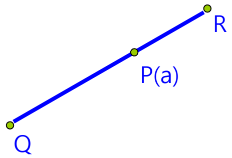
Geometric Objects and Transformation

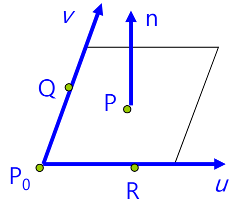
514780
2016년 봄학기
10/07/2016
박경신

Geometric Objects

- 선 (Line)
  - 2 points: R, Q
  - v = R- Q
    $P = Q + \alpha v = Q + \alpha(R - Q) = \alpha R + (1 - \alpha)Q$
  - $P = \alpha_1 R + \alpha_2 Q$ where $\alpha_1 + \alpha_2 = 1$ (affine sum)



- 평면 (Plane)
  - 3 points: P_0, Q, R
  - $T(\alpha, \beta) = P + \alpha u + \beta v$
  - $n \cdot (P - P_0) = 0$ where $n = u \times v$



- 3차원 객체 (3D objects)
  - 3차원 공간 내의 정점들의 집합이다.
  - 표면(surface)에 의해 기술되고, 속이 비었다 (hollow).
  - 평면 볼록 다각형 (convex polygons)으로 구성될 수 있다.
  - 임의의 다각형 (arbitrary polygons)은 삼각 다각형(triangular polygons)으로 분할(tessellate)된다.

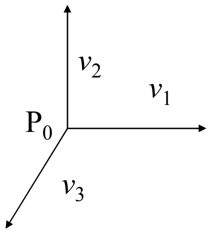
Coordinate Systems

- 기저벡터 (basis), $v_1, v_2, \ldots, v_n$
- 임의의 벡터 v는 $v = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n$
- $\{\alpha_1, \alpha_2, \ldots \alpha_n\}$는 기저에 대한 v의 성분(component)이다.
- 기저에 대한 v의 표현 (representation)을 행 행렬 또는 열 행렬로 기술 할 수 있다.

$$\mathbf{a} = [\alpha_1 \ \ \alpha_2 \ \ \ldots \ \ \alpha_n]^T = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \cdot \\ \alpha_n \end{bmatrix}$$

Frames

- 아핀공간 (affine space)은 점(point)을 포함한다.
- 원점(origin)으로부터 기저벡터로 좌표축을 그리는 표현을 프레임 (frame)이라 부른다.
- 프레임: (P_0, v_1, v_2, v_3)
- 프레임에서 벡터의 표현: $v = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n$
- 프레임에서 점의 표현: $P = P_0 + \beta_1 v_1 + \beta_2 v_2 + \ldots + \beta_n v_n$



$\mathbf{v} = [\alpha_1 \ \alpha_2 \ \alpha_3 \ 0]^T$
$\mathbf{p} = [\beta_1 \ \beta_2 \ \beta_3 \ 1]^T$

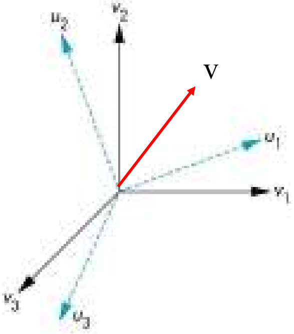
Change of Coordinate Systems

- 예를 들어, OpenGL에서는 모델 좌표계를 이용해서 기하를 정의한다. 그리고 세계 좌표계로 옮겨진다.
- 2개의 다른 기저벡터 $\{v_1, v_2, v_3\}$, $\{u_1, u_2, u_3\}$

$$u_1 = \gamma_{11}v_1 + \gamma_{12}v_2 + \gamma_{13}v_3$$
$$u_2 = \gamma_{21}v_1 + \gamma_{22}v_2 + \gamma_{23}v_3$$
$$u_3 = \gamma_{31}v_1 + \gamma_{32}v_2 + \gamma_{33}v_3$$

$$\mathbf{M} = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} \\ \gamma_{21} & \gamma_{22} & \gamma_{23} \\ \gamma_{31} & \gamma_{32} & \gamma_{33} \end{bmatrix}$$

$$\underline{\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \mathbf{M} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}}$$



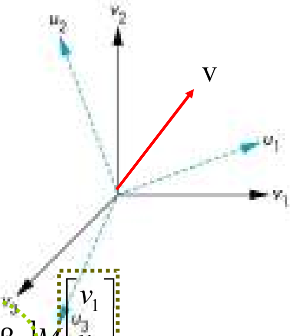
Change of Coordinate Systems

$$v = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 = \underbrace{\begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \end{bmatrix}}_{a^T} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

$$v = \underbrace{\begin{bmatrix} \beta_1 & \beta_2 & \beta_3 \end{bmatrix}}_{b^T} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$$

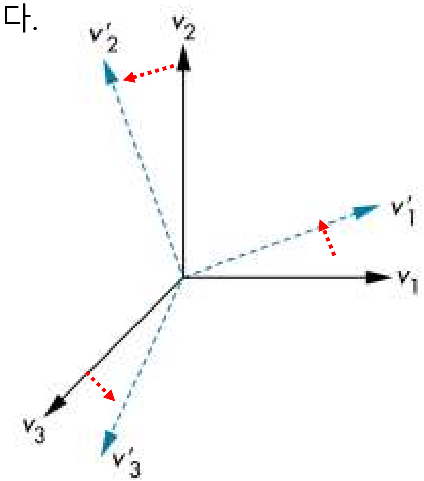
$$\begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \end{bmatrix} \boxed{\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}} = \begin{bmatrix} \beta_1 & \beta_2 & \beta_3 \end{bmatrix} \underline{\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}} = \begin{bmatrix} \beta_1 & \beta_2 & \beta_3 \end{bmatrix} \underline{M \boxed{\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}}}$$

$$\underbrace{\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}}_{a} = M^T \underbrace{\begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}}_{b} \quad \therefore \mathrm{a} = \mathrm{M}^T \mathrm{b} \quad \therefore \mathrm{b} = (\mathrm{M}^T)^{-1} \mathrm{a}$$



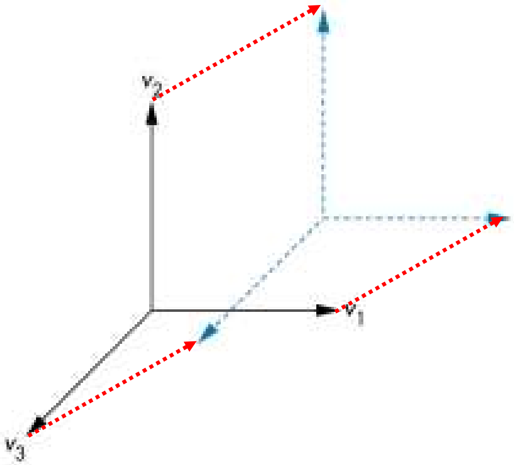
Rotation and Scaling of a Basis

- 기저 벡터들의 집합의 회전(rotation)과 크기(scaling) 변환을 표현함으로써 다른 기저 벡터들의 집합을 유도할 수 있다.



Translation of a Basis

- 그러나, 간단한 원점의 이동(translation) 변환은 3x3 matrix로 표현되지 않는다.

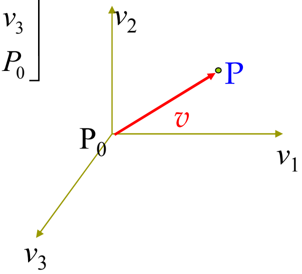


Homogeneous Coordinates

$$\textbf{vector}\ v=\sum \alpha_i v_i=\begin{bmatrix}\alpha_1 & \alpha_2 & \alpha_3 & 0\end{bmatrix}\begin{bmatrix}v_1\\ v_2\\ v_3\\ P_0\end{bmatrix}$$

$$\textbf{point}\ P=P_0+\sum \alpha_i v_i=\begin{bmatrix}\alpha_1 & \alpha_2 & \alpha_3 & 1\end{bmatrix}\begin{bmatrix}v_1\\ v_2\\ v_3\\ P_0\end{bmatrix}$$

$$P=\begin{bmatrix}\alpha_1\\ \alpha_2\\ \alpha_3\\ 1\end{bmatrix}, v=\begin{bmatrix}\alpha_1\\ \alpha_2\\ \alpha_3\\ 0\end{bmatrix}$$



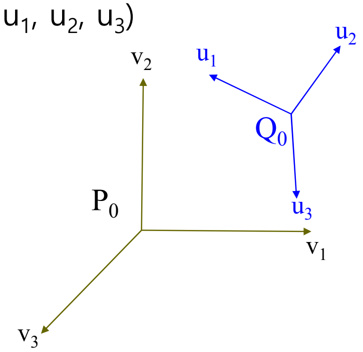
Change of Frames

- 2개의 프레임 (P_0, v_1, v_2, v_3) (Q_0, u_1, u_2, u_3)

$$u_1=\gamma_{11}v_1+\gamma_{12}v_2+\gamma_{13}v_3$$
$$u_2=\gamma_{21}v_1+\gamma_{22}v_2+\gamma_{23}v_3$$
$$u_3=\gamma_{31}v_1+\gamma_{32}v_2+\gamma_{33}v_3$$
$$Q_0=\gamma_{41}v_1+\gamma_{42}v_2+\gamma_{43}v_3+P_0$$

$$\begin{bmatrix}u_1\\ u_2\\ u_3\\ Q_0\end{bmatrix}=\mathbf{M}\begin{bmatrix}v_1\\ v_2\\ v_3\\ P_0\end{bmatrix}$$

$$\mathbf{M}=\begin{bmatrix}\gamma_{11} & \gamma_{12} & \gamma_{13} & 0\\ \gamma_{21} & \gamma_{22} & \gamma_{23} & 0\\ \gamma_{31} & \gamma_{32} & \gamma_{33} & 0\\ \gamma_{41} & \gamma_{42} & \gamma_{43} & 1\end{bmatrix}$$



Change of Frames

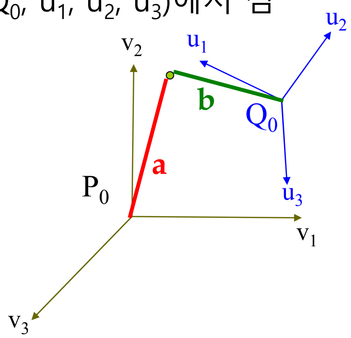
- a, b는 두 프레임 (P_0, v_1, v_2, v_3) (Q_0, u_1, u_2, u_3)에서 점 또는 벡터의 동차 좌표 표현이다.

$$\underline{b^T\begin{bmatrix}u_1\\ u_2\\ u_3\\ Q_0\end{bmatrix}}=b^T\mathrm{M}\boxed{\begin{bmatrix}v_1\\ v_2\\ v_3\\ P_0\end{bmatrix}}=\underline{a^T\boxed{\begin{bmatrix}v_1\\ v_2\\ v_3\\ P_0\end{bmatrix}}}$$

$$\mathrm{M}^T=\begin{bmatrix}\gamma_{11} & \gamma_{21} & \gamma_{31} & \gamma_{41}\\ \gamma_{12} & \gamma_{22} & \gamma_{32} & \gamma_{42}\\ \gamma_{13} & \gamma_{23} & \gamma_{33} & \gamma_{43}\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$\therefore a=\mathrm{M}^T b$$

$$\therefore b=(\mathrm{M}^T)^{-1}a$$



OpenGL Frames

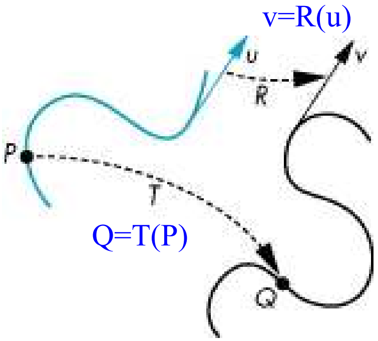
- 객체(모델) 좌표계 (model-view coordinate system)
- 세계 좌표계 (world coordinate system)
- 눈(카메라) 좌표계 (camera coordinate system)
- 투영변환 및 클리핑 (clipping coordinate system)
- 정규화 장치 좌표계 (normalized device coordinate system)
- 윈도우(화면) 좌표계 (screen coordinate system)

General Transformations

- 동차좌표(homogenous coordinates)를 사용한다면 벡터와 점을 4차원 열 행렬로 나타낼 수 있고, 주어진 프레임에서 점과 벡터의 표현을 변환하는 단일 함수 f로 변환을 정의할 수 있다.

$q = f(p)$

$v = f(u)$



Affine Transformations

- 아핀변환 (affine transformation)은 선형성(colinearity)을 유지한다.
  - 즉, 직선을 보존한다. 한 선에 있는 모든 점이 변환 후에도 변환된 선 상에 존재 한다.
- 또한, 거리의 비례(ratio of distance)를 유지한다.
  - 즉, 한 선분의 중점 (midpoint of a line)이 변환 후에도 변환된 선분의 중점에 위치한다.
- $P' = f(P)$
- $P' = f(\alpha P_1 + \beta P_2) = \alpha\, f(P_1) + \beta\, f(P_2)$

Affine Transformation

- 컴퓨터 그래픽스에서 필요로 하는 변환의 대부분은 아핀 변환이다. 아핀변환은 이동 (Translation), 회전 (Rotation), 크기변환 (Scaling), 밀림 (Shearing)을 포함한다.
- 변환된 점 P′ (x′, y′, z′)는 원래 점 P (x, y, z)의 선형조합 (linear combination)으로 표현할 수 있다., i.e.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Affine Transformation

- 변환된 점 P′ (x′, y′)는 원래 점 P (x, y)의 선형조합 (linear combination)으로 표현할 수 있다., i.e.

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha_{11}\,x + \alpha_{12}\,y + \alpha_{13} \\ \alpha_{21}\,x + \alpha_{22}\,y + \alpha_{23} \\ 1 \end{pmatrix}$$

⬇

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Geometric Transformation

- 기하학적 객체의 변환(geometric transformation)이란 하나 또는 그 이상의 기하학적 객체를 묘사하는 점의 그룹을 새로운 위치로 옮기는 함수를 의미한다.
- 이 때, 객체들의 정점들 사이의 관계를 유지하면서 하나의 변환으로 점들을 새로운 위치로 옮긴다.
- Basic transformation
  - 이동 (Translation)
  - 회전 (Rotation)
  - 크기변환 (Scaling)

OpenGL Column-Major Order

- **2D transformation matrix, M**

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

- **Point p가 column vector이면 (OpenGL) :**

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

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

- Point p가 row vector이면:

$$\mathbf{p}' = \mathbf{p}\mathbf{M}^{\mathrm{T}}$$

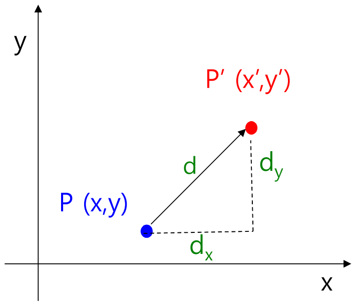
$$\begin{bmatrix} x' & y' \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

2D Translation

- 이동 (translation)은 주어진 방향으로 일정한 거리 d (d_x, d_y)만큼 점 P(x, y)을 옮겨서 새로운 점 P′(x′, y′)을 찾는 연산이다.

$x' = x + d_x$

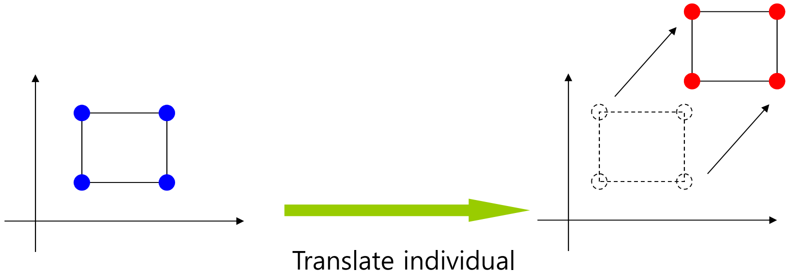
$y' = y + d_y$



$$P' = P + d \;\; where \;\; P' = \begin{pmatrix} x' \\ y' \end{pmatrix} P = \begin{pmatrix} x \\ y \end{pmatrix} d = \begin{pmatrix} d_x \\ d_y \end{pmatrix}$$

2D Translation

- 다수의 정점을 갖고 있는 객체를 이동한다면?



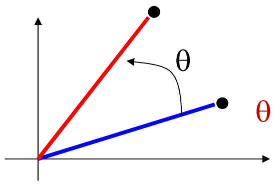
Translate individual vertices

2D Translation

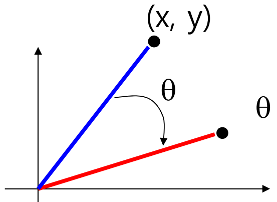
- **2D translation**
  $x' = x + d_x$
  $y' = y + d_y$
- **Inverse translation**
  $x = x' - d_x$
  $y = y' - d_y$
- **Identity translation**
  $x' = x + 0$
  $y' = y + 0$

Rotation

- 2차원 점이 원점(0,0)에 대해 각도 θ 만큼 회전 (2D rotation about the origin by θ)



θ > 0 : Rotate counter clockwise in RHS



θ < 0 : Rotate clockwise in LHS

$$\theta = \arctan\left(\frac{y}{x}\right)$$

OpenGL RHS CCW

2D Rotation

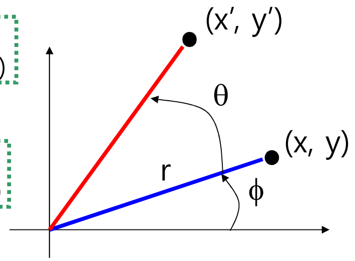
- Rotation of a point P(x,y) by θ about an origin (0,0)

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

$$x' = \boxed{r\cos(\phi+\theta)} = \boxed{r\cos(\phi)}\cos(\theta) - \boxed{r\sin(\phi)}\sin(\theta) = x\cos(\theta) - y\sin(\theta)$$

$$y' = \boxed{r\sin(\phi + \quad)} = \boxed{r\sin(\phi)}\cos(\theta) + \boxed{r\cos(\phi)}\sin(\theta) = y\cos(\theta) + x\sin(\theta)$$

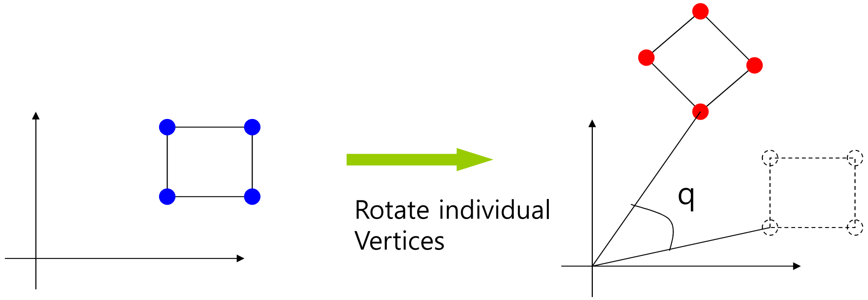
$x' = x\cos(\theta) - y\sin(\theta)$
$y' = y\cos(\theta) + x\sin(\theta)$



$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

2D Rotation

- 다수의 정점을 갖고 있는 객체를 회전한다면?



Rotate individual Vertices

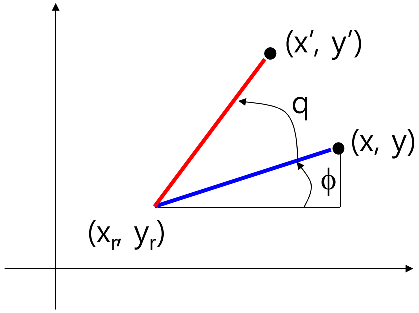
2D Rotation about an Arbitrary Pivot

- Rotation of a point P(x,y) by θ about an arbitrary pivot point, (x_r, y_r) :

  $P' = R(\theta)\, P$

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

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



2D Rotation

- **2D rotation**

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

- **Inverse rotation**

$$R^{-1} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

- **Identity rotation**

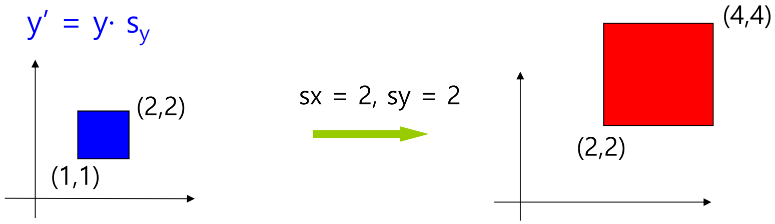
$$R_{\theta=0} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

2D Scale

- 크기변환 (scaling)은 객체를 scaling factor (s_x, s_y)로 크게 만들거나 작게 만들 수 있다. 즉, 아핀 비강체 (affine non-rigid-body transformation) 변환이다.
- 크기 변환은 고정점(pivot point)을 가진다. 따라서 객체의 크기만 커지는 것이 아니라 위치도 바뀌게 된다.

  $x' = x \cdot s_x$

  $y' = y \cdot s_y$



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

2D Scale about an Arbitrary Pivot

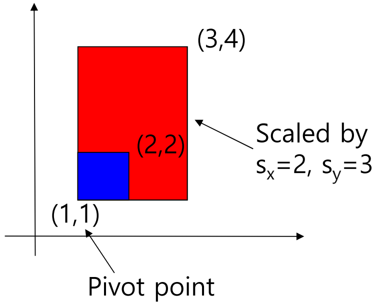
- Scale a point P(x,y) by a scaling factor relative to an arbitrary pivot point, (x_f, y_f) : $P' = S(s_x, s_y)\, P$

$x' = x_f + (x - x_f)\, s_x$

$y' = y_f + (y - y_f)\, s_y$

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

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



2D Scale

- **2D scale**

$$S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

- **Inverse scale**

$$S^{-1} = \begin{bmatrix} 1/s_x & 0 \\ 0 & 1/s_y \end{bmatrix}$$

- **Identity scale**

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

2D Reflection (Mirror)

- 반사 (Reflection)는 고정점에 대하여 객체가 반대방향으로 변환되는 것이다.
  - 반사는 각도 (angles)과 크기 (lengths)를 보존한다.
- 2D reflection over x axis

$x' = x$

$y' = -y$

- 2D reflection over y axis

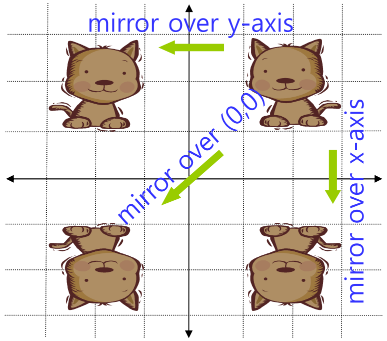
$x' = -x$

$y' = y$

- 2D reflection over (0,0)

$x' = -x$

$y' = -y$


mirror over y-axis
mirror over (0,0)
mirror over x-axis

2D Reflection (Mirror)

- 2D reflection over a line, y = x

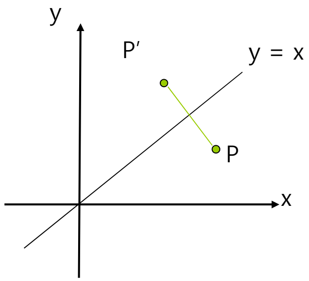
$x' = y$

$y' = x$

- 2D reflection over a line, y = -x

$x' = -y$

$y' = -x$



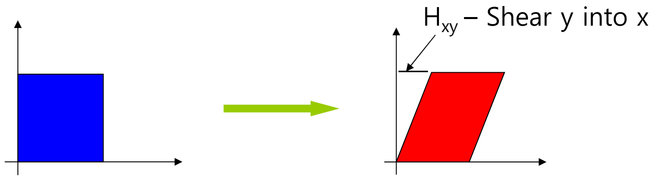
2D Shearing

- Y-축은 변하지 않고, x-축 방향으로 밀림(shearing) 변환한다:

$x' = x + y \cdot h_{xy}$

$y' = y$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & h_{xy} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$


H_{xy} – Shear y into x

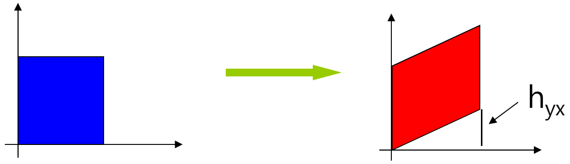
2D Shearing

- 밀림 (Shearing)변환은 객체의 크기에 변화를 주지 않는다.
- X-축은 변하지 않고, y-축 방향으로 밀림 변환한다:

$x' = x$

$y' = x \cdot h_{yx} + y$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ h_{yx} & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$



Homogeneous Coordinates

- 이동, 회전, 크기변환 행렬을 곱하기 위하여 동차좌표 (homogeneous coordinates)에서의 변환행렬로 바꿔서 사용한다.
- 동차 좌표는 2차원의 점 P (x, y)는 P (x, y, w)로 표현한다.
- (1, 2, 3)과 (2, 4, 6)은 같은 동차좌표 표현이다.
- 만약 점 P (x, y, w)의 w가 0이면, 그 점은 무한지점에 위치한다. 만약 w가 0가 아니라면, 그 점은 (x/w, y/w, 1)로 표현할 수 있다.

Transforming Homogeneous Coordinates

$$T(dx, dy) = \begin{pmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{pmatrix}$$

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

$$S(sx, sy) = \begin{pmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- 2차원 변환행렬은 동차좌표의 3x3 행렬로 표현할 수 있다.

3x3 2D Translation Matrix

- 행렬과 벡터의 곱 (matrix-vector multiplication)으로 표현된다.

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} d_x \\ d_y \end{pmatrix}$$

$$\Downarrow$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

3x3 2D Rotation Matrix

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

↓

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

3x3 2D Scale Matrix

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

↓

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

3x3 2D Shearing Matrix

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{pmatrix} 1 & h_{xy} \\ h_{yx} & 1 \end{pmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

↓

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{pmatrix} 1 & h_{xy} & 0 \\ h_{yx} & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Inverse 2D Transformation Matrix

$$T^{-1} = \begin{pmatrix} 1 & 0 & -d_x \\ 0 & 1 & -d_y \\ 0 & 0 & 1 \end{pmatrix}$$

$$R^{-1} = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$S^{-1} = \begin{pmatrix} 1/s_x & 0 & 0 \\ 0 & 1/s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Composing Transformation

- *Composing transformation* 이란 여러 개의 변환을 순서대로 적용시켜서 하나의 변환을 이루는 과정이다.
- 한 점을 변환하려고 한다면 한번에 한 개의 변환의 적용을 하거나 행렬의 곱셈을 수행한 후에 이 행렬을 점에 곱한다. $Q = (M3 \cdot \underbrace{(M2 \cdot (M1 \cdot P))}_{M}) = M3 \cdot M2 \cdot M1 \cdot P$

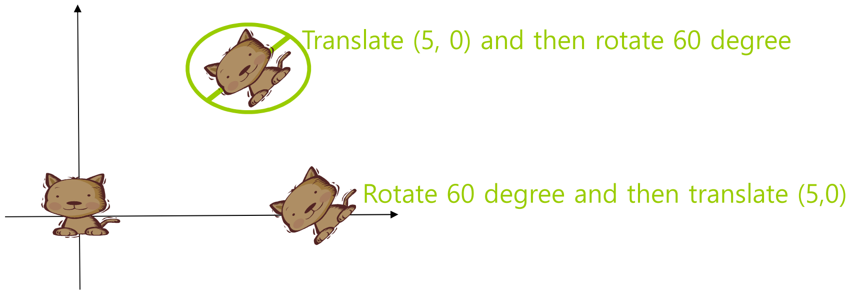
  (pre-multiply)
- 행렬의 곱은 결합법칙 (associative)을 만족한다.

  $M3 \cdot M2 \cdot M1 = (M3 \cdot M2) \cdot M1 = M3 \cdot (M2 \cdot M1)$
- 행렬의 곱은 교환법칙 (commutative)이 성립하지 않는다.

  $A \cdot B \;!=\; B \cdot A$

Transformation Order Matters!

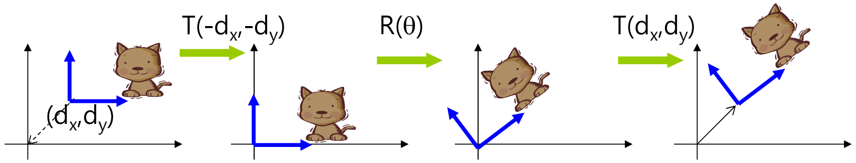
- 변환 행렬의 곱셈은 교환법칙이 성립하지 않는다.
- 만약 같은 변환 행렬이라 하더라도 곱하는 순서에 따라 완전히 다른 결과를 가질 수 있다.


Translate (5, 0) and then rotate 60 degree
Rotate 60 degree and then translate (5,0)

2D Rotate about an Arbitrary Pivot

- 임의의 점 $P(d_x, d_y)$에서 θ 만큼 2차원 회전 :
  1. $T(-d_x, -d_y)$
  2. $R(\theta)$
  3. $T(d_x, d_y)$

$$\begin{pmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & -d_x \\ 0 & 1 & -d_y \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & d_x(1-\cos\theta)+d_y\sin\theta \\ \sin\theta & \cos\theta & d_y(1-\cos\theta)-d_x\sin\theta \\ 0 & 0 & 1 \end{pmatrix}$$

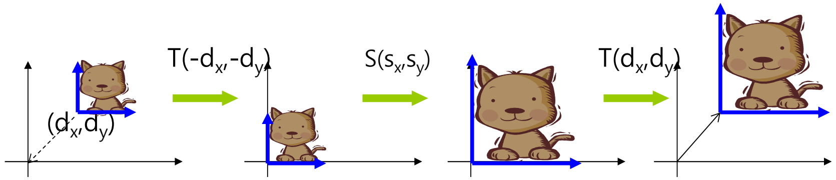


$T(-d_x,-d_y) \cdot R(\theta) \cdot T(d_x,d_y) = R(d_x, d_y, \theta)$

2D Scale about an Arbitrary Pivot

- 임의의 점 $P(d_x, d_y)$를 중심으로 크기변환:
  1. $T(-d_x, -d_y)$
  2. $S(s_x, s_y)$
  3. $T(d_x, d_y)$

$$\begin{pmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & -d_x \\ 0 & 1 & -d_y \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & d_x(1-s_x) \\ 0 & s_y & d_y(1-s_y) \\ 0 & 0 & 1 \end{pmatrix}$$

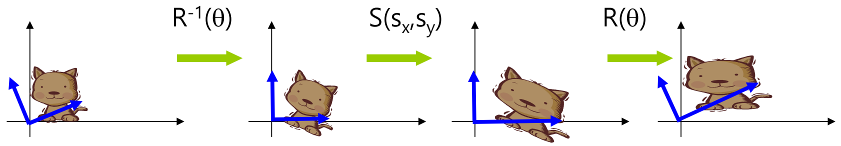


$T(-d_x,-d_y) \cdot S(s_x, s_y) \cdot T(d_x,d_y) = S(d_x, d_y, s_x, s_y)$

2D Scale in an Arbitrary Direction

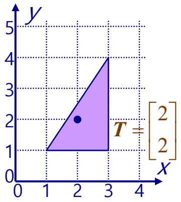
- 임의의 방향으로 크기변환 (*by rotating the object to align the desired scaling directions with the coordinate axes* before scale transformation)
  1. $R^{-1}(\theta)$
  2. $S(s_x, s_y)$
  3. $R(\theta)$

$$\begin{pmatrix}\cos\theta & -\sin\theta & 0\\ \sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}s_x & 0 & 0\\ 0 & s_y & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\cos\theta & \sin\theta & 0\\ -\sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{pmatrix} = \begin{pmatrix}s_x\cos^2\theta+s_y\sin^2\theta & (s_x-s_y)\cos\theta\sin\theta & 0\\ (s_x-s_y)\cos\theta\sin\theta & s_y\cos^2\theta+s_x\sin^2\theta & 0\\ 0 & 0 & 1\end{pmatrix}$$



Example: 2D Rotate about an Arbitrary Pivot

정점 (1,1), (3,1), (3,4)를 갖는 한 삼각형을 점 (2,2)에 대하여 45도 회전하라.



1. Translate point to origin $\mathbf{T}_1 = \begin{bmatrix}-2\\-2\end{bmatrix}$
2. Rotate 45 degrees $\mathbf{R} = \begin{bmatrix}.707 & -.707\\ .707 & .707\end{bmatrix}$
3. Translate back to original location $\mathbf{T}_2 = \begin{bmatrix}2\\2\end{bmatrix}$
4. Composite transformation $\boldsymbol{P'} = \boldsymbol{R}(\boldsymbol{P} + \boldsymbol{T_1}) + \boldsymbol{T_2}$

$$P' = \begin{bmatrix}.707 & -.707\\ .707 & .707\end{bmatrix}\left(\begin{bmatrix}x\\y\end{bmatrix}+\begin{bmatrix}-2\\-2\end{bmatrix}\right)+\begin{bmatrix}2\\2\end{bmatrix}$$

Example: 2D Rotate about an Arbitrary Pivot

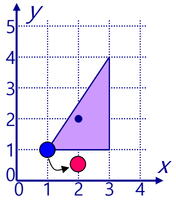
- P_1 (1, 1)

$$P_1' = \begin{bmatrix}.707 & -.707\\ .707 & .707\end{bmatrix}\left(\begin{bmatrix}1\\1\end{bmatrix}+\begin{bmatrix}-2\\-2\end{bmatrix}\right)+\begin{bmatrix}2\\2\end{bmatrix}$$

$$= \begin{bmatrix}.707 & -.707\\ .707 & .707\end{bmatrix}\left(\begin{bmatrix}-1\\-1\end{bmatrix}\right)+\begin{bmatrix}2\\2\end{bmatrix}$$

$$= \begin{bmatrix}0\\-1.414\end{bmatrix}+\begin{bmatrix}2\\2\end{bmatrix}$$

$$= \begin{bmatrix}2\\0.586\end{bmatrix}$$



Example: 2D Rotate about an Arbitrary Pivot

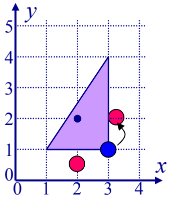
- P_2 (3, 1)

$$P_2' = \begin{bmatrix}.707 & -.707\\ .707 & .707\end{bmatrix}\left(\begin{bmatrix}3\\1\end{bmatrix}+\begin{bmatrix}-2\\-2\end{bmatrix}\right)+\begin{bmatrix}2\\2\end{bmatrix}$$

$$= \begin{bmatrix}.707 & -.707\\ .707 & .707\end{bmatrix}\left(\begin{bmatrix}1\\-1\end{bmatrix}\right)+\begin{bmatrix}2\\2\end{bmatrix}$$

$$= \begin{bmatrix}1.414\\0\end{bmatrix}+\begin{bmatrix}2\\2\end{bmatrix}$$

$$= \begin{bmatrix}3.414\\2\end{bmatrix}$$



Example: 2D Rotate about an Arbitrary Pivot

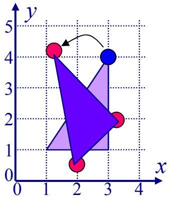
- P_3 (3, 4)

$$P_3' = \begin{bmatrix} .707 & -.707 \\ .707 & .707 \end{bmatrix}\left(\begin{bmatrix} 3 \\ 4 \end{bmatrix} + \begin{bmatrix} -2 \\ -2 \end{bmatrix}\right) + \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

$$= \begin{bmatrix} .707 & -.707 \\ .707 & .707 \end{bmatrix}\left(\begin{bmatrix} 1 \\ 2 \end{bmatrix}\right) + \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

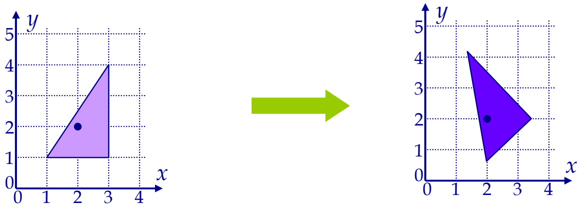
$$= \begin{bmatrix} -.707 \\ 2.121 \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

$$= \begin{bmatrix} 1.293 \\ 4.121 \end{bmatrix}$$



Example: 2D Rotate about an Arbitrary Pivot

- Result:



변환 전 삼각형의 세 점:
(1, 1), (3, 1), (3, 4)

점 (2, 2)를 중심으로 회전
변환 후:
(2, 0.59), (3.41, 2), (1.29, 4.2)

Example: 2D Rotate about an Arbitrary Pivo Using Composite Transformation Matrix

- 정점 (1,1), (3,1), (3,4)를 갖는 한 삼각형을 점 (2,2)에 대하여 45도 회전하라.
- P′= T(2,2)R(45)T(-2,-2)P = M P

$$M = T_{(2,2)} R_{45} T_{(-2,-2)})$$

$$= \left(\begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \cos(\mathbf{4}5°) & -\sin(\mathbf{4}5°) & 0 \\ \sin(\mathbf{4}5°) & \cos(\mathbf{4}5°) & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}\right)$$

$$= \begin{bmatrix} .707 & -.707 & 2 \\ .707 & .707 & 2 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} .707 & -.707 & 2 \\ .707 & .707 & -.828 \\ 0 & 0 & 1 \end{bmatrix} = \mathbf{M}$$

Example: 2D Rotate about an Arbitrary Pivo Using Composite Transformation Matrix

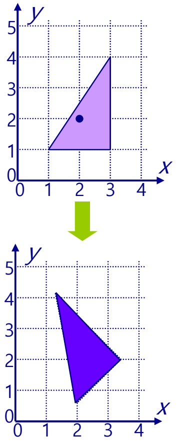
1. P_1

$$P_1' = MP_1 = \begin{bmatrix} .707 & -.707 & 2 \\ .707 & .707 & -.828 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ .586 \\ 1 \end{bmatrix}$$

2. P_2

$$P_2' = MP_2 = \begin{bmatrix} .707 & -.707 & 2 \\ .707 & .707 & -.828 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3.414 \\ 2 \\ 1 \end{bmatrix}$$

3. P_3

$$P_3' = MP_3 = \begin{bmatrix} .707 & -.707 & 2 \\ .707 & .707 & -.828 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 3 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 1.293 \\ 4.121 \\ 1 \end{bmatrix}$$