Out[1]: (Show / Hide code)

CSE 4303 / CSE 5365 Computer Graphics

Brian A. Dalio, 2019 Spring

A certain amount of linear algebra is required for the study of Computer Graphics. As one of **CSE 3380** — *Linear Algebra for CSE* or **MATH 3330** — *Introduction to Matrices and Linear Algebra* is a prerequisite for CSE 4303 / CSE 5365, everything in this introduction should be familiar.

Euclidean Space

$\Re^n$ represents the real n-dimensional Euclidean space. Each element of this space is a real n-tuple, or ordered list of n real numbers. We will index the components of the element starting at 0 since that is the more common programming convention. (Be aware that many references — including *Introduction to Computer Graphics* by Foley, van Dam, Feiner, Hughes, and Phillips — follow the mathematical convention of starting indices at 1.)

$$e \in \Re^n \text{ is } \begin{bmatrix} e_0 \\ e_1 \\ \vdots \\ e_{n-1} \end{bmatrix} \text{ where } e_i \in \Re,\ 0 \leq i < n.$$

Many references call any such e a *vector*, but we will be more selective in this class and distinguish *points* from *vectors*.

Note that the n-tuple is presented as a column. This is important as columns and rows are different. If we have to write a point or vector horizontally, we will use the *transpose* operator T to indicate that the row has to be 'stood up' into a column.

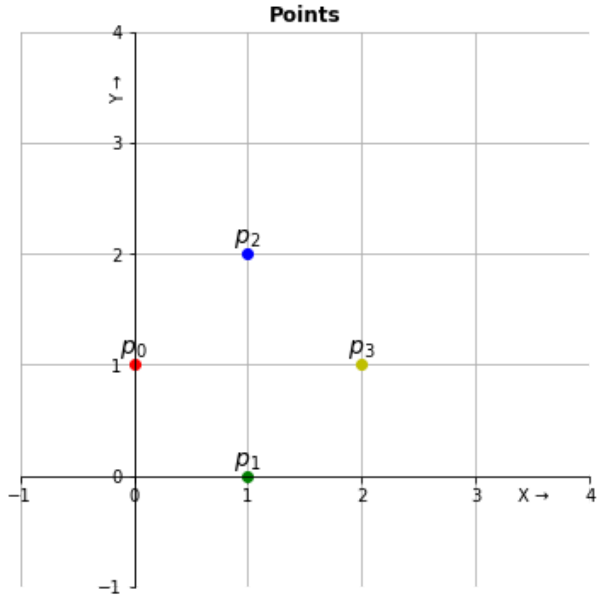
As an example, an $e \in \Re^2 = \begin{bmatrix} e_0, e_1 \end{bmatrix}^T = \begin{bmatrix} e_0 \\ e_1 \end{bmatrix}$

Points

A point $\in \Re^n$ is an n-tuple representing a location. Points have no other attributes; no length, width, depth, etc.

For example,

$$\mathbf{p}_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \mathbf{p}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{p}_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \mathbf{p}_3 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$



In the figure, the points are shown as small circles to make them visible. Actual points have no size (or color, for that matter).

Vectors

A vector $\in \Re^n$ is an n-tuple representing a direction and a magnitude. Unlike a point, a vector does *not* have a location. When we draw a vector, we start from some location and move in the vector's direction for its magnitude. However, as far as the vector itself is concerned, that start location is arbitrary.

Operations

The two simplest operations for vectors are multiplication by a scalar and addition of two vectors. Each of these operations has as its result another vector in $\Re^n$ (indicated below by the $\in \Re^n$ specification).

Scalar Multiplication

Multiplication of a vector $\mathbf{v}$ by a scalar a is defined as

$$a\,\mathbf{v} = \begin{bmatrix} a\,v_0 \\ a\,v_1 \\ \vdots \\ a\,v_{n-1} \end{bmatrix} \in \Re^n \text{ for all } \mathbf{v} \in \Re^n$$

Scalar muliplication has some interesting properties. When $a > 0$,

- The resulting vector has its *magnitude* scaled by the factor a. The orientation of the resulting vector is the same as the original vector.
- Identity: When $a = 1$ the resulting vector is the same as the original vector. 1 is therefore the *identity* element for scalar multiplication. $1 \cdot \mathbf{v} = \mathbf{v}$ for any $\mathbf{v} \in \Re^n$.

The scalar a does not have to be greater than 0. Scalar multiplication is well-defined for $a \leq 0$.

- When $a = 0$, the result is the n-component *Zero* or *Null* vector $= [0, 0, \ldots, 0]^T$, usually written $\mathbf{0}$.
- More interesting is when $a < 0$. In this case, the magnitude of the resulting vector is scaled by the factor $|a|$. Its orientation does not change, but its direction is reversed.
- Additive inverse: When $a = -1$, one writes just $-\mathbf{u}$, indicating $\mathbf{u}$'s additive inverse vector.

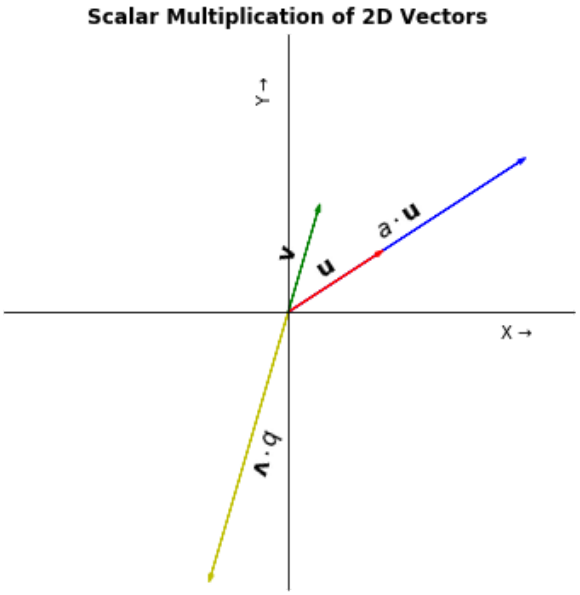
Finally, for any $a, b \in \Re$ and any $\mathbf{u} \in \Re^n$,

- $(ab)\mathbf{u} = a(b\mathbf{u})$

Example

With $\mathbf{u} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$ and $a = 2.5$, the scalar multiplication $a\,\mathbf{u} = 2.5 \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 2.5 \cdot 3 \\ 2.5 \cdot 2 \end{bmatrix} = \begin{bmatrix} 7.5 \\ 5 \end{bmatrix}$.

With $\mathbf{v} = \begin{bmatrix} 1 \\ 3.5 \end{bmatrix}$ and $b = -2.5$, the scalar multiplication $b\,\mathbf{v} = -2.5 \begin{bmatrix} 1 \\ 3.5 \end{bmatrix} = \begin{bmatrix} -2.5 \cdot 1 \\ -2.5 \cdot 3.5 \end{bmatrix} = \begin{bmatrix} -2.5 \\ -8.75 \end{bmatrix}$.



Vector Addition

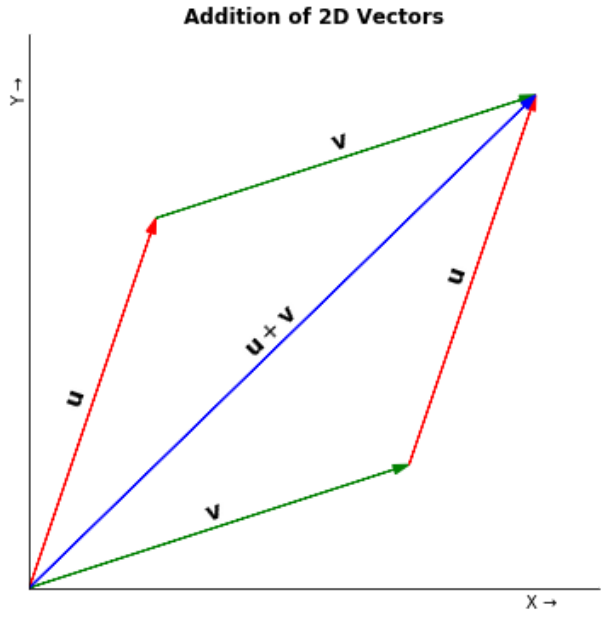
Vector addition is defined as

$$\mathbf{u} + \mathbf{v} = \begin{bmatrix} u_0 + v_0 \\ u_1 + v_1 \\ \vdots \\ u_{n-1} + v_{n-1} \end{bmatrix} \in \Re^n \text{ for all } \mathbf{u}, \mathbf{v} \in \Re^n.$$

Note that vector addition is defined only if the two vectors have the same number of components.

Example

With $\mathbf{u} = \begin{bmatrix} 2 \\ 6 \end{bmatrix}$ and $\mathbf{v} = \begin{bmatrix} 6 \\ 2 \end{bmatrix}$, the vector sum $\mathbf{u} + \mathbf{v} = \begin{bmatrix} 2 + 6 \\ 6 + 2 \end{bmatrix} = \begin{bmatrix} 8 \\ 8 \end{bmatrix}$.



Note that in the *Scalar Multiplication of 2D Vectors* example, all of the vectors were shown with their start point at the graph's origin $(0, 0)$. This was purely for convenience as vectors in and of themselves have no particular origin, being only a direction and a magnitude. This characteristic was clearly shown in the *Addition of 2D Vectors* example as both $\mathbf{u}$ and $\mathbf{v}$ are shown twice, with different origins. Each time, however, the vectors have the same direction (are *parallel*) and magnitudes.

Vector addition has some interesting properties.

- Additive Identity: $\mathbf{0}$ is the vector addition *identity* element, as $\mathbf{u} + \mathbf{0} = \mathbf{0} + \mathbf{u} = \mathbf{u}$ for any vector $\mathbf{u}$
- Commutativity: $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$. This is clearly shown in the *Addition of 2D Vectors* example.
- Associativity: $(\mathbf{u} + \mathbf{v}) + \mathbf{w} = \mathbf{u} + (\mathbf{v} + \mathbf{w})$
- Additive inverse: $\mathbf{u} + (-\mathbf{u}) = \mathbf{0}$, which can also be written $\mathbf{u} - \mathbf{u} = \mathbf{0}$

Using the definition of a vector's inverse, we can define vector *subtraction* $\mathbf{u} - \mathbf{v} = \mathbf{u} + (-\mathbf{v})$. However, be careful as vector subtraction is *neither* commutative *nor* associative.

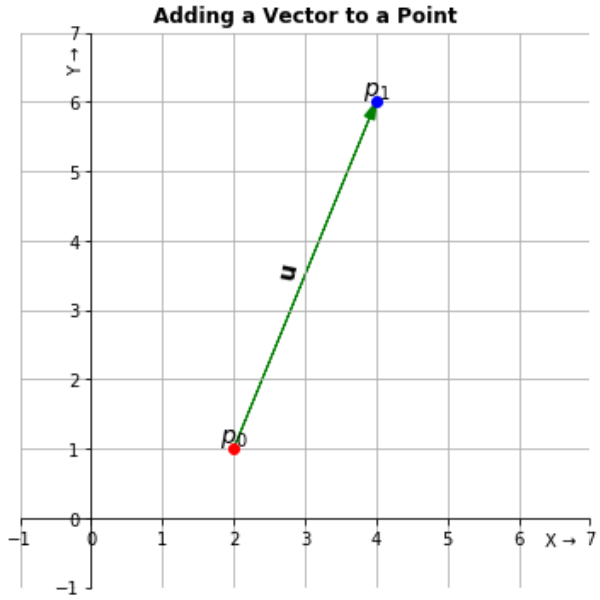
Vectors and Points

Aside from adding two vectors, it is possible to add a vector to a point. In this case, we get another point. The interpretation is that we started at the first point, went some distance in a direction, and ended up at another point.

Thus adding vector $\mathbf{u} = \begin{bmatrix} 2 \\ 5 \end{bmatrix}$ to point $\mathbf{p}_0 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ gives point $\mathbf{p}_1 = \begin{bmatrix} 4 \\ 6 \end{bmatrix}$.

Conversely, subtracting a point from another point yields a vector.

Thus $\mathbf{p}_1 - \mathbf{p}_0 = \mathbf{u}$, $\begin{bmatrix} 4 \\ 6 \end{bmatrix} - \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \end{bmatrix}$. To get from $\mathbf{p}_0$ to $\mathbf{p}_1$, we need to travel according to the direction and magnitude of vector $\mathbf{u}$.



Combining Operations

Combining scalar multiplication of a vector with vector addition yields some useful and interesting results.

- Distribution: $(a + b)\,\mathbf{u} = a\,\mathbf{u} + b\,\mathbf{u}$
- Distribution: $a\,(\mathbf{u} + \mathbf{v}) = a\,\mathbf{u} + a\,\mathbf{v}$

Linear Combination

Going a bit further with scalar multiplication of a vector and vector addition, we can define a *linear combination* $\mathbf{c}$ of m scalar factors $a_i \in \Re$ and m vectors $\mathbf{u}_i$ each $\in \Re^n$ as

$$\mathbf{c} = \sum_{i=0}^{n-1} a_i\,\mathbf{u}_i, \text{ where } \mathbf{c} \in \Re^n$$

Linear combinations of vectors are used to describe many objects and are especially useful in the representation of curves and surfaces, as we shall see later.

Demonstration

In the first following demonstration, we have $\mathbf{u}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\mathbf{u}_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

a_0 and a_1 are each adjustable in the range $[-10 \ldots 10]$.

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

In the second demonstration, let's make it a bit more interesting by rotating u_0 and u_1 by 45° counterclockwise so that they are no longer aligned with the axes. We therefore now have $\mathbf{u}_0 = \left[\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2} \right]^T$ and $\mathbf{u}_1 = \left[-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2} \right]^T$.

a_0 and a_1 are still each adjustable in the range $[-10 \ldots 10]$.

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

In either case, we can direct $\mathbf{c}$ in any direction and magnitude (within the limits of the display), but the required scalar coefficients a_0, a_1 depend on the particular values of the base vectors u_0, u_1. (Not surprising.)

Dot Product

The next operation to consider is the *dot product* of two vectors, defined as

$$\mathbf{u} \cdot \mathbf{v} = \sum_{i=0}^{n-1} u_i v_i, \text{ for } \mathbf{u}, \mathbf{v} \in \Re^n$$

In words, the dot product is the sum of multiplying the corresponding components of the two vectors. Therefore, as with vector addition, the dot product is defined only if the two vectors have the same number of components. However, unlike vector addition, the result of the dot product operation is a *scalar* not a vector.

The dot product has some useful and interesting properties.

- $\mathbf{u} \cdot \mathbf{u} \geq 0$, with $\mathbf{u} \cdot \mathbf{u} = 0$ if and only if $\mathbf{u} = \mathbf{0}$
- Commutivity: $\mathbf{u} \cdot \mathbf{v} = \mathbf{v} \cdot \mathbf{u}$
- Distribution: $\mathbf{u} \cdot (\mathbf{v} + \mathbf{w}) = \mathbf{u} \cdot \mathbf{v} + \mathbf{u} \cdot \mathbf{w}$
- $(a\mathbf{u}) \cdot (b\mathbf{v}) = ab(\mathbf{u} \cdot \mathbf{v})$

And finally a very interesting property,

- $\mathbf{u} \cdot \mathbf{v} = 0 \iff \mathbf{u} \perp \mathbf{v}$, for $\mathbf{u}, \mathbf{v} \neq \mathbf{0}$

In words, the dot product of two non-zero vectors is equal to zero if and only if the two vectors are orthogonal, $\perp$. (Technically, the vectors do not have to be non-zero as mathematicians consider the zero vector orthogonal to all vectors, including itself.)

Norm

In the 2D plane defined by $\Re^2$, the distance from any point (x, y) to the origin is $\sqrt{x^2 + y^2}$. This can be extended to n dimensions so that we have the distance of point $\mathbf{p} \in \Re^{\mathbf{n}}$ to the origin $= \sqrt{\sum_{i=0}^{n-1} p_i^2}$.

Given the definition of the dot product, it's clear that the magnitude of a vector $\mathbf{u}$ is given by $\sqrt{\mathbf{u} \cdot \mathbf{u}}$. This is known as the *norm* of the vector and is written as $\|\mathbf{u}\|$.

Some properties of the norm include,

- $\|\mathbf{u}\| = 0 \iff \mathbf{u} = \mathbf{0}$
- $\|a\mathbf{u}\| = |a|\|\mathbf{u}\|$
- Triangle inequality: $\|\mathbf{u} + \mathbf{v}\| \leq \|\mathbf{u}\| + \|\mathbf{v}\|$
- $\|\mathbf{u} + \mathbf{v}\| = \|\mathbf{u}\| + \|\mathbf{v}\| \iff \mathbf{u} \| \mathbf{v}$ for $\mathbf{u}, \mathbf{v} \neq \mathbf{0}$
- Cauchy-Schwarz inequality: $|\mathbf{u} \cdot \mathbf{v}| \leq \|\mathbf{u}\|\|\mathbf{v}\|$
- $|\mathbf{u} \cdot \mathbf{v}| = \|\mathbf{u}\|\|\mathbf{v}\| \iff \mathbf{u} \| \mathbf{v}$ for $\mathbf{u}, \mathbf{v} \neq \mathbf{0}$

For both the Triangle and Cauchy-Schwarz inequalities, the two sides are equal if and only if the (non-zero) vectors $\mathbf{u}, \mathbf{v}$ are parallel, $\|$. (Technically, the vectors do not have to be non-zero as mathematicians consider the zero vector parallel to all vectors, including itself.)

Finally, to *normalize* a non-zero vector means to scale its magnitude to 1. We compute the normalized version of a vector thusly,

$$\hat{\mathbf{u}} = \frac{\mathbf{u}}{\|\mathbf{u}\|}, \text{ for } \mathbf{u} \neq \mathbf{0}$$

Any vector with a magnitude of 1 is known as a *unit vector*. The $\hat{\mathbf{u}}$ notation means 'the unit vector in the direction of $\mathbf{u}$'.

Geometrical Interpretation of the Dot Product

As observed above, the dot product $\mathbf{u} \cdot \mathbf{v}$ is equal to 0 when the vectors $\mathbf{u}, \mathbf{v}$ are perpendicular to each other and is equal to $\|\mathbf{u}\|\|\mathbf{v}\|$ when the vectors are parallel.

If we take the dot product of the normalized versions of two vectors, we get the cosine of the angle between them.

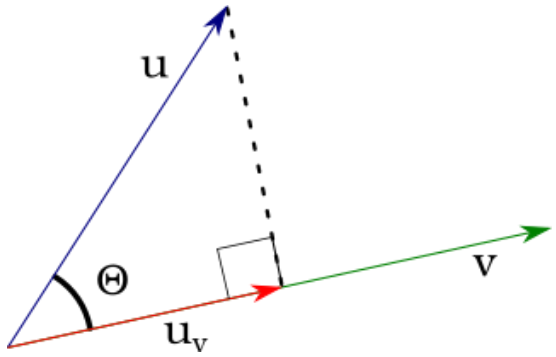
$$\cos\theta_{uv} = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}$$

We can use this property of the dot product to find the *projection* of one of the vectors onto the other. For example, to find the projection of $\mathbf{u}$ onto $\mathbf{v}$ we first compute the magnitude of the projection

$$u_v = \|\mathbf{u}\|\cos\theta_{uv} = \|\mathbf{u}\|\frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|} = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{v}\|} = \mathbf{u} \cdot \hat{\mathbf{v}}$$

Then the projected vector $\mathbf{u}_\mathbf{v} = u_v\hat{\mathbf{v}}$. Similarly, the projection of $\mathbf{v}$ onto $\mathbf{u}$ is $\mathbf{v}_\mathbf{u} = (\mathbf{v} \cdot \hat{\mathbf{u}})\hat{\mathbf{u}}$.

The following illustration shows this projection.



A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

Matrices

For us a *matrix* is a two-dimensional arrangement of numbers organized in rows and columns. A matrix $\mathbf{M}$ with p rows and q columns is called an $p \times q$ matrix and appears thus,

$$\mathbf{M} = \begin{bmatrix} m_{0,0} & m_{0,1} & \cdots & m_{0,q-1} \\ m_{1,0} & m_{1,1} & \cdots & m_{1,q-1} \\ \vdots & \vdots & \ddots & \vdots \\ m_{p-1,0} & m_{p-1,1} & \cdots & m_{p-1,q-1} \end{bmatrix} \in \Re^{p \times q}$$

As with vectors, we start the indices at 0. A vector $\vec{\mathbf{u}} \in \Re^n$ is an $n \times 1$ matrix. Mathematically, it is called a *column vector*, though we use the term vector in a more restricted sense. Similarly, an $1 \times n$ matrix is called a *row vector*.

If the number of rows of a matrix is equal to its number of columns, the matrix is known as a *square* matrix. Many matrix operations and processes are applicable only to square matrices. We will note this requirement as necessary.

As might be expected, a *zero* or *null* matrix is a matrix whose entries are all 0. We represent this matrix as $\mathbf{0}$, when its dimensions are clear from the context, or $\mathbf{0_n}$ for the square zero matrix, or $\mathbf{0_{pq}}$ when not square.

The *identity* matrix $\mathbf{I_n}$ is the $n \times n$ (and therefore *square*) matrix whose entries are

$$m_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}, \text{ for } 0 \leq i, j < n.$$

For example,

$$\mathbf{I_3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{I_4} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \cdots$$

Operations

There are a number of elementary operations on matrices, some analgous to vector operations and some a bit more complex.

Transpose

The *transpose* of the matrix $\mathbf{M} \in \Re^{p \times q}$ is the $q \times p$ matrix (notice the swapping of p and q) built by taking the rows of $\mathbf{M}$ as its columns, or equivalently the columns of $\mathbf{M}$ as its rows. As with vectors, we use T as the transpose operator and denote the transpose of $\mathbf{M}$ as $\mathbf{M^T}$. More precisely,

$$m_{ij}^T = m_{ji}, \text{ where } 0 \leq i < q,\ 0 \leq j < p$$

For example,

$$\begin{bmatrix} 0 & 1 \\ 2 & 3 \\ 4 & 5 \\ 6 & 7 \end{bmatrix}^T = \begin{bmatrix} 0 & 2 & 4 & 6 \\ 1 & 3 & 5 & 7 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix}^T = \begin{bmatrix} 0 & 3 & 6 \\ 1 & 4 & 7 \\ 2 & 5 & 8 \end{bmatrix}$$

Notice that the matrix in the second example above is a square matrix. When a square matrix is transposed, the elements on the diagonal do not change position. Also, the elements in the upper left and lower right 'triangles' are mirrored about the diagonal.

Properties of the transpose operation include

- $(\alpha\mathbf{M})^T = \alpha\mathbf{M}^T$
- $(\mathbf{M} + \mathbf{N})^T = \mathbf{M}^T + \mathbf{N}^T$
- $(\mathbf{M}^T)^T = \mathbf{M}$
- $(\mathbf{MN})^T = \mathbf{N}^T\mathbf{M}^T$

A matrix is called *symmetric* if it is equal to its transpose, $\mathbf{M} = \mathbf{M}^T$. All symmetric matrices are square.

Trace

The *trace* of a matrix is the sum of its diagonal elements. The trace is defined only for square matrices. It is denoted $\text{tr}(\mathbf{M})$ and is defined as

$$\text{tr}(\mathbf{M}) = \sum_{i=0}^{n-1} m_{ii} \text{ for } \mathbf{M} \in \Re^{n \times n}$$

For example,

$$\text{tr}\begin{pmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{pmatrix} = 12$$

Properties of the trace operation include

- $\text{tr}(\mathbf{M} + \mathbf{N}) = \text{tr}(\mathbf{M}) + \text{tr}(\mathbf{N})$
- $\text{tr}(a\mathbf{M}) = a\,\text{tr}(\mathbf{M})$
- $\text{tr}(\mathbf{M}) = \text{tr}(\mathbf{M}^T)$

The trace of the identity matrix $\mathbf{I}_n$ is n since its diagonal has n elements all of which are 1.

Scalar Multiplication

The multiplication of a matrix $\mathbf{B} \in \Re^{p \times q}$ by a scalar a is denoted $\mathbf{M} = a\,\mathbf{B}$ and is defined as

$$m_{ij} = ab_{ij}, \text{ where } 0 \le i < p,\ 0 \le j < q$$

As in the vector case, each element of the matrix is multiplied by the scalar.

For example,

$$10\begin{bmatrix} 0 & 1 \\ 2 & 3 \\ 4 & 5 \\ 6 & 7 \end{bmatrix} = \begin{bmatrix} 0 & 10 \\ 20 & 30 \\ 40 & 50 \\ 60 & 70 \end{bmatrix} \qquad -2\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix} = \begin{bmatrix} 0 & -2 & -4 \\ -6 & -8 & -10 \\ -12 & -14 & -16 \end{bmatrix}$$

Properties of scalar multiplication include some trivial observations.

- $0\,\mathbf{M} = \mathbf{0}$
- $\alpha\,\mathbf{0} = \mathbf{0}$
- $1\,\mathbf{M} = \mathbf{M}$

Analagous to the vector scalar multiplication operation, matrix scalar multiplication has these properties.

- $a(b\mathbf{M}) = (ab)\mathbf{M}$
- Distribution: $(a + b)\mathbf{M} = a\mathbf{M} + b\mathbf{M}$
- Distribution: $a(\mathbf{M} + \mathbf{N}) = a\mathbf{M} + a\mathbf{N}$

Addition

Two matrices may be added if they are the same size (have the same number of rows and columns). The addition of two matrices $\mathbf{A}, \mathbf{B} \in \Re^{p \times q}$ is denoted $\mathbf{S} = \mathbf{A} + \mathbf{B}$ and is defined as

$$s_{ij} = a_{ij} + b_{ij}, \text{ where } 0 \le i < p,\ 0 \le j < q$$

For example,

$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix} + \begin{bmatrix} 0 & -2 & -4 \\ -6 & -8 & -10 \\ -12 & -14 & -16 \end{bmatrix} \Longrightarrow \begin{bmatrix} 0 & -1 & -2 \\ -3 & -4 & -5 \\ -6 & -7 & -8 \end{bmatrix}$$

Analogous to the vector addition operation, matrix addition has these properties.

- Additive Identity: $\mathbf{M} + \mathbf{0} = \mathbf{M}$
- Commutivity: $\mathbf{M} + \mathbf{N} = \mathbf{N} + \mathbf{M}$
- Associtivity: $(\mathbf{L} + \mathbf{M}) + \mathbf{N} = \mathbf{L} + (\mathbf{M} + \mathbf{N})$
- Additive Inverse: $\mathbf{M} + (-\mathbf{M}) = \mathbf{0}$, which can also be written $\mathbf{M} - \mathbf{M} = \mathbf{0}$

We can define matrix subtraction $\mathbf{M} - \mathbf{N} = \mathbf{M} + (-\mathbf{N})$, but be careful as subtraction is neither commutative nor associative.

Multiplication

Matrix multiplication is intricate. There are requirements on the sizes of the two matrices and the process of computing the elements of the product is involved and lengthy. The straightforward algorithm is $O(n^3)$ though if the matrices are large enough, there are clever algorithms that reduce that to $O(n^{\log_2 7})$ and even lower. Here we will be concerned with very small matrices ($2 \times 2, 3 \times 3, 4 \times 4, \ldots$) so that cleverness is not required.

Two matrices may be multiplied if and only if the number of columns of the left matrix is equal to the number of rows of the right matrix. The product has the same number of rows as the left matrix and the same number of columns as the right matrix.

More precisely, for $\mathbf{A} \in \Re^{p \times q}$ and $\mathbf{B} \in \Re^{q \times r}$, the product $\mathbf{P} = \mathbf{AB}$ will be $\in \Re^{p \times r}$. The elements of $\mathbf{P}$ are

$$p_{ij} = \sum_{k=0}^{q-1} a_{ik} b_{kj} \quad \text{where } 0 \le i < p, 0 \le j < r$$

For example,

$$\begin{bmatrix} 0 & 2 & 4 \\ 1 & 3 & 5 \end{bmatrix} \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix} = \begin{bmatrix} 30 & 36 & 42 \\ 39 & 48 & 57 \end{bmatrix}$$

Out[9]:

$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} g & h \\ i & j \\ k & l \end{bmatrix} = \begin{bmatrix} ag + bi + ck & ah + bj + cl \\ dg + ei + fk & dh + ej + fl \end{bmatrix}$$

Out[10]:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{k} & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & m \\ -\sin(\theta) & 0 & \cos(\theta) & n \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & m \\ 0 & 0 & 0 & 0 \\ -\frac{\sin(\theta)}{k} & 0 & \frac{\cos(\theta)}{k} & 1 + \frac{n}{k} \end{bmatrix}$$

Out[11]:

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

Out[12]:

$$\begin{bmatrix} 1 & h_x \\ h_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} h_x y + x \\ h_y x + y \end{bmatrix}$$

Out[13]:

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x\cos(\theta) - y\sin(\theta) \\ x\sin(\theta) + y\cos(\theta) \end{bmatrix}$$

Out[14]:

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} 1 & h_x \\ h_y & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} = \begin{bmatrix} s_x\left(-h_y\sin(\theta) + \cos(\theta)\right) & s_y\left(h_x\cos(\theta) - \sin(\theta)\right) \\ s_x\left(h_y\cos(\theta) + \sin(\theta)\right) & s_y\left(h_x\sin(\theta) + \cos(\theta)\right) \end{bmatrix}$$

Out[15]:

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} 1 & h_x \\ h_y & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} s_x x\left(-h_y\sin(\theta) + \cos(\theta)\right) + s_y y\left(h_x\cos(\theta) - \sin(\theta)\right) \\ s_x x\left(h_y\cos(\theta) + \sin(\theta)\right) + s_y y\left(h_x\sin(\theta) + \cos(\theta)\right) \end{bmatrix}$$

Out[16]:

$$\begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 & t_x \\ -\sin(\theta) & \cos(\theta) & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} t_x + x\cos(\theta) + y\sin(\theta) \\ t_y - x\sin(\theta) + y\cos(\theta) \\ t_z + z \\ 1 \end{bmatrix}$$

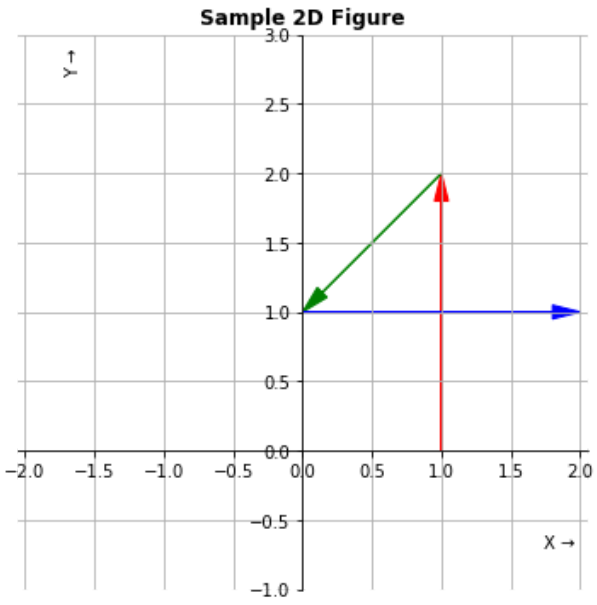
Properties of matrix multiplication include

- Associativity: $(\mathbf{LM})\mathbf{N} = \mathbf{L}(\mathbf{MN})$
- Distribution: $(\mathbf{L} + \mathbf{M})\mathbf{N} = \mathbf{LM} + \mathbf{LN}$
- Identity: $\mathbf{IM} = \mathbf{MI} = \mathbf{M}$

One property that we do *not* have for matrix multiplication is *commutivity*. In general, $\mathbf{MN} \neq \mathbf{NM}$.

2D Transformations

There are four fundamental transformations, *translation*, *scaling*, *shearing*, and *rotation*. All four can be represented by fairly simple vector and matrix operations. In the following, we will use a simple three-vector figure '4' to demonstrate the operations of the various transforms.



This figure has its origin at $\mathbf{p}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and from there three vectors are connected in series representing a non-symmetric outline of the numeral '4'. The non-symmetry makes it easier to recognize when the figure is flipped or otherwise distorted by transformations. Starting at $\mathbf{p}_0$, the rest of the points are

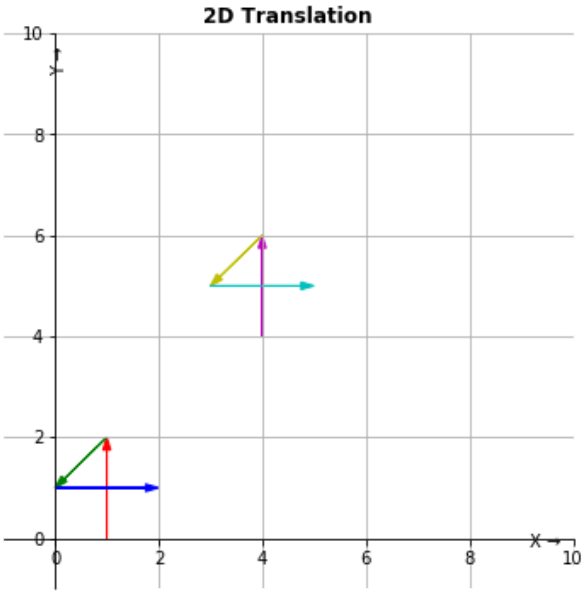
$$\mathbf{p}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \mathbf{p}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \mathbf{p}_3 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

Translation

The *translation* transform is a change from one location to another. In $\Re^2$, translation is expressed as the desired change in the x or y or both coordinates, that is $\mathbf{T}_{xy} = \left[t_x, t_y \right]^T$. This change can be added to any point or vector to translate it.

$$\mathbf{p}' = \mathbf{p} + \mathbf{T}_{xy} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \end{bmatrix}$$

For example, we can translate our simple figure by $\mathbf{T}_{xy} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$.



The translation did not alter the size or orientation of the figure. It's just been moved over to the right and up a bit.

Scaling

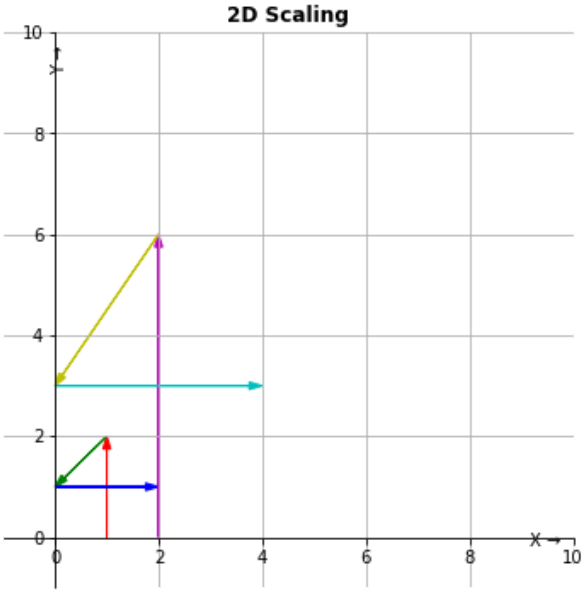
The *scaling* transform is used to stretch or shrink a point's location along the x or y axis or along both axes at the same time. The scaling transform is expressed as a matrix to be multiplied with the point or vector that's being scaled.

We represent scaling by the factor s_x in the x dimension and s_y in the y dimension with the matrix $\mathbf{S}_{xy} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$.

To scale a point $\mathbf{p}$, we compute

$$\mathbf{p}' = \mathbf{S}_{xy}\mathbf{p} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} s_x x \\ s_y y \end{bmatrix}$$

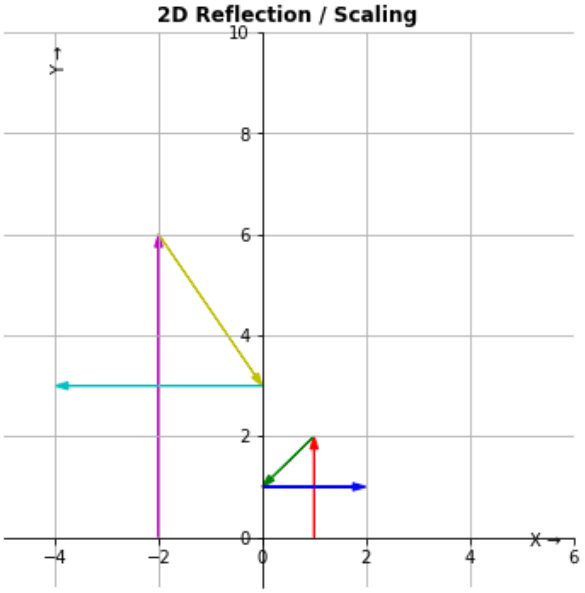
For example, we can scale our simple figure by $s_x = 2, s_y = 3$.



The orientation of the figure did not change, but it has been stretched in both the x and y dimensions.

If a scaling operation is by the same factor in both dimensions, it is said to be *uniform*, otherwise it is called *non-uniform* or *differential* (as in this case).

The stretching factor can be less than 1, in which case instead of a stretching, we have a shrinking. A negative stretching factor cause a reflection about the corresponding axis. (A zero stretching factor collapses the figure to the axis.)



Shearing

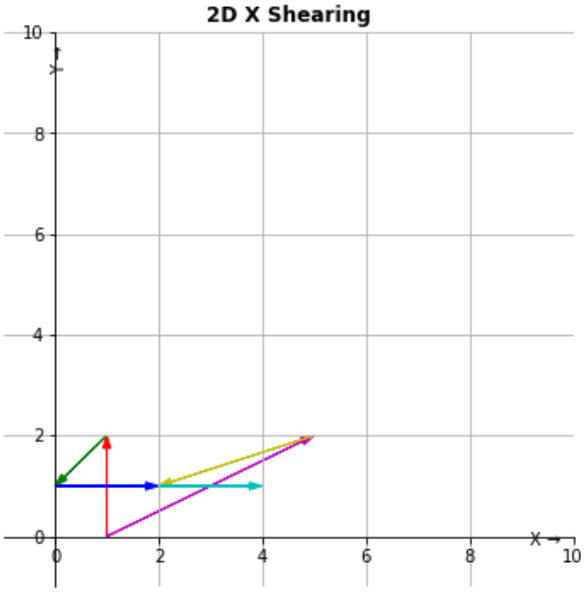
The *shearing* transform is used to proportionately change the values along one axis based on the distance along the other axis. As with the scaling transform, the shearing transform is expressed as a matrix to be multiplied with the point or vector that's being scaled.

We represent shearing by the factor h_x in the x dimension and h_y in the y dimension with the matrix $\mathbf{H}_{xy} = \begin{bmatrix} 1 & h_x \\ h_y & 1 \end{bmatrix}$.

To shear a point $\mathbf{p}$, we compute

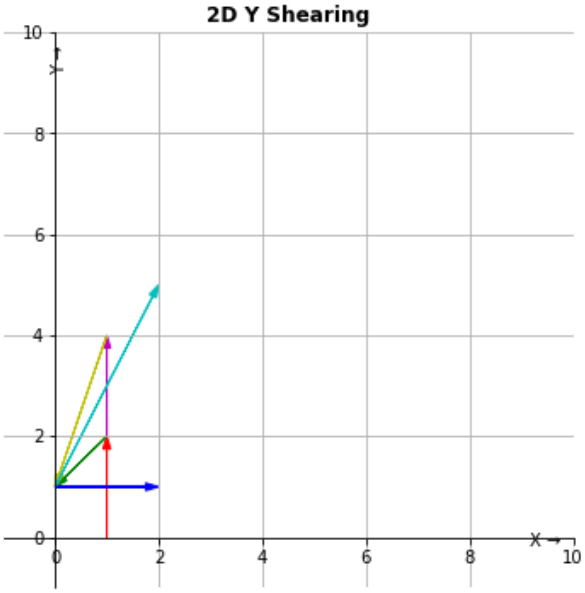
$$\mathbf{p}' = \mathbf{H}_{xy}\mathbf{p} = \begin{bmatrix} 1 & h_x \\ h_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} h_x y + x \\ h_y x + y \end{bmatrix}$$

For example, we can shear our simple figure by $h_x = 2$.



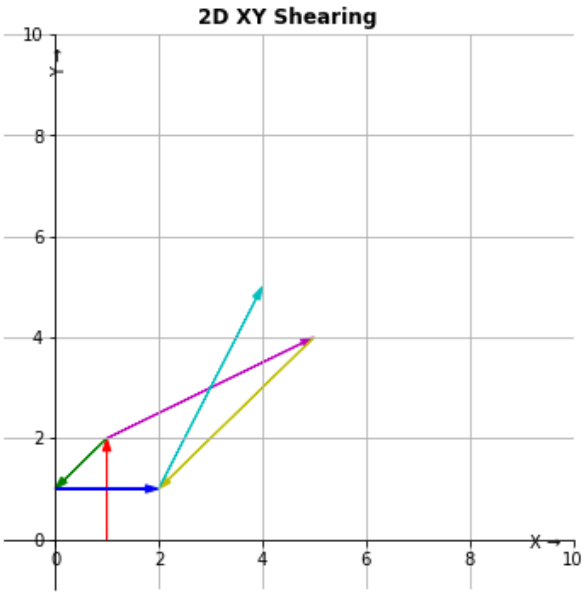
The figure is shifted to the right (along the x dimension) and the shift in x values gets larger as y increases.

A similar shearing can be done in the y dimension, $h_y = 2$.



This time the shear is along the y dimension and the shift in y values gets larger as x increases.

It's possible to shear in both dimensions simultaneously. We next transform the figure with $h_x = 2, h_y = 2$.



Now the positions are distorted in both the x and y dimensions simultaneously and the distortion increases as the points are farther along either axis.

Rotation

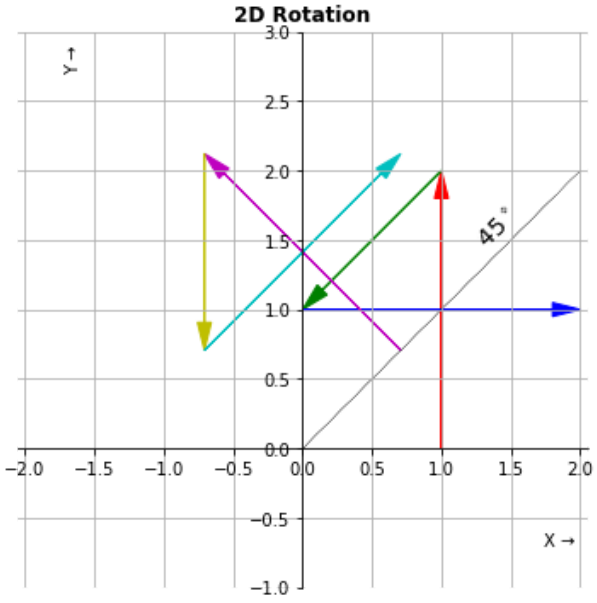
The *rotation* transfom is used to turn a location around the origin by a given angle. As with the scaling and shearing transforms, the rotation transform is expressed as a matrix to be multiplied with the point or vector that's being rotated.

We represent rotation by the angle θ with the matrix $\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$.

To rotate a point $\mathbf{p}$, we compute

$$\mathbf{p}' = \mathbf{R}_\theta \mathbf{p} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{bmatrix}$$

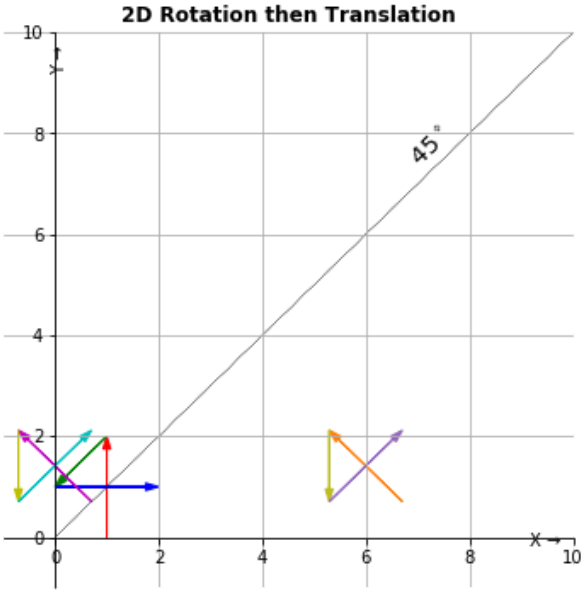
For example, we can rotate our simple figure by $\theta = 45^\circ$.



The orientation of the figure has been turned by 45° about the origin. The size and relationships between the parts of the figure have not been changed.

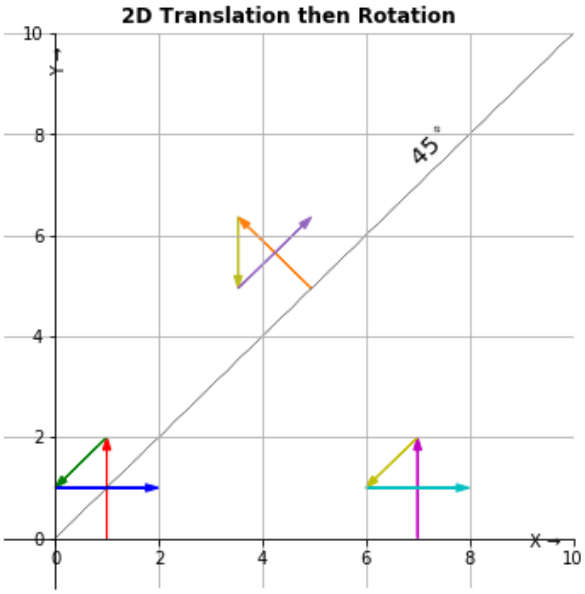
Rotation then Translation

After rotating our simple figure by 45°, we can then translate it by $t_x = 6, t_y = 0$.



As expected, the figure is turned and then moved over to the right. No big surprises here.

But suppose we did the translation first and then the rotation? What would the result be?



Doing the transformations in the opposite order yields a significantly different result. We see therefore that we can't commute translation and rotation transforms.

Composition of 2D Transforms

Given the structure of scaling, shearing, and rotation, it's easy to see that two or more of these transforms (even multiple instances of the same kind of transformation) can be combined into a single matrix.

Since transforming a point by scaling, shearing, and rotation are defined as

$$\mathbf{p}' = \mathbf{S}_{xy}\mathbf{p} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} s_x x \\ s_y y \end{bmatrix}$$

$$\mathbf{p}' = \mathbf{H}_{xy}\mathbf{p} = \begin{bmatrix} 1 & h_x \\ h_y & 1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} h_x y + x \\ h_y x + y \end{bmatrix}$$

$$\mathbf{p}' = \mathbf{R}_\theta\mathbf{p} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{bmatrix}$$

if we wanted for example to first scale a point $\mathbf{p}$ by s_x, s_y, then shear it by h_x, h_y, and finally rotate it by θ, we would just compute

$$\mathbf{p}_1 = \mathbf{S}_{xy}\mathbf{p}\mathbf{p}_2 = \mathbf{H}_{xy}\mathbf{p}_1\mathbf{p}' = \mathbf{R}_\theta\mathbf{p}_2$$

But there's no reason to break it up into multiple operations on multiple points. We can chain the matrix multiplications together and then use the associativity of matrix multiplication to get

$$\mathbf{p}' = \mathbf{R}_\theta(\mathbf{H}_{xy}(\mathbf{S}_{xy}\mathbf{p})) = (\mathbf{R}_\theta\mathbf{H}_{xy}\mathbf{S}_{xy})\mathbf{p} \quad \text{with}$$

$$\mathbf{R}_\theta\mathbf{H}_{xy}\mathbf{S}_{xy} = \mathbf{T}_{R_\theta, H_{xy}, S_{xy}} = \begin{bmatrix} s_x(-h_y\sin\theta + cos\theta) & s_y(h_x\cos\theta - \sin\theta) \\ s_x(h_y\cos\theta + sin\theta) & s_y(h_x\sin\theta + \cos\theta) \end{bmatrix}$$

By doing this composition, we have combined three transformations into one. The matrix $\mathbf{T}_{R_\theta, H_{xy}, S_{xy}}$ can be computed once and then used repeatedly to transform a series of points or vectors as long as they all needed the same scaling, shearing, and rotation (in that order).

What does not appear here is a translation. Since translation is an addition instead of a matrix multiplication, it doesn't fit in with this composition technique.

Homogeneous Coordinates

By moving from the purely Cartesian coordinates that we have used so far to *homogeneous coordinates*, we will be able to represent translation, scaling, shearing, and rotation transformations in a uniform way. (Homogeneous coordinates were introduced in 1827 by August Möbius in his work on barycentric coordinates, *Der barycentrische Calcül*.).

In homogeneous coordinates, an additional component is added to the representation of points and vectors, w. Thus a 2D point now has three components, x, y, w. We set $w = 1$ when constructing the representation of a point. So the point $\mathbf{p} = \begin{bmatrix} x & y \end{bmatrix}^T$ is $\begin{bmatrix} x & y & 1 \end{bmatrix}^T$ in homogeneous coordinates.

Translation

Translation was defined as an addition and not as a matrix multiplication.

$$\mathbf{p}' = \mathbf{p} + \mathbf{T}_{xy} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \end{bmatrix}$$

By moving to homogeneous coordinates, we can represent a translation t_x, t_y as the matrix $\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$.

Multiplying that transform matrix by the augmented point, we find

$$\mathbf{p}' = \mathbf{T}^h_{xy}\mathbf{p} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + 0 + t_x \\ 0 + y + t_y \\ 0 + 0 + 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$$

which is the properly translated point, also in homogeneous coordinates.

Scaling

Previously we performed scaling thus

$$\mathbf{p}' = \mathbf{S}_{xy}\mathbf{p} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} s_x x \\ s_y y \end{bmatrix}$$

In homogeneous coordinates, this becomes

$$\mathbf{p}' = \mathbf{S}^h_{xy}\mathbf{p} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x x + 0 + 0 \\ 0 + s_y y + 0 \\ 0 + 0 + 1 \end{bmatrix} = \begin{bmatrix} s_x x \\ s_y y \\ 1 \end{bmatrix}$$

Again, we see that the point is properly transformed.

Shearing

Shearing was defined as

$$\mathbf{p}' = \mathbf{H}_{xy}\mathbf{p} = \begin{bmatrix} 1 & h_x \\ h_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} h_x y + x \\ h_y x + y \end{bmatrix}$$

This becomes

$$\mathbf{p}' = \mathbf{H}^h_{xy}\mathbf{p} = \begin{bmatrix} 1 & h_x & 0 \\ h_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + h_x y + 0 \\ h_y x + y + 0 \\ 0 + 0 + 1 \end{bmatrix} = \begin{bmatrix} h_x y + x \\ h_y x + y \\ 1 \end{bmatrix}$$

Again, properly transformed.

Rotation

Rotation was defined as

$$\mathbf{p}' = \mathbf{R}_\theta\mathbf{p} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{bmatrix}$$

In homogeneous coordinates, this becomes

$$\mathbf{p}' = \mathbf{R}^h_\theta\mathbf{p} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x\cos\theta - y\sin\theta + 0 \\ x\sin\theta + y\cos\theta + 0 \\ 0 + 0 + 1 \end{bmatrix} = \begin{bmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \\ 1 \end{bmatrix}$$

Properly transformed, as expected.

Observations

In the homogeneous coordinate representation, all four of the elementary transformations are easily represented as a matrix. As such, it's now possible to make arbitrary compositions of these transforms and reduce any combination and sequence of them into a single transformation matrix.

You may have noticed that the w component is always 1 in each of the original points, transformation matrices, and transformed points. This is true because each of these is an *affine* transformation.

Affine transformations ...

- Preserve points and straight lines.
- Preserve the parallelism of lines.
- Do not preserve angles.
- Do not preserve lengths, but preserve the ratios of distances between points on a straight line.

Perspective projections are not affine and can result in $w \neq 1$. We will see how to deal with that later in the course.

Extension to 3D Transformations

The two-dimensional homogeneous transformations are easily extended to three dimensional homogeneous transformations.

As in the 2D case, an additional component is added to the representation of points and vectors, w. Thus a 3D point now has four components, x, y, z, w. We set $w = 1$ when constructing the representation of a point. So the point $\mathbf{p} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ is $\begin{bmatrix} x & y & z & 1 \end{bmatrix}^T$ in homogeneous coordinates.

Translation

In homogeneous coordinates, the three-dimensional translation matrix is $\mathbf{I}_4$ (the 4×4 identity matrix) with the three required translations t_x, t_y, t_z placed in the fourth column. We find the transformed point $\mathbf{p}'$ by multiplying the transformation matrix $\mathbf{T}^h_{xyz}$ and the original point $\mathbf{p}$. The transformation matrix is on the left, the original point on the right.

Out[28]:

$$\mathbf{p}' = \mathbf{T}^h_{xyz}\mathbf{p} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} t_x + x \\ t_y + y \\ t_z + z \\ 1 \end{bmatrix}$$

The matrix multiplication process does the required adding of the proper translation amounts to the corresponding coordinates. Translation is an affine transformation so the w element is 1 for the original point, the transformation matrix, and the resulting transformed point.

Scaling

The three-dimensional scaling matrix for homogeneous coordinates is $\mathbf{0}_4$ (the 4×4 zero matrix) with the three required scaling factors s_x, s_y, s_z placed along the main diagonal and with $w = 1$. We find the transformed point $\mathbf{p}'$ by multiplying the transformation matrix $\mathbf{S}^h_{xyz}$ and the original point $\mathbf{p}$. The transformation matrix is on the left, the original point on the right.

Out[29]:

$$\mathbf{p}' = \mathbf{S}^h_{xyz}\mathbf{p} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} s_x x \\ s_y y \\ s_z z \\ 1 \end{bmatrix}$$

The matrix multiplication process does the required scaling of the point's coordinates according to the corresponding scaling factor. Scaling is an affine transformation so the w element is 1 for the original point, the transformation matrix, and the resulting transformed point.

Shearing

Moving shearing into three dimensions is a bit more complex than what was required for translation or rotation. Since there are now three axes, we have a choice not only of which axis is to be sheared, but also which axis it will be sheared with respect to.

Each of the three axes x, y, z may be sheared with respect to two alternative axes, giving six possibilities in all. For example, h_x^y means *shear along the x axis with respect to the y axis*. The five other possibilities are h_x^z, h_y^x, h_y^z, h_z^x, and h_z^y.

The three-dimensional homogeneous shearing matrix is $\mathbf{I}_4$ (the 4×4 identity matrix) with the six required shearing factors h_x^y, h_x^z, h_y^x, h_y^z, h_z^x, h_z^y placed in the upper and lower triangle positions of the upper-left 3×3 submatrix. The shearing factors for the x axis are in the first row, those for the y axis are in the second row, and those for the z axis are in the third row. The shearing factors with respect to the x axis are in the first column, those with respect to the y axis in the second column, and those with respect to the z axis in the third column.

We find the transformed point $\mathbf{p}'$ by multiplying the transformation matrix $\mathbf{H}_{xyz}^h$ and the original point $\mathbf{p}$. The transformation matrix is on the left, the original point on the right.

The matrix is as follows.

Out[30]:

$$\mathbf{p}' = \mathbf{H}_{xyz}^h \mathbf{p} = \begin{bmatrix} 1 & h_x^y & h_x^z & 0 \\ h_y^x & 1 & h_y^z & 0 \\ h_z^x & h_z^y & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} h_x^y y + h_x^z z + x \\ h_y^x x + h_y^z z + y \\ h_z^x x + h_z^y y + z \\ 1 \end{bmatrix}$$

The matrix multiplication process does the required shearing of the point's coordinates along each of the axes and with respect to the other two axes according to the corresponding shearing factor. Shearing is an affine transformation so the w element is 1 for the original point, the transformation matrix, and the resulting transformed point.

Rotation

As with shearing, moving rotation into three dimensions requires some additional complexity. We now have three separate axes around which the rotation might occur so there are three possible rotation matrices, $\mathbf{R}_x^h$, $\mathbf{R}_y^h$, and $\mathbf{R}_z^h$.

A three-dimensional homogeneous rotation matrix is formed by starting with $\mathbf{I}_4$, the 4×4 identity matrix, and then placing two cosine and two sine functions of the rotation angle in the matrix in positions depending on which axis is being rotated about.

- x axis rotation: the functions are placed in the second and third rows and second and third columns. The first row and first column (corresponding to x) are left as they are in the identity matrix.
- y axis rotation: the functions are placed in the first and third rows and first and third columns. The second row and second column (corresponding to y) are left as they are in the identity matrix.
- z axis rotation: the functions are placed in the first and second rows and first and second columns. The third row and third column (corresponding to z) are left as they are in the identity matrix.

We find the transformed point $\mathbf{p}'$ by multiplying the desired transformation matrix $\mathbf{R}_x^h$, $\mathbf{R}_y^h$, or $\mathbf{R}_z^h$ and the original point $\mathbf{p}$. The transformation matrix is on the left, the original point on the right.

Out[31]:

Rotation

$$\mathbf{p}' = \mathbf{R}_x^h \mathbf{p} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta_x) & -\sin(\theta_x) & 0 \\ 0 & \sin(\theta_x) & \cos(\theta_x) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y\cos(\theta_x) - z\sin(\theta_x) \\ y\sin(\theta_x) + z\cos(\theta_x) \\ 1 \end{bmatrix}$$

$$\mathbf{p}' = \mathbf{R}_y^h \mathbf{p} = \begin{bmatrix} \cos(\theta_y) & 0 & \sin(\theta_y) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta_y) & 0 & \cos(\theta_y) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x\cos(\theta_y) + z\sin(\theta_y) \\ y \\ -x\sin(\theta_y) + z\cos(\theta_y) \\ 1 \end{bmatrix}$$

$$\mathbf{p}' = \mathbf{R}_z^h \mathbf{p} = \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x\cos(\theta_z) - y\sin(\theta_z) \\ x\sin(\theta_z) + y\cos(\theta_z) \\ z \\ 1 \end{bmatrix}$$

The matrix multiplication process does the required rotation of the point's coordinates about the corresponding axis *with respect to the origin* and for the given angle θ. Rotation is an affine transformation so the w element is 1 for the original point, the transformation matrix, and the resulting transformed point.

Composition of Transformations

As seen in the two-dimensional case, a series of three-dimensional transformations may be combined to reduce the cost of applying the same transformation series to many different points.

For example, suppose a point $\mathbf{p}$ needs to be translated by f_x, f_y, f_z, then rotated by θ about the z axis, and finally translated by g_x, g_y, g_z. That transformation would be computed as:

Out[32]:

$$\mathbf{p}' = \mathbf{T}_g \mathbf{R}_\theta^z \mathbf{T}_f \mathbf{p}$$

$$= \begin{bmatrix} 1 & 0 & 0 & g_x \\ 0 & 1 & 0 & g_y \\ 0 & 0 & 1 & g_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & f_x \\ 0 & 1 & 0 & f_y \\ 0 & 0 & 1 & f_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 & f_x\cos(\theta_z) - f_y\sin(\theta_z) + g_x \\ \sin(\theta_z) & \cos(\theta_z) & 0 & f_x\sin(\theta_z) + f_y\cos(\theta_z) + g_y \\ 0 & 0 & 1 & f_z + g_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} f_x\cos(\theta_z) - f_y\sin(\theta_z) + g_x + x\cos(\theta_z) - y\sin(\theta_z) \\ f_x\sin(\theta_z) + f_y\cos(\theta_z) + g_y + x\sin(\theta_z) + y\cos(\theta_z) \\ f_z + g_z + z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} g_x + (f_x + x)\cos(\theta_z) - (f_y + y)\sin(\theta_z) \\ g_y + (f_x + x)\sin(\theta_z) + (f_y + y)\cos(\theta_z) \\ f_z + g_z + z \\ 1 \end{bmatrix}$$

Since the rotation in this combined transformation is about the z axis, the rotation does not affect the z element of the point. (On the other hand, the two translations do affect the z element.)

In the special case where the translation g is the inverse of the translation f (that is, $g = -f$), the transformation can be simplified somewhat.

$$\mathbf{p}' = \mathbf{T}_f^{-1} \mathbf{R}_\theta^z \mathbf{T}_f \mathbf{p}$$

$$= \begin{bmatrix} 1 & 0 & 0 & -f_x \\ 0 & 1 & 0 & -f_y \\ 0 & 0 & 1 & -f_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & f_x \\ 0 & 1 & 0 & f_y \\ 0 & 0 & 1 & f_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 & f_x\cos(\theta_z) - f_x - f_y\sin(\theta_z) \\ \sin(\theta_z) & \cos(\theta_z) & 0 & f_x\sin(\theta_z) + f_y\cos(\theta_z) - f_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} f_x\cos(\theta_z) - f_x - f_y\sin(\theta_z) + x\cos(\theta_z) - y\sin(\theta_z) \\ f_x\sin(\theta_z) + f_y\cos(\theta_z) - f_y + x\sin(\theta_z) + y\cos(\theta_z) \\ z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} -f_x + (f_x + x)\cos(\theta_z) - (f_y + y)\sin(\theta_z) \\ -f_y + (f_x + x)\sin(\theta_z) + (f_y + y)\cos(\theta_z) \\ z \\ 1 \end{bmatrix}$$

Since $g = -f$, they cancel out for the translation of z element, which is now not affected at all by this combined transformation.

Each of these transformations is an affine transformation and so is the composition of all three of them. Therefore, the w element is 1 for the original point, all of the transformation matrices, and the resulting transformed point.

Out[33]:

Efficiency of Transformations

As mentioned above, once the transformation is computed, it can then be applied to any number of points without having to be recomputed. The associativity of matrix multiplication ensures that the same answer is obtained no matter which grouping is used for the multiply.

Original transformation

$$(\mathbf{T}_f^{-1})(\mathbf{R}_\theta^z)(\mathbf{T}_f) = \begin{bmatrix} 1 & 0 & 0 & -f_x \\ 0 & 1 & 0 & -f_y \\ 0 & 0 & 1 & -f_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & f_x \\ 0 & 1 & 0 & f_y \\ 0 & 0 & 1 & f_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Combined transformation

$$(\mathbf{T}_f^{-1}\mathbf{R}_\theta^z\mathbf{T}_f) = \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 & f_x\cos(\theta_z) - f_x - f_y\sin(\theta_z) \\ \sin(\theta_z) & \cos(\theta_z) & 0 & f_x\sin(\theta_z) + f_y\cos(\theta_z) - f_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In this case, $\mathbf{T}_f^{-1}\mathbf{R}_\theta^z\mathbf{T}_f$ can be computed to obtain a single 4×4 matrix which is then used to transform any number of points. Two $4 \times 4 \cdot 4 \times 4$ matrix multiplies and one $4 \times 4 \cdot 4 \times 1$ matrix multiply, costing 80 multiplies in total, is replaced with a single $4 \times 4 \cdot 4 \times 1$ matrix multiply, costing 16 multiplies, or only 20% of the original cost.

To take a specific numeric example, let $\theta = 30\,^\circ$ and $f = \begin{bmatrix} 2 & 3 & 4 \end{bmatrix}^T$. We then find (rounded to three digits):

Original transformation

$$(\mathbf{T}_f^{-1})(\mathbf{R}_\theta^z)(\mathbf{T}_f) = \begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & -4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.866 & -0.5 & 0.0 & 0.0 \\ 0.5 & 0.866 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Combined transformation

$$(\mathbf{T}_f^{-1}\mathbf{R}_\theta^z\mathbf{T}_f) = \begin{bmatrix} 0.866 & -0.5 & 0 & -1.768 \\ 0.5 & 0.866 & 0 & 0.598 \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 1.0 \end{bmatrix}$$

Out[34]:

Order of Transformations

The composition of transformations is order dependent as matrix multiplication is *not* commutative. This can be shown by simply reversing the order of the transformations in the previous example. Instead of computing $\mathbf{T}_f^{-1}\mathbf{R}_\theta^z\mathbf{T}_f$, if we compute $\mathbf{T}_f\mathbf{R}_\theta^z\mathbf{T}_f^{-1}$ we see that a different result is obtained.

Original transformation

$$\mathbf{p}' = \mathbf{T}_f^{-1}\mathbf{R}_\theta^z\mathbf{T}_f\mathbf{p} = \begin{bmatrix} 1 & 0 & 0 & -f_x \\ 0 & 1 & 0 & -f_y \\ 0 & 0 & 1 & -f_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & f_x \\ 0 & 1 & 0 & f_y \\ 0 & 0 & 1 & f_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 & f_x\cos(\theta \\ \sin(\theta_z) & \cos(\theta_z) & 0 & f_x\sin(\theta \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & \end{bmatrix}$$

Reversed transformation

$$\mathbf{p}' = \mathbf{T}_f\mathbf{R}_\theta^z\mathbf{T}_f^{-1}\mathbf{p} = \begin{bmatrix} 1 & 0 & 0 & f_x \\ 0 & 1 & 0 & f_y \\ 0 & 0 & 1 & f_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -f_x \\ 0 & 1 & 0 & -f_y \\ 0 & 0 & 1 & -f_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 & -f_x\cos(\\ \sin(\theta_z) & \cos(\theta_z) & 0 & -f_x\sin(\\ 0 & 0 & 1 & \\ 0 & 0 & 0 & \end{bmatrix}$$

The cancellation of the effects on the z element still occur since the rotation about the z axis did not affect the z element and ordinary scalar addition *is* commutative, but the aspects of the combined transformation that affected the x and y elements is quite different.

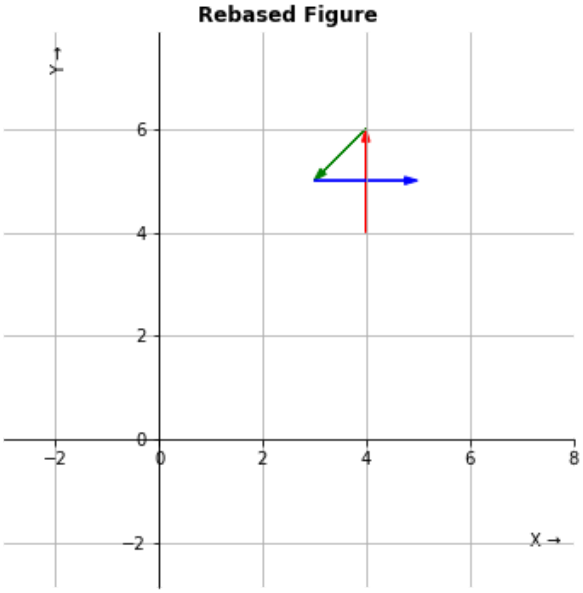
Geometric Meaning of Combined Transformations

Any combination of any number of translation, scaling, shearing, and rotation transformations may be combined into a single transformation matrix. The geomerical meaning of the individual transformations is normally quite clear, but what about the geometrical meaning of combined transformations?

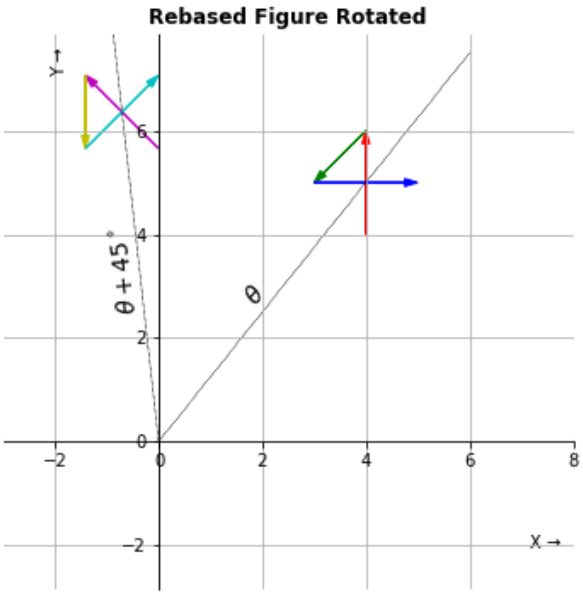
Given the infinite number of possible transformations, it is not possible to make a definitive geomerical statement about all of them, but certain combinations are amenable to intuitive geometrical explanation.

Remember that all of the four elementary transformations perform their transformation about the origin. If we want a transformation about some other point, we have to perform a combined transformation.

Consider our original, asymmetric two-dimensional figure, here shown based at $\begin{bmatrix} 3 \\ 4 \end{bmatrix}$ instead of the origin.

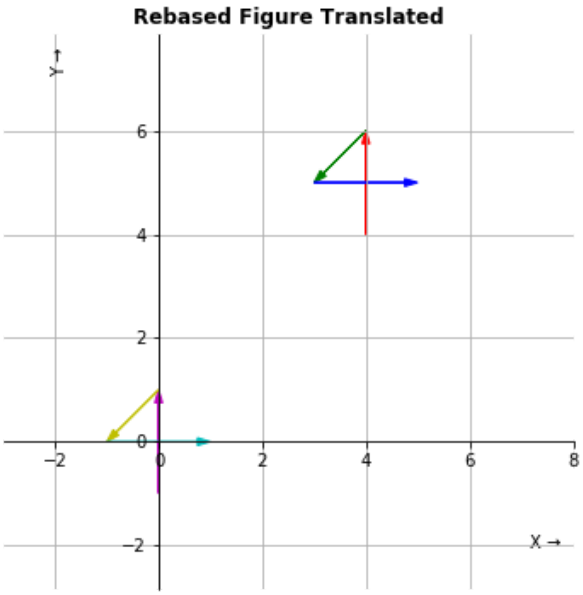


If we now rotate the figure 45°, we obtain the following.

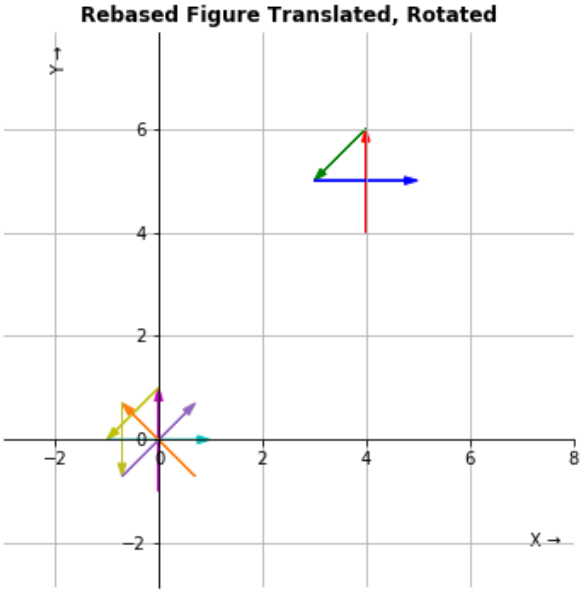


While it is indisputable that the figure has been rotated 45°, the rotation was with respect to the origin, and probably not what was wanted or expected. Intuitively, one thinks of rotation as being with respect to the center of the object being rotated.

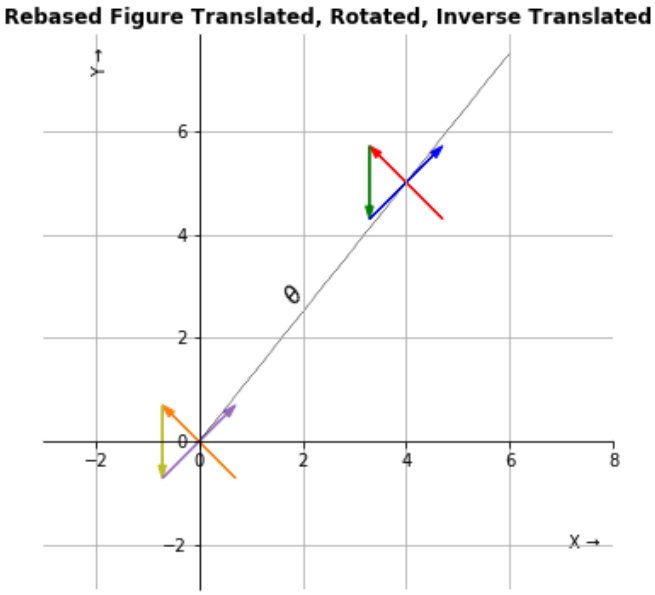
If we want our figure to be rotated about its own center, we must ensure that the center of the object is at the origin when the object is rotated. That's easy enough to do, as we can see in the following.



A translation has been used to move the center of the object to the origin. We can now rotate the object the desired 45° and the rotation will be with respect to the center of the object.



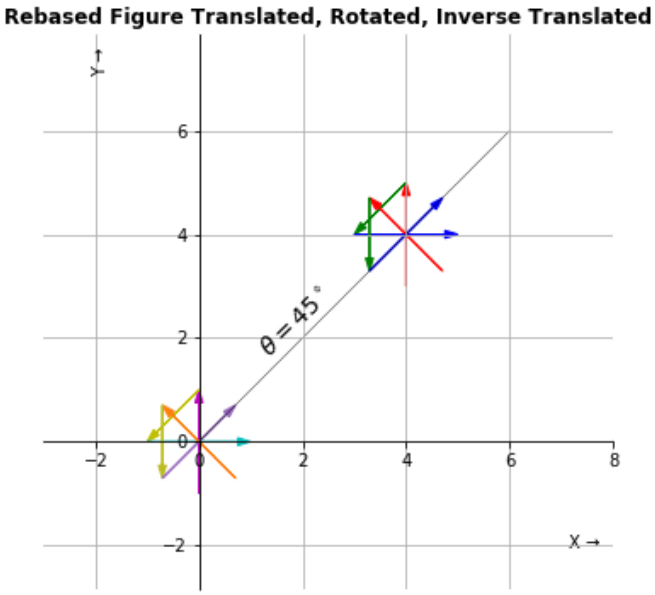
Now that the figure has been rotated about its center for 45°, all that remains is to move it back to its original position. We do this with an inverse translation, that is, a translation that is the negative of the translation that was used to move the object to the origin.



Out[40]: The figure is now at its original location and is rotated 45° about its own center, as expected.

The diagram above shows that the horizontal bar of the figure is not precisely aligned with the line even though the figure was rotated 45°. This is because the original angle $\theta \neq 45^\circ$. The center of the figure was at point $\begin{bmatrix} 4 & 5 \end{bmatrix}^T$, which makes an angle of about 51.340° with the x axis.

If we want an angle of precisely 45°, we can position the figure so that its center is at $\begin{bmatrix} 4 & 4 \end{bmatrix}^T$ instead, as in the following diagram. In this case, the horizontal bar of the figure will align precisely with an angle of 45° from the x axis.



This plot is a bit busy, but shows the four phases of the rotation of the figure.

1. In the original position.
2. Center translated to the origin.
3. Rotated at the origin (and therefore about its own center).
4. Center translated to the original position.

This technique of moving an object to the origin, performing a transformation, and then returning it to its original position can be used with any of the elementary transformations.