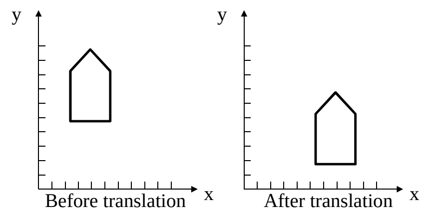
Overview

- 2D and 3D
  - Translation
  - Rotation
  - Scaling
- Homogeneous Coordinates
- Coordinate Systems

Geometric Transformations 2D Translation

- Repositioning an object along a straight line path from one coordinate location to another.
- Adding translation distances, t_x and t_y, to the original coordinate position.
- Rigid-body transformation: $x' = x + t_x$ $\quad y' = y + t_y$
  - Moves object without deformation.



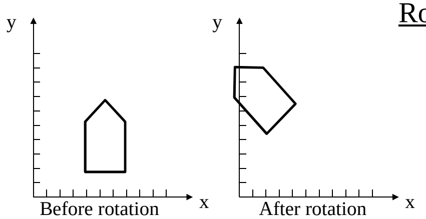
Matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix},$$

P' = P + T

Geometric Transformations 2D Rotation

- Repositioning an object along a circular path.
- Need a rotation angle θ and the position (x_r, y_r) of the pivot point which the object is to be rotated about.
- Positive rotation angles give counterclockwise rotation and negative angles give clockwise rotation.



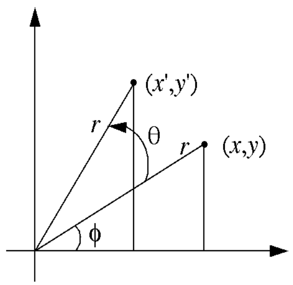
Rotation equation about origin:

$$x' = r\cos(\phi + \theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta$$

$$y' = r\sin(\phi + \theta) = r\cos\phi\sin\theta + r\sin\phi\cos\theta$$

Geometric Transformations 2D Rotation

Rotation about origin:



Original polar coordinates:

$$x = r\cos\phi \qquad y = r\sin\theta$$

After substitution:

$$x' = x\cos\theta - y\sin\theta$$

$$y' = x\sin\theta + y\cos\theta$$

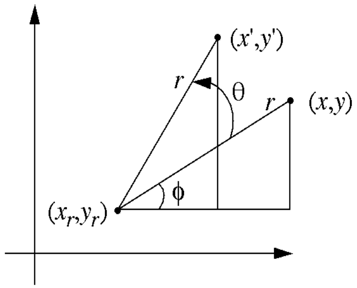
Matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix},$$

P' = R · P

Geometric Transformations
2D Rotation

Rotation about an arbitrary pivot point:



After substitution:

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

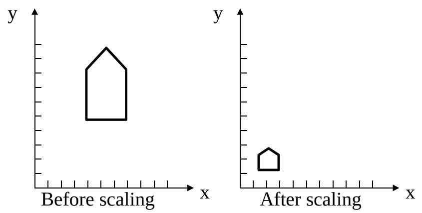
Geometric Transformations
2D Rotation

- Rigid-body transformation
- A line is rotated by applying the rotation equation to each of the line endpoints and redrawing the line between the tow endpoints.
- Polygons are rotated by moving each vertex through the specified rotation angle and then redraw.
- Curves are rotated by repositioning the defining points and redrawing the curve.

Geometric Transformations
2D Scaling

- Alters the size of an object.
- An object is scaled by multiplying the coordinates (x,y) of each vertex by a scaling factor s_x and s_y.
- s_x and s_y can be any positive value.
  - Values < 1 reduces the size of the object.
  - Values > 1 produces and enlargement.
  - If s_x and s_y is 1, then the size is unchanged.

$$x' = x \cdot s_x \qquad y' = y \cdot s_y$$



Matrix form:

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

P' = S · P

Geometric Transformations
2D Scaling

- Uniform scaling:
  - s_x and s_y have the same value.
- Differential scaling:
  - Unequal values of s_x and s_y.
- Scaling values < 1 moves the object closer to the origin.
- Scaling values > 1 moves the object further away from the origin.
- Fixed point:
  - To control the location after scaling.
  - Coordinates for the fixed point (x_f, y_f) can be any vertices, centroids or any other position.
  - $x_f(1-s_x)$ and $y_f(1-s_y)$ are constants for all points in the object.

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

Homogeneous Coordinates

Problem:

- Translation is addition and scaling and rotation are multiplication of matrices.

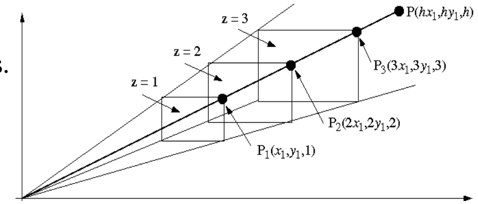
P' = P + T
P' = R · P
P' = S · P

Solved by:

- Using homogeneous coordinates instead of cartesian coordinates. Then translation, scaling and rotation can be expressed in a general matrix form (multiplication).

Homogeneous Coordinates

- A 2D coordinate $P_1(x_1,y_1)$ lying in 3D can be represented as $P(x,y,z) = P(hx_1,hy_1,h)$.
- Given P(m,n,h) in homogeneous coordinates the cartesian coordinates can be found by P(m/h,n/h,1).
- Each point can have many different homogeneous coordinate representations.



2D Translation Homogeneous Coordinates

Matrix representation:
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \qquad \mathbf{P' = T(}t_x,t_y\mathbf{) \cdot P}$$

The inverse translation matrix is obtained by replacing t_x and t_y with $-t_x$ and $-t_y$.

Translating from P to P' to P'':
$$\begin{bmatrix} 1 & 0 & t_{x_2} \\ 0 & 1 & t_{y_2} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & t_{x_1} \\ 0 & 1 & t_{y_1} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{x_1}+t_{x_2} \\ 0 & 1 & t_{y_1}+t_{y_2} \\ 0 & 0 & 1 \end{bmatrix}$$

2D Rotation Homogeneous Coordinates

Matrix representation:
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \qquad \mathbf{P' = R(\theta) \cdot P}$$

The inverse rotation matrix is obtained by replacing θ by $-\theta$.

Two successive rotations:
$$\begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2D Scaling Homogeneous Coordinates

Matrix representation: $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$, **P' = S($s_x$,$s_y$) · P**

The inverse scaling matrix is obtained by replacing s_x and s_y with $1/s_x$ and $1/s_y$.

Scaling from P to P' to P'':

$$\begin{bmatrix} s_{x_2} & 0 & 0 \\ 0 & s_{y_2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_{x_1} & 0 & 0 \\ 0 & s_{y_1} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_{x_1} \cdot s_{x_2} & 0 & 0 \\ 0 & s_{x_1} \cdot s_{x_2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

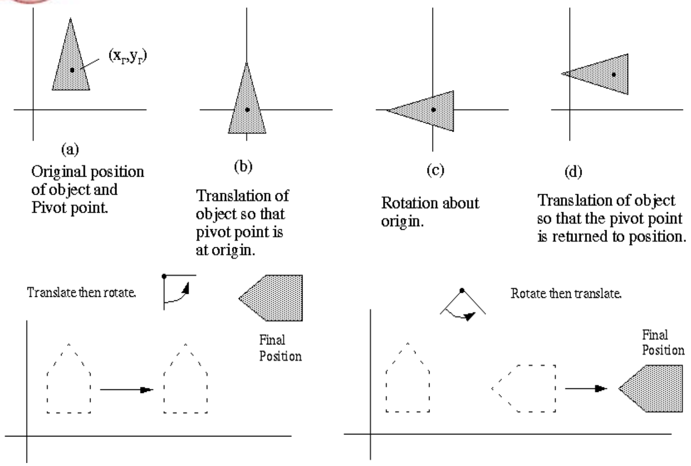
Composition of 2D Transformations

- OpenGL provides a rotation function only about the origin.
- To rotate an object about an arbitrary point (pivot point) we need to do a sequence of three fundamental transformations.
    1. Translate the pivot point to origin.
    2. Rotate about the origin.
    3. Translate the pivot point back to the original position.

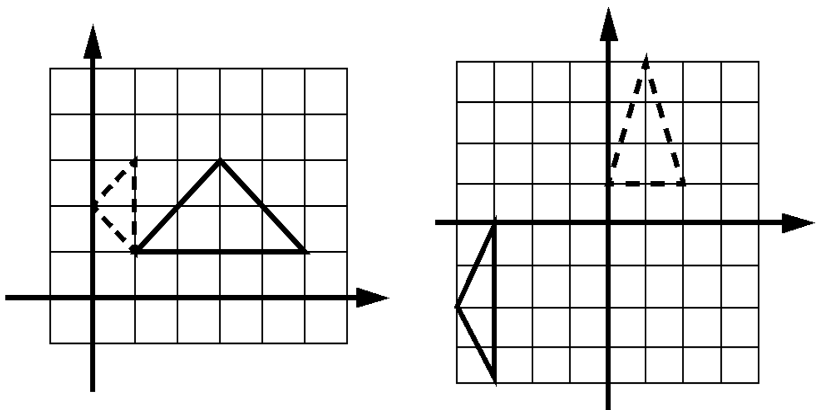
$$T(x_1, y_1) \cdot R(\theta) \cdot T(-x_1, -y_1) =$$

$$\begin{bmatrix} 1 & 0 & x_1 \\ 0 & 1 & y_1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -y_1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & x_1(1-\cos\theta)+y_1\sin\theta \\ \sin\theta & \cos\theta & y_1(1-\cos\theta)-x_1\sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

Composition of 2D Transformation



Composition of 2D Transformation Examples

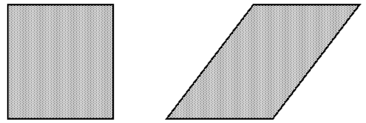


Affine Transformation

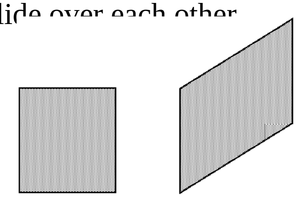
- Preserves parallelism of lines, but not lengths and angles.
- Rotation, translation and reflection preserves angles and lengths as well.
- Shear and scaling preserves only parallelism.
- These properties also applies to 3D.

Shear Transformation

- Distorts the shape of an object such that the transformed shape appears as if the object were composed of internal layer that had been caused to slide over each other



x-direction shear of unit cube.



y-direction shear of unit cube.

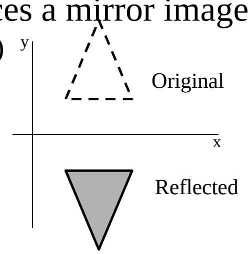
$$SH_x = \begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad SH_y = \begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Reflection Transformation

- Produces a mirror image of an object.

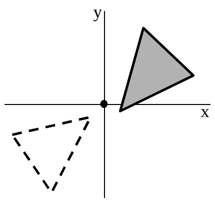
x-axis (y = 0)

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



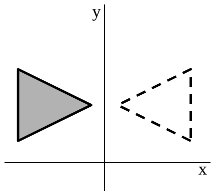
xy-plane

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



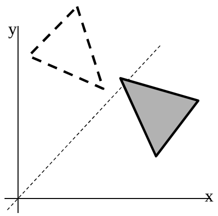
y-axis (x = 0)

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



xy-plane

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



3D Transformations
3D Translation

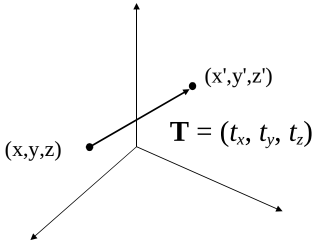
A 4 by 4 homogenized matrix

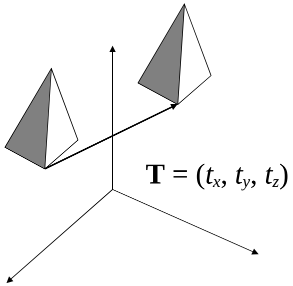
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \quad \mathbf{P'} = \mathbf{T} \cdot \mathbf{P}$$

The inverse translation matrix is obtained by replacing t_x, t_y and t_z with $-t_x$,$-t_y$ and $-t_z$.

3D Transformation
3D Translation

- Each of the defining points are translated.
- If the object is a polygon, each vertex of the polygon is translated.





3D Transformations
3D Scaling

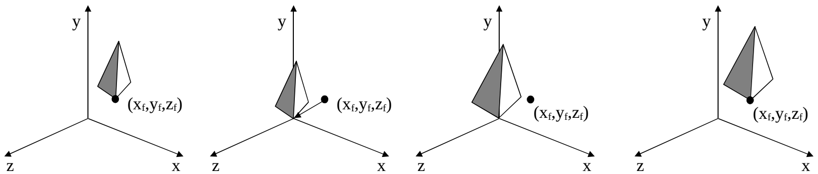
A 4 by 4 homogenized matrix $\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$, **P' = S ·P**

The inverse scaling matrix is obtained by replacing s_x, s_y and s_z with $1/s_x$, $1/s_y$ and $1/s_z$.

3D Transformation
3D Scaling

Scaling with respect to a fixed position

$$\mathrm{T}(x_f, y_f, z_f) \cdot \mathrm{S}(s_x, s_y, s_z) \cdot \mathrm{T}(-x_f, -y_f, -z_f) = \begin{bmatrix} s_x & 0 & 0 & (1-s_x)x_f \\ 0 & s_y & 0 & (1-s_y)y_f \\ 0 & 0 & s_z & (1-s_z)z_f \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

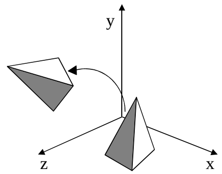


3D Transformation
3D Rotation

Rotation about z-axis

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

P' = R$_z$(θ) ·P



Rotation about x-axis

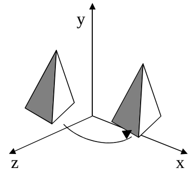
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

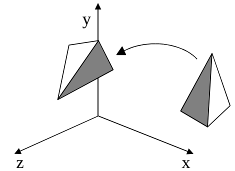
P' = R$_x$(θ) ·P

Rotation about y-axis

$$\begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

P' = R$_y$(θ) ·P





Geometric Transformation

A transformation matrix of the form: (translation and rotation)

$$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \quad or \quad \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

is called *special* orthogonal.

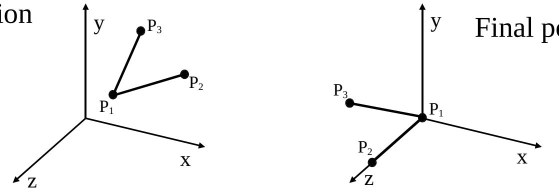
It preserves angles and length.
The inverse is the transpose.

Each row vector in the matrix has 3 properties:

1. Each is a unit vector
2. Each is perpendicular to the other
3. The first and second vector will be rotated by R(θ) to lie on the positive x and y axes, respectively.

Composition of 3D Transformation

Initial position



Final position

Two ways to achieve the transformation:

1. Compose the transformation T, R_x, R_y, R_z.
2. Using the properties of the orthogonal matrix.

Composition of 3D Transformation

Done the same way as 2D composition.

1. Translate P_1 to the origin
2. Rotate about the y-axis (P_1,P_2 lies in the (y,z) plane)
3. Rotate about the x-axis (P_1,P_2 lies on the z-axis)
4. Rotate about the z-axis (P_1,P_3 lies in the (y,z) plane)

The composite matrix will be

$$R_z(\alpha)\cdot R_x(\phi)\cdot R_y(\theta-90)\cdot T(-x_1,-y_1,-z_1)$$

Composition of 3D Transformation

Create the rotation matrix by using cross product.

R_z will rotate into z-axis.

$$R_z = \begin{bmatrix} r_{1z} & r_{2z} & r_{3z} \end{bmatrix}^T = \frac{P_1P_2}{\|P_1P_2\|}$$

R_x will rotate into x-axis.

$$R_x = \begin{bmatrix} r_{1x} & r_{2x} & r_{3x} \end{bmatrix}^T = \frac{P_1P_3 \times P_1P_2}{\|P_1P_3 \times P_1P_2\|}$$

R_y will rotate into y-axis.

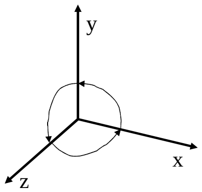
$$R_y = \begin{bmatrix} r_{1y} & r_{2y} & r_{3y} \end{bmatrix}^T = \frac{R_z \times R_x}{\|R_z \times R_x\|}$$

Composite Matrix

$$\begin{bmatrix} r_{1x} & r_{2x} & r_{3x} & 0 \\ r_{1y} & r_{2y} & r_{3y} & 0 \\ r_{1z} & r_{2z} & r_{3z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot T(-x_1,-y_1,-z_1) = R\cdot T$$

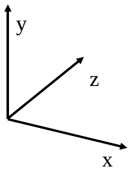
Coordinate Systems

Right handed:



Positive rotation gives
counterclockwise rotation

Left handed:



Positive rotation is clockwise