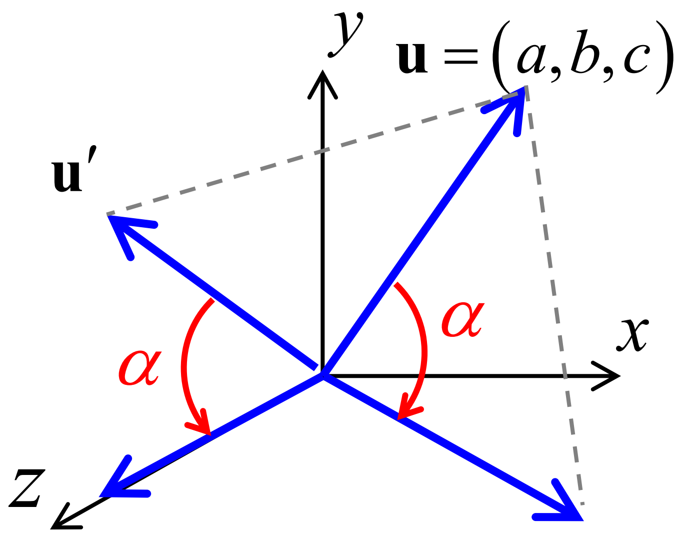
First rotate **u** around x axis to lay in $x-z$ plane.

Equivqlent to rotation **u**'s projection on $y-z$ plane around x axis.

$$\cos\alpha = c\Big/\sqrt{b^2+c^2} = c/d\,, \quad \sin\alpha = b/d\,.$$

We obtained a unit vector $\mathbf{w} = \left(a, 0, \sqrt{b^2+c^2} = d\right)$ in $x-z$ plane.

$$\mathbf{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c/d & -b/d & 0 \\ 0 & b/d & c/d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



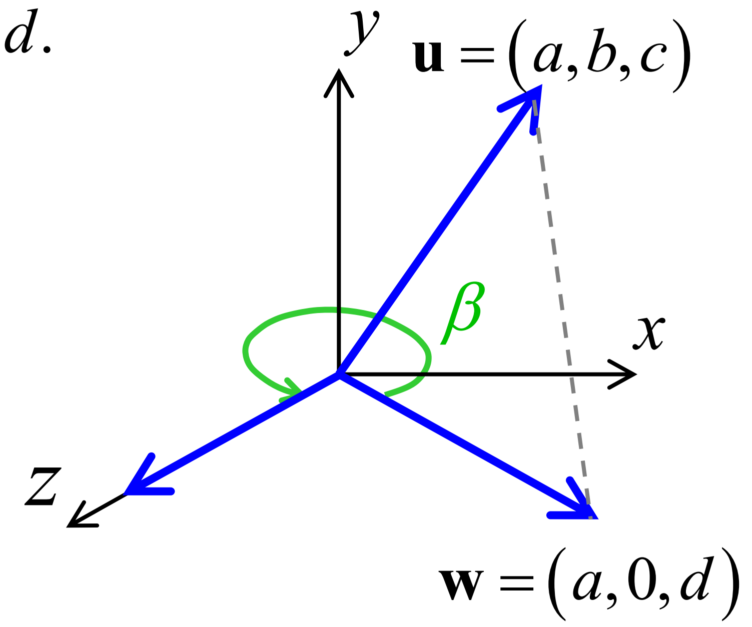
Rotate $\mathbf{w}$ counterclockwise around y axis.

$\mathbf{w}$ is a unit vector whose $x-$component is a, $y-$component is 0,

hence $z-$component is $\sqrt{b^2+c^2}=d$.

$\cos\beta = d, \quad \sin\beta = -a$

$$\mathbf{R}_y(\beta) = \begin{bmatrix} d & 0 & a & 0 \\ 0 & 1 & 0 & 0 \\ -a & 0 & d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$\mathbf{R}(\theta) = \mathbf{T}^{-1}\cdot\mathbf{R}_x^{-1}(\alpha)\cdot\mathbf{R}_y^{-1}(\beta)\cdot\mathbf{R}_z(\theta)\cdot\mathbf{R}_y(\beta)\cdot\mathbf{R}_x(\alpha)\cdot\mathbf{T}$$

$$\mathbf{R}_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R}(\theta) = \mathbf{T}^{-1} \cdot \mathbf{R}_x^{-1}(\alpha) \cdot \mathbf{R}_y^{-1}(\beta) \cdot \mathbf{R}_z(\theta) \cdot \mathbf{R}_y(\beta) \cdot \mathbf{R}_x(\alpha) \cdot \mathbf{T}$$

$$\mathbf{M}_R(\theta) =$$

$$\begin{bmatrix} a^2(1-\cos\theta)+\cos\theta & ab(1-\cos\theta)-c\sin\theta & ac(1-\cos\theta)+b\sin\theta \\ ba(1-\cos\theta)+c\sin\theta & b^2(1-\cos\theta)+\cos\theta & bc(1-\cos\theta)-a\sin\theta \\ ca(1-\cos\theta)-b\sin\theta & cb(1-\cos\theta)+a\sin\theta & c^2(1-\cos\theta)+\cos\theta \end{bmatrix}$$

Efficient 3D Rotations by Quaternions

Quaternions are extensions of complex numbers to 4-dimension.

$q = s + ia + jb + kc = (s, \mathbf{v})$ $\quad s$: real $\quad \mathbf{v} = (a, b, c)$: imaginary

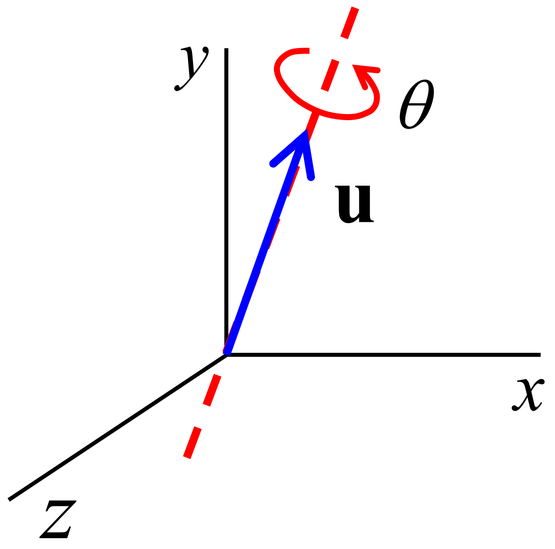
$$i^2 = j^2 = k^2 = -1, \quad ij = -ji = k, \quad jk = -kj = i, \quad ki = -ik = j$$

Addition: $q_1 + q_2 = (s_1 + s_2 \ , \mathbf{v_1} + \mathbf{v_2})$

Multiplication: $q_1 q_2 = (s_1 s_2 - \mathbf{v_1} \cdot \mathbf{v_2} \ , s_1 \mathbf{v}_2 + s_2 \mathbf{v}_1 + \mathbf{v_1} \times \mathbf{v_2})$

$$|q|^2 = s^2 + \mathbf{v} \cdot \mathbf{v}, \quad q^{-1} = \frac{1}{|q|^2}(s, -\mathbf{v}), \quad qq^{-1} = q^{-1}q = (1, 0)$$

Rotate a point position $\mathbf{p} = (x, y, z)$ about the unit vector $\mathbf{u}$.



Quarternion representation:

Rotation: $q = \left(\cos\dfrac{\theta}{2}, \mathbf{u}\sin\dfrac{\theta}{2} \right)$

Position: $\mathbf{P} = (0, \mathbf{p}), \quad \mathbf{p} = (x, y, z)$

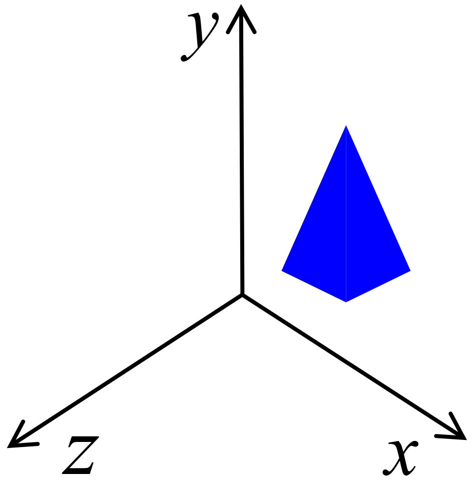
Rotation of $\mathbf{P}$ is carried out with the quarternion operation:

$$\mathbf{P}' = q\mathbf{P}q^{-1} = \left(0, s^2\mathbf{p} + \mathbf{v}(\mathbf{p}\cdot\mathbf{v}) + 2s(\mathbf{v}\times\mathbf{p}) + \mathbf{v}\times(\mathbf{v}\times\mathbf{p})\right)$$

This can be calculated by efficient HW for fast 3D rotations

when many rotation operations are involved.

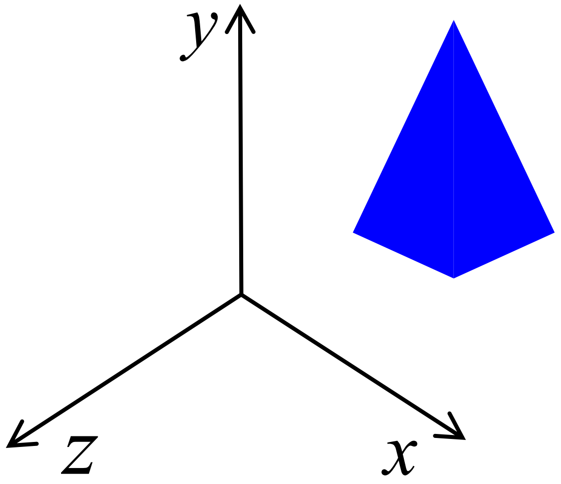
3D Scaling



$$x' = x \cdot S_x$$

$$y' = y \cdot S_y$$

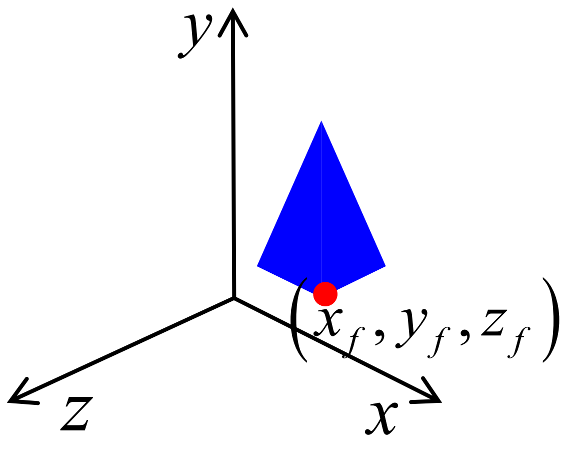
$$z' = x \cdot S_z$$

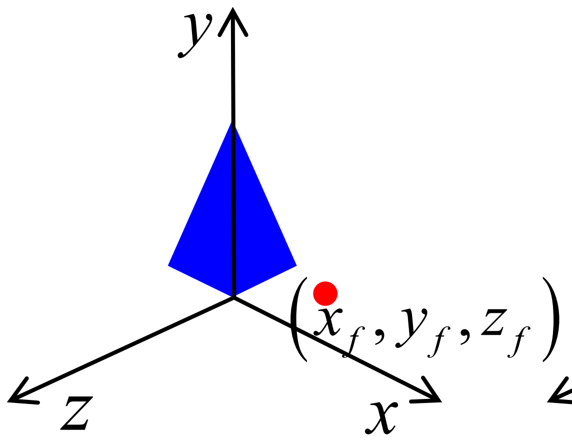


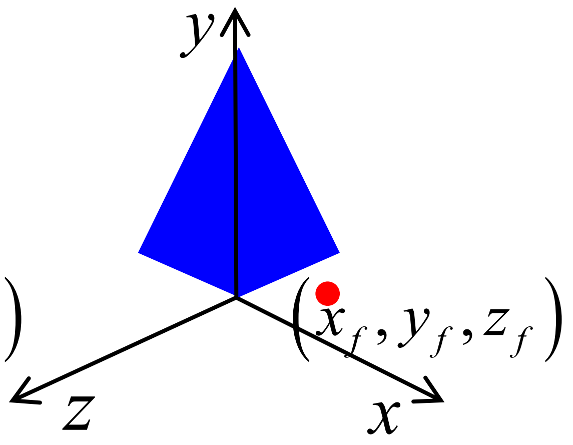
Enlarging object also moves it from origin

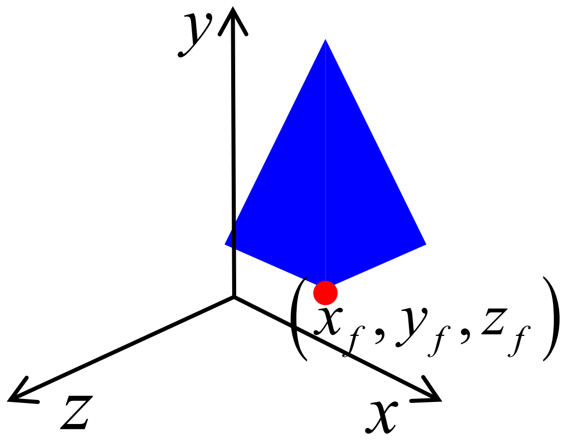
$$\mathbf{P}' = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \mathbf{S} \cdot \mathbf{P}$$

Scaling with respect to a fixed point (not necessarily of object)









$$\mathbf{T}\cdot\mathbf{S}\cdot\mathbf{T}^{-1} = \begin{bmatrix} S_x & 0 & 0 & (1-S_x)x_f \\ 0 & S_y & 0 & (1-S_y)y_f \\ 0 & 0 & S_z & (1-S_z)z_f \\ 0 & 0 & 0 & 1 \end{bmatrix}$$