D2D1::Matrix3x2F 클래스를 사용해서 저희는 변환(Transformation)을 구현했습니다. 그려진 물체가 회전하고, 커지고, 이동하고 했었던 것 기억나시나요? 거기에는 행렬(Matrix)가 숨어있습니다. 이러한 연산들은 행렬의 연산을 통해서 이루어지게 됩니다. 이번 장에서 행렬에 대해 알아보고, 행렬을 어떻게 사용해서 그러한 결과를 얻어낸 것인지 알아봅니다. 또한 왜 행렬을 사용하는지, 그리고 행렬을 이용한 변환의 특성에 대해서도 아실 수 있습니다.

행렬(Matrix)

위키백과에 따르면, 행렬은 수(스칼라)나 수식, 기호 등을 네모꼴로 변형한 것으로 괄호로 묶어서 표시한 것입니다. 행렬의 각 항들을 요소(Element)라고 한다. 아래 그림처럼 나타내며, 우리나라의 고등학교에서는 행렬을 소괄호 ()로 감싸지만 대학에서는 대괄호[]를 사용합니다. m행 n열 행렬을 $\mathrm{m} \times \mathrm{n}$ 행렬 이라고 합니다. 또한 행렬의 모든 원소가 0인 행렬을 영행렬이라고 합니다.

	1열	2열	3열
1행	1	2	3
2행	4	5	6
3행	7	8	9

행렬 M이 있을 때, i항 j열의 원소를 M_{ij} 혹은 $\mathrm{M}(i,j)$라고 합니다. 행렬의 행과 열의 수가 같다면 그 행렬을 정방행렬(Square Matrix)이라고 합니다. 행렬에도 몇 가지 연산들이 존재합니다.

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

덧셈

$$\begin{bmatrix} \mathrm{a}_1 & b_1 & c_1 \\ d_1 & e_1 & f_1 \\ g_1 & h_1 & i_1 \end{bmatrix} + \begin{bmatrix} \mathrm{a}_2 & b_2 & c_2 \\ d_2 & e_2 & f_2 \\ g_2 & h_2 & i_2 \end{bmatrix} = \begin{bmatrix} \mathrm{a}_1 + \mathrm{a}_2 & b_1 + b_2 & c_1 + c_2 \\ d_1 + d_2 & e_1 + e_2 & f_1 + f_2 \\ g_1 + g_2 & h_1 + h_2 & i_1 + i_2 \end{bmatrix}$$

뺄셈

$$\begin{bmatrix} \mathrm{a}_1 & b_1 & c_1 \\ d_1 & e_1 & f_1 \\ g_1 & h_1 & i_1 \end{bmatrix} - \begin{bmatrix} \mathrm{a}_2 & b_2 & c_2 \\ d_2 & e_2 & f_2 \\ g_2 & h_2 & i_2 \end{bmatrix} = \begin{bmatrix} \mathrm{a}_1 - \mathrm{a}_2 & b_1 - b_2 & c_1 - c_2 \\ d_1 - d_2 & e_1 - e_2 & f_1 - f_2 \\ g_1 - g_2 & h_1 - h_2 & i_1 - i_2 \end{bmatrix}$$

스칼라 곱셈

$$x \cdot \begin{bmatrix} a_1 & b_1 & c_1 \\ d_1 & e_1 & f_1 \\ g_1 & h_1 & i_1 \end{bmatrix} = \begin{bmatrix} xa_1 & xb_1 & xc_1 \\ xd_1 & xe_1 & xf_1 \\ xg_1 & xh_1 & xi_1 \end{bmatrix}$$

스칼라 나눗셈

$$\begin{bmatrix} a_1 & b_1 & c_1 \\ d_1 & e_1 & f_1 \\ g_1 & h_1 & i_1 \end{bmatrix} / x = \frac{1}{x} \begin{bmatrix} a_1 & b_1 & c_1 \\ d_1 & e_1 & f_1 \\ g_1 & h_1 & i_1 \end{bmatrix} = \begin{bmatrix} \frac{a_1}{x} & \frac{b_1}{x} & \frac{c_1}{x} \\ \frac{d_1}{x} & \frac{e_1}{x} & \frac{f_1}{x} \\ \frac{g_1}{x} & \frac{h_1}{x} & \frac{i_1}{x} \end{bmatrix}$$

행렬의 한 열이나 행을 하나의 벡터로 생각할 수 있습니다. $m \times n$ 행렬의 행 벡터는 n차원 벡터이고, 열 벡터는 m차원 벡터입니다. 행렬의 i행 벡터를 $\mathrm{row}_i(\mathrm{A})$, 행렬의 j열을 $\mathrm{col}_i(\mathrm{B})$ 행렬에도 곱셈 연산이 존재하는데, 행렬의 곱셈 연산은 생각과는 많이 다릅니다. 행렬 A, B가 있다고 하고, 행렬 C가 A와 B의 곱셈이라고 한다면. 행렬 $\mathrm{C_{ij}}$는 A의 i행열과 B의 j열 벡터의 내적이에요.

$$\mathrm{A} \times \mathrm{B} = \mathrm{C}$$
$$\mathrm{C}_{ij} = row_i(\mathrm{A}) \cdot col_j(\mathrm{B})$$

그렇기 때문에, A의 열의 수와 B의 행의 수가 같아야만 행렬의 곱을 계산할 수 있으며, 곱의 결과인 C의 행의 길이는 A의 해의 길이와 같고, C의 열의 길이는 B의 열의 길이와 같습니다. 좀 복잡하죠? $\mathrm{a} \times \mathrm{b}$ 크기의 행렬과 $\mathrm{c} \times \mathrm{d}$ 크기의 행렬의 곱이 성립할 때, $\mathrm{b} = \mathrm{c}$ 여야 합니다. 또한 $m \times n$ 행렬과 $n \times l$ 행렬을 곱한 행렬의 크기는 $m \times l$ 입니다.

$$\begin{bmatrix} \mathrm{a} & \mathrm{b} \\ \mathrm{c} & \mathrm{d} \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix}$$

$$\begin{bmatrix} \mathrm{a}_{11} & \mathrm{a}_{12} \\ \mathrm{a}_{21} & \mathrm{a}_{22} \\ \mathrm{a}_{31} & \mathrm{a}_{32} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \\ a_{31}b_{11} + a_{32}b_{21} & a_{31}b_{12} + a_{32}b_{22} \end{bmatrix}$$

1. 행렬의 성질

1.1. 행렬의 성질

두 행렬 A, B가 존재하고 O는 영행렬(모든 항이 0인 행렬)일 때, 아래와 같은 성질이 존재합니다.

이름	성질

덧셈의 교환 법칙	$A + B = B + A$
덧셈의 결합 법칙	$A + B + C = A + (B + C)$
곱셈의 교환 법칙(비성립)	모든 행렬이 $AB \neq BA$ 을 만족하지는 않는다.
곱셈의 결합 법칙	$ABC = (AB)C = A(BC)$
분배 법칙	$A(B + C) = AB + AC$
기타	$AB = O$ 일 때, A 혹은 B는 영행렬이 아닐 수도 있다.
	$AB = BA = A$ 일 때, B는 항등행렬(Identity Matrix)이라 하고, I로 나타낸다.
	$AB = BA = I$ 일 때, A를 B의 역행렬이라고 하고, B를 A의 역행렬이라고 한다.
	모든 행렬이 역행렬을 갖지는 않는다.

1.2. 항등행렬(Identity Matrix)

항등행렬은 I로 나타내며, 아래와 같은 성질을 만족합니다.

$$AI = IA = A$$

항등행렬은, 아래와 같이 행과 열이 같은 항이 전부 1이고, 나머지는 0인 행렬입니다. $n \times n$ 항등행렬을 n차 항등행렬이라고 합니다.

$$\begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix}$$

$$I_{ij} = \begin{cases} 1 \ (i = j) \\ 0 \ (i \neq j) \end{cases}$$

1.3. 역행렬(Inverse Matrix)

행렬 A가 존재할 때, 아래의 법칙을 만족하는 행렬을 역행렬이라 하고, A^{-1}과 같이 나타냅니다.

$$AA^{-1} = A^{-1}A = I$$

추가적으로 아래와 같은 성질로 성립합니다(AB의 역행렬이 존재한다는 가정 하에).

$$(AB)^{-1} = B^{-1}A^{-1}$$

2차원 정방행렬의 역행렬을 구하는 공식은 아래와 같습니다.

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

$$\mathrm{A}^{-1} = \frac{1}{ad - bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

위 식을 보면, 만약 $\mathrm{ad} - \mathrm{bc}$ 가 0이라면 위 식은 성립하지 않습니다. 즉 역행렬이 존재하지 않게 되죠. 이렇게 행렬이 역행렬을 가지는지를 알 수 있는 식을 행렬식(Determinant)이라고 합니다. 행렬식은 행렬의 차수에 따라 다릅니다. 행렬 M의 행렬식은 $\det M$이라고 쓰거나 아래처럼 막대기 괄호로 감싸기도 합니다. 행렬식이 0이 아닌 행렬, 즉 역행렬이 존재하는 행렬을 가역행렬(Invertible Matrix)라고 합니다.

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

$$\det M = \begin{vmatrix} a & b \\ c & d \end{vmatrix}$$

1.4. 행렬식 계산하기

행렬 M에서 i행과 j열을 삭제해서 행의 수와 열의 수가 1씩 줄어든 행렬을 $M_{<i,j>}$라고 해보죠, 그렇다면 n차 정방행렬 M의 행렬식은 아래와 같은 공식으로 얻을 수 있습니다. 시그마(Σ)에 대해서는 아마 모르실 테지만 위의 식을 간편하게 줄인 것이에요.

$$\begin{aligned} \det M &= M_{11} \det M_{<1,1>} - M_{12} \det M_{<1,2>} \\ &\quad + \cdots (-1)^{n+1} M(1,n) \det M_{<1,n>} \\ &= \sum_{1}^{n} (-1)^{n+1} M(1,n) \det\ M_{<1,n>} \end{aligned}$$

1차 행렬 $\mathrm{M} = [\mathrm{a}]$ 이 주어질 때 $\det M = a$ 입니다.

위 공식을 통해서 2차 정방행렬의 행렬식을 계산해 보면

$$\begin{aligned} \mathrm{M} &= \begin{bmatrix} \mathrm{a} & \mathrm{b} \\ \mathrm{c} & \mathrm{d} \end{bmatrix} \\ \det M &= (-1)^2 ad + (-1)^3 bc \\ &= ad - bc \end{aligned}$$

3차 정방행렬은

$$\begin{aligned} \mathrm{M} &= \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \\ \mathrm{def}\, M &= a(ei - fh) - b(di - fg) + c(dh - eg) \end{aligned}$$

4차 정방행렬의 행렬식은 너무 길어서 여기서 쓰진 않겠습니다.

1.5. 전치행렬(Transpose Matrix)

전치행렬은 아래와 같은 성질을 만족하는 행렬로, $\mathrm{m}\times\mathrm{n}$ 행렬 M의 전치행렬은 M^{T}라고 씁니다. M^{T}는 $\mathrm{n}\times\mathrm{m}$ 크기의 행렬이 됩니다.

$$\mathrm{M}_{ij}^{\mathrm{T}} = M_{ji}$$

$$\text{ex) } \mathrm{M} = \begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix}$$

$$\mathrm{M}^{\mathrm{T}} = \begin{bmatrix} a & c & e \\ b & d & f \end{bmatrix}$$

전치 행렬은 아래와 같은 성질이 있습니다.

$$(\mathrm{M}^{\mathrm{T}})^{\mathrm{T}} = \mathrm{M}$$
$$(\mathrm{A}+\mathrm{B})^{\mathrm{T}} = \mathrm{A}^{\mathrm{T}} + \mathrm{B}^{\mathrm{T}}$$
$$(\mathrm{AB})^{\mathrm{T}} = \mathrm{B}^{\mathrm{T}}\mathrm{A}^{T}$$

1.6. 여인수행렬

행렬 M이 있을 때, M의 여인수행렬 C(M)은 아래의 식을 만족합니다.

$$\mathrm{C}(\mathrm{M})_{ij} = (-1)^{i+j} \det M_{<i,j>}$$

2차 정방행렬의 여인수행렬은

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

$$\mathrm{C}(\mathrm{M}) = \begin{bmatrix} \mathrm{d} & -\mathrm{c} \\ -\mathrm{b} & \mathrm{a} \end{bmatrix}$$

행렬의 역행렬을 여인수행렬을 통해서 나타낼 수 있습니다.

$$\mathrm{M}^{-1} = \frac{1}{\det M} C(M)^{T}$$

2. 아핀 변환(Affine Transformation)

아핀 변환은 벡터를 다른 벡터공간으로 대응시키는 변환을 의미합니다. 벡터공간이란 개념은

선형대수에서 더 깊은 개념이기 때문에 여기서는 다루지 않겠습니다. 그냥 단순히 시점이라고 생각하면 됩니다. 책상 위에 컴퓨터가 있다고 하면, 컴퓨터의 위치는 바뀌지 않아도 내가 컴퓨터를 바라보는 위치가 바뀐다면 컴퓨터는 나에게 상대적으로 다르게 보입니다. 이 개념을 적용한 것이 아핀 변환입니다. 아핀 변환에는 여러가지가 있지만 대표적으로 사용되는 스케일(Scale), 회전(Rotate), 평행이동(Translation)만을 알아보도록 하겠습니다.

변환은 변환 행렬 M과 변환할 벡터 V의 곱으로 계산합니다. n차원 벡터를 $1 \times n$ 혹은 $n \times 1$ 행렬로 생각하여 곱하는 것인데, 중요한 것은 곱하는 순서입니다. $V \times M$ 처럼 벡터에 행렬을 곱하는 전위곱(pre-multiplication)이냐, 아니면 반대로 행렬에 벡터를 곱하는 후위곱(post-multiplication)이냐에 따라서 행렬이 달라지기 때문입니다. 여기서는 일반적으로 수학책 같은 곳에서 많이 쓰이는 후위곱을 사용하도록 하겠습니다. 또한 여기서는 2차원 변환만을 다룰 것입니다.

2.1. 척도 변환(Scaling)

스케일은 벡터의 원소에 일정한 수를 곱하는 연산입니다. X축으로 S_x만큼, Y축 값을 S_y만큼 곱하고 싶다면 변환 행렬 S는 아래와 같은 모습이 됩니다. 스케일은 곱셈 순서가 뒤바뀌어도 상관 없습니다.

$$S \times v = (S_x v_x,\ S_y v_y) = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$

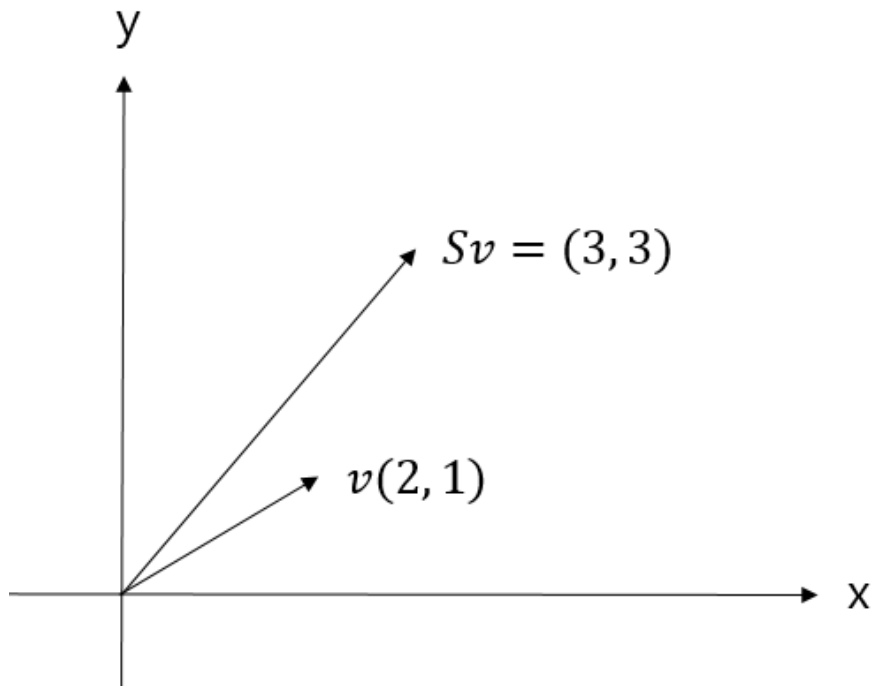

그림 1. 스케일 변환

만약 S_x나 S_y가 음수라면 그 축으로 뒤집히게 됩니다.

2.2. 회전 변환(Rotation)

회전 변환은 원점을 기준으로 일정 각만큼 회전하는 변환입니다. 각이 양수이면 반시계방향으로, 음수이면 시계방향으로 회전하게 됩니다. θ°만큼 회전하는 회전 변환 행렬은 아래와 같습니다.

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

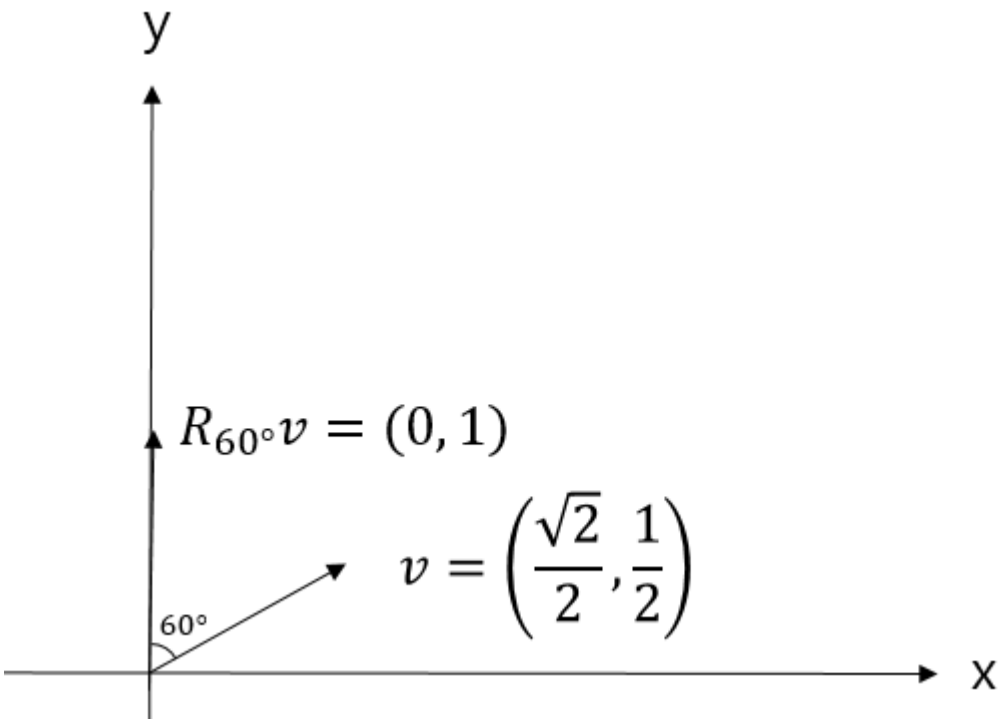

그림 2. 회전 변환

만약 후위곱이 아닌 전위곱 방식을 사용하고 있다면(벡터 곱하기 행렬) 위 변환행렬의 전치를 곱하면 같은 결과를 얻을 수 있습니다. 즉

$$vR = \mathrm{R}^{\mathrm{T}} v$$

2.3. 변환 결합(Combination)

이런 변환을 행렬로 나타내는 이유는 여러 단계의 변환을 행렬 하나에 몰아서 나타낼 수 있기 때문입니다. 예를 들어, 3개의 변환 A, B, C가 있다고 할 때, 벡터 V를 변환하기 위해선 $\mathrm{A}\times\mathrm{B}\times\mathrm{C}\times\mathrm{V}$총 4번의 곱셈을 해야 합니다. 그러나 결합법칙에 의하며 이를 $(\mathrm{A}\times\mathrm{B}\times\mathrm{C})\times\mathrm{V}$로 나타낼 수 있으며, $\mathrm{A}\times\mathrm{B}\times\mathrm{C}=\mathrm{T}$라 할 때, $\mathrm{T}\times\mathrm{V}$로 더 짧게 나타낼 수 있습니다. 만약 300개의 점으로 이루어진 도형이 있다고 했을 때, T를 미리 계산해 놓는다면 A, B, C를 곱하는 것보다 더 적은 횟수의 연산으로 같은 결과를 얻어낼 수 있습니다. 잘 믿어지지 않는다면 실제로 계산해 보길 바랍니다.

3. 동차 좌표계(Homogeneous Coordinate System)

기존의 변환은 한가지 문제점을 안고 있습니다. 기존의 변환으로는 평행이동을 나타낼 수 없다는 것이죠. 평행이동 행렬 T와 벡터 v의 곱은 아래와 같은 형태를 띠어야 합니다.

$$\mathrm{T} \times v = v + (dx,\ dy)$$

하지만 기존의 행렬곱의 형태로는 이것이 불가능합니다. 이에 대한 해법으로 수학자들은 벡터와 행렬의 차수를 1늘리는 방법을 생각해 내었습니다. 하지만 우리는 2차원 벡터만을 다룰 것입니다. 그렇기 때문에 3차원 좌표값은 존재하면 안 됩니다. 따라서 모든 3차원 좌표값은 1로 통일합니다. 즉 행렬곱은 아래와 같은 모양이 됩니다.

$$\begin{bmatrix} \mathrm{a} & \mathrm{b} & \mathrm{c} \\ \mathrm{d} & \mathrm{e} & \mathrm{f} \\ \mathrm{g} & \mathrm{h} & \mathrm{i} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} av_x + bv_y + c \\ dv_x + ev_y + f \\ gv_x + hv_y + i \end{bmatrix}$$

이를 이용해서 X축으로 $\mathrm{T_x}$만큼, Y축으로 $\mathrm{T_y}$만큼 이동하는 평행이동 행렬 $\mathrm{T(x,y)}$는 아래와 같이 나타낼 수 있습니다.

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

실제로 곱해보면 원하는 결과가 나오게 됩니다.

$$\begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} v_x + T_x \\ v_y + T_y \\ 1 \end{bmatrix}$$

만약 벡터의 Z축 값이 1이 아니면, 벡터를 Z축 값으로 나눠줍니다. 3D에서는 이를 응용하여 투영 변환 등에 활용하기도 합니다.

$$\frac{1}{v_z}\begin{bmatrix} v_x & v_y & v_z \end{bmatrix} = \begin{bmatrix} \frac{v_x}{v_z} & \frac{v_y}{v_z} & 1 \end{bmatrix}$$

이렇게 차원을 하나 늘린 좌표계를 동차 좌표계라고 합니다. 이러한 이유로 2D 그래픽스에서 3차원 행렬이 사용됩니다. 기존의 회전 변환과 척도 변환도 3차 정방행렬로 다시 나타내면.

$$S(x,y) = \begin{bmatrix} x & 0 & 0 \\ 0 & y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

행렬을 응용할 때에는 한가지 주의해야 할 점이 있습니다. 회전과 척도 변환은 순서가 바뀌어도 결과는 같았지만, 이동이 추가되면 순서에 주의해야 합니다. 회전 변환은 원점을 기준으로 하기 때문에 이동한 다음 회전하는 것과, 회전한 다음 이동하는 것의 결과는 매우 다릅니다.

그림 3의 1번 정사각형은 45도 회전한 다음 X축으로 이동했으나, 2번 정사각형은 X축으로 이동한 다음 45도 회전했습니다. 결합법칙에 의해 후위곱 연산의 경우는 나중에 곱한 행렬이 가장 먼저 계산에 적용된다는 점을 반드시 명심하고 계산해야 합니다.

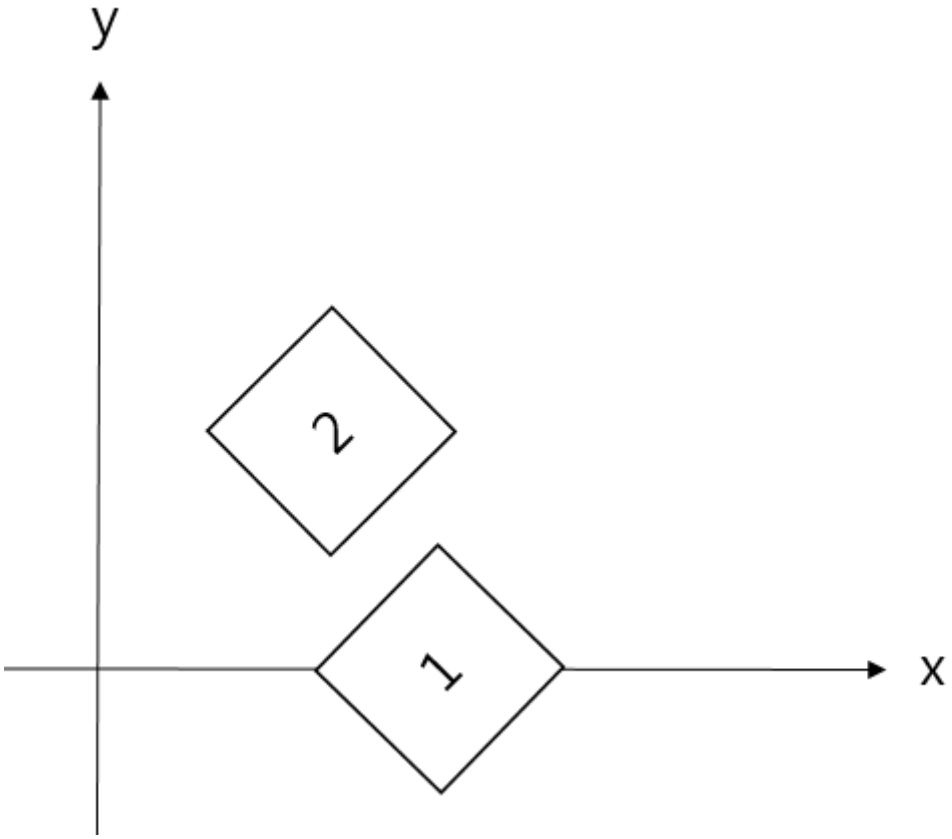

그림 3. 연산 순서에 따른 차이

$\mathrm{T} = x$축으로 1만큼 이동

$\mathrm{R} = 45°$회전

1. $\mathrm{T} \times \mathrm{R} \times v = T(R \times v)$
2. $\mathrm{R} \times \mathrm{T} \times v = R(T \times v)$

대부분의 경우 이동을 나중에 하기 때문에 후위곱을 할 때에는 1번과 같이 변환 행렬이 이동 곱하기 회전과 같은 형태가 되어야 할 것입니다. 하지만 전위곱을 할 때에는 반대라는 것 명심하셔야 합니다.

만약 전위곱을 사용하는데 여러분의 변환 행렬이 후위곱용이라면 어떻게 해야 할까요? 의외로 방법은 간단합니다 바로 전치행렬을 구해서 앞에서 곱하면 됩니다.

$$v = (x, y)$$
$$\mathrm{M} \cdot v = t$$
$$\mathrm{v} \cdot \mathrm{M}^{\mathrm{T}} = t$$

$$\begin{bmatrix} \mathrm{a} & \mathrm{b} \\ \mathrm{c} & \mathrm{d} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$
$$[x \; y] \begin{bmatrix} \mathrm{a} & \mathrm{c} \\ \mathrm{b} & \mathrm{d} \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$

마찬가지로 위에서 설명 드린 아핀변환의 행렬들도 전위곱용으로 사용하고자 한다면 그저 전치행렬로만 바꿔주면 됩니다.