Out[1]: (Show / Hide code)

Basic Transforms

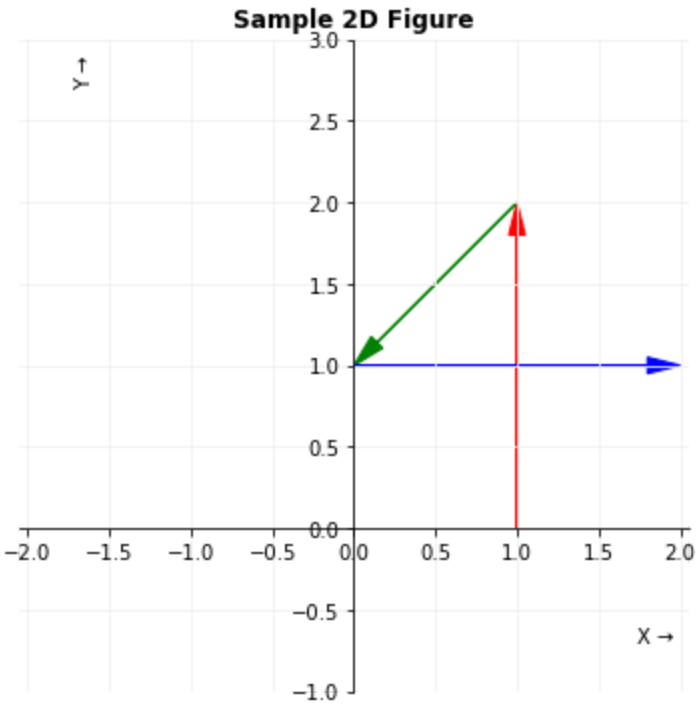
CSE 4303 / CSE 5365 Computer Graphics

2020 Fall Semester, Version 1.6, 2020 October 22

Four fundamental transforms underlie computer graphics. These transforms can be made in any number of dimensions, but in computer graphics we rarely have to consider any space other than two or three dimensions.

2D Transforms

The four fundamental transforms, *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 transforms. 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

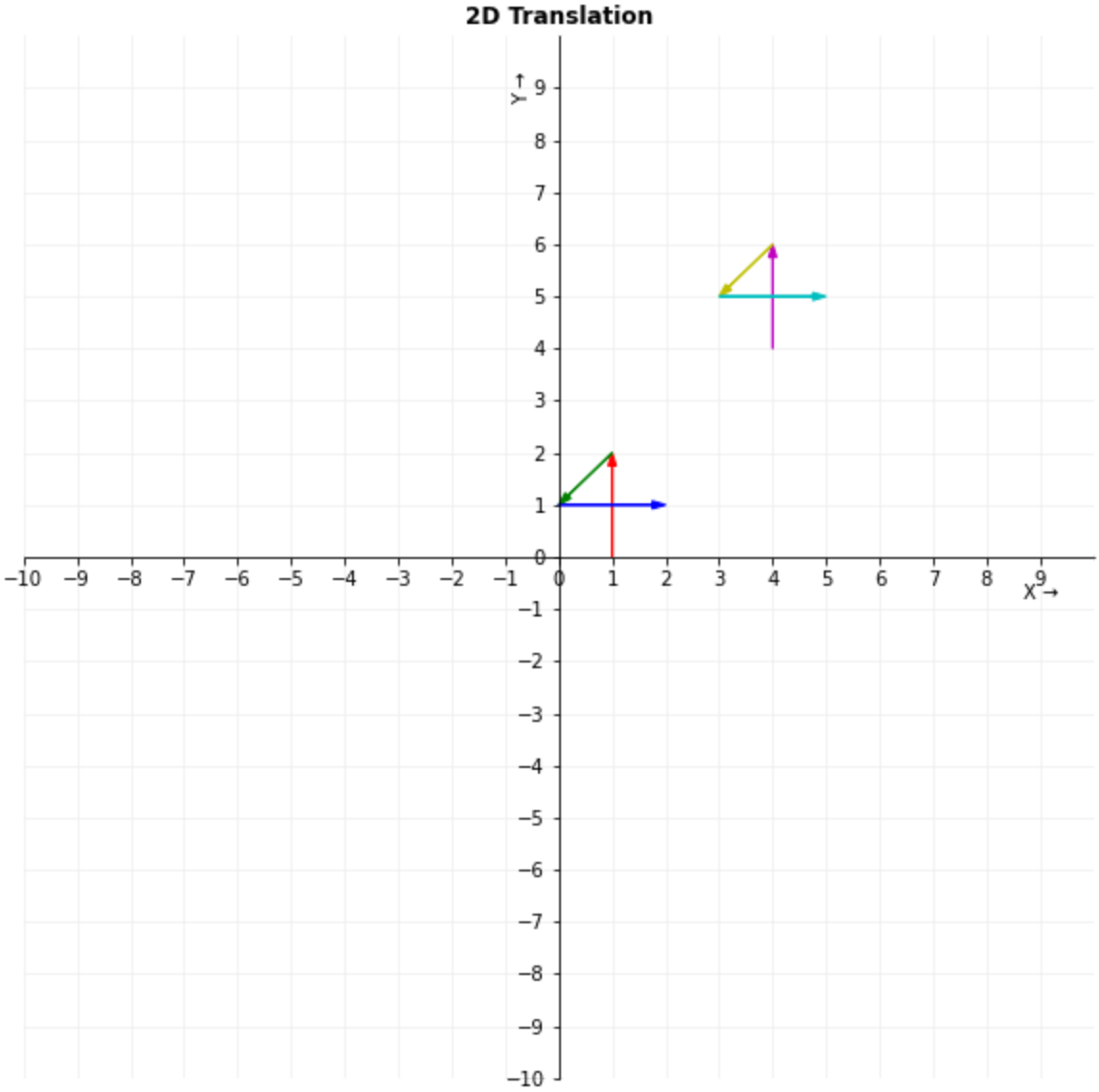
The *translation* transform is a change from one location to another. In $\mathbb{R}^2$, translation is expressed as the desired change in the x or y or both coordinates, that is $\mathbf{T}_{xy} = \begin{bmatrix} t_x, t_y \end{bmatrix}^T$. This change can be added to any point or vector to translate 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 $\mathbf{T}_{xy} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$.

tx 3.0

ty 4.0


2D Translation
Y
X
9
8
7
6
5
4
3
2
1
0
−1
−2
−3
−4
−5
−6
−7
−8
−9
−10
−10 −9 −8 −7 −6 −5 −4 −3 −2 −1 0 1 2 3 4 5 6 7 8 9

The translation does not alter the size or orientation of the figure. It just gets moved around.

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 $\mathbf{S}_{xy} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$.

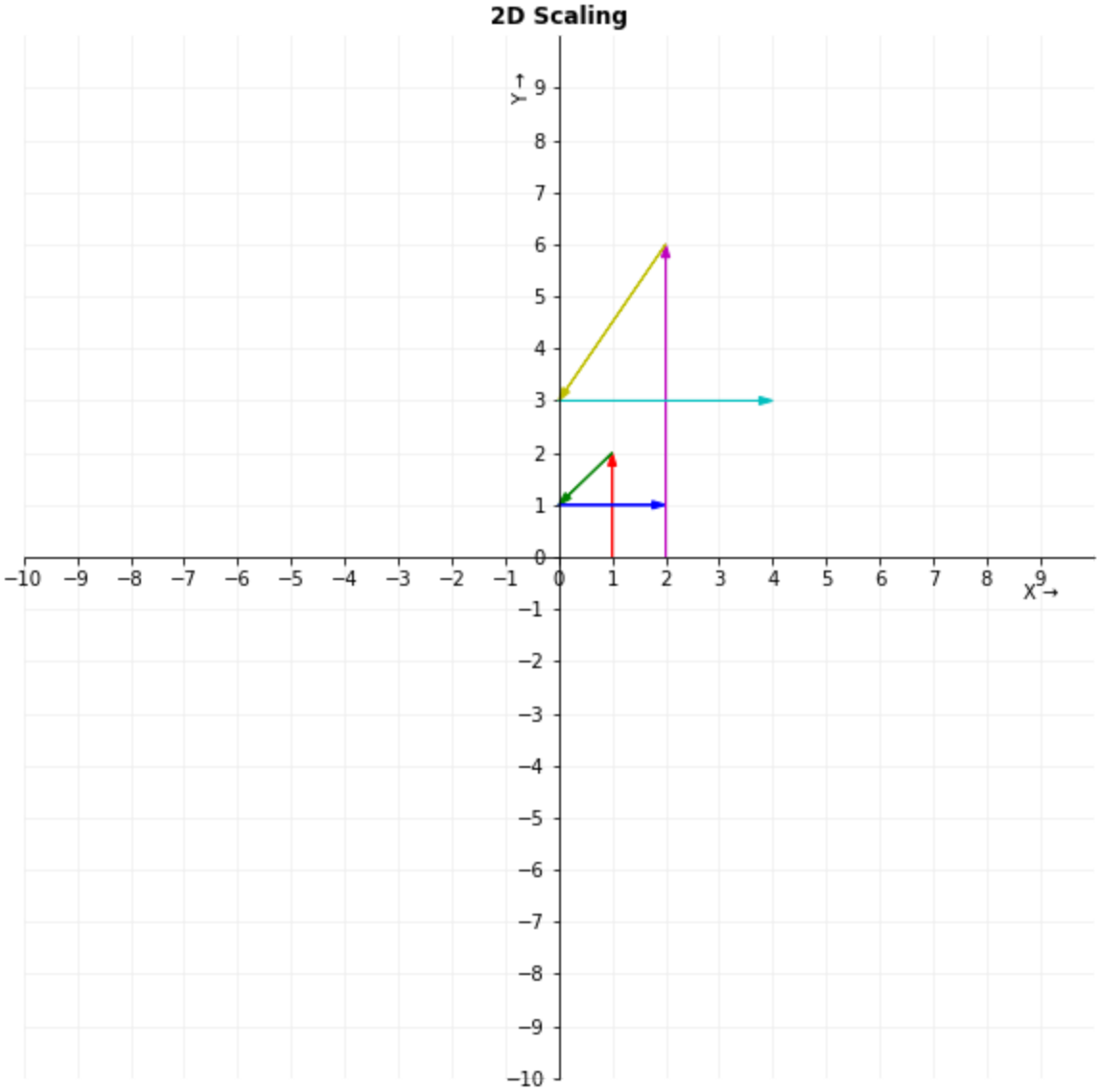
To scale a point $\mathbf{p}$, we compute

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

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

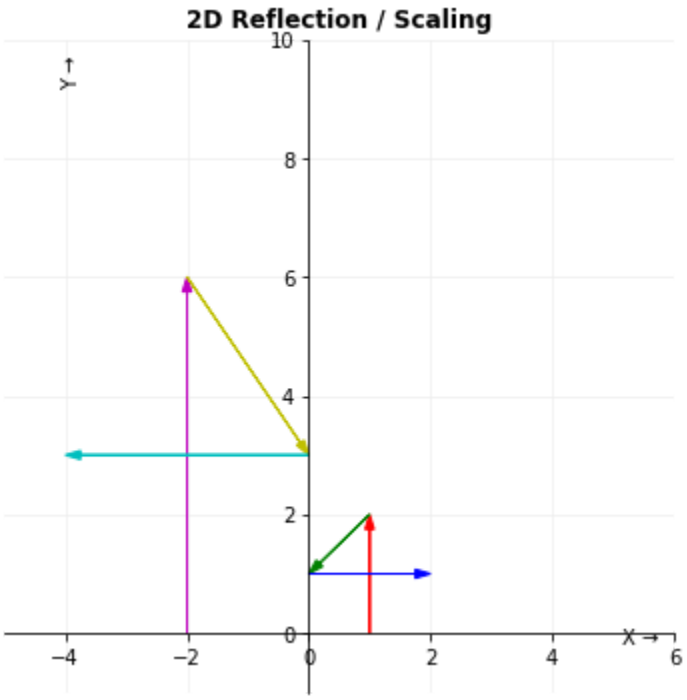
sx 2.0

sy 3.0


2D Scaling
Y
X
9
8
7
6
5
4
3
2
1
0
−1
−2
−3
−4
−5
−6
−7
−8
−9
−10
−10 −9 −8 −7 −6 −5 −4 −3 −2 −1 0 1 2 3 4 5 6 7 8 9

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).

For a stretching factor in the range $0 < s < 1$, we have a *shrinking* instead of a *stretching*. A stretching factor < 0 causes a reflection about the corresponding axis. The degenerate case of a stretching factor $= 0$ collapses the figure to the corresponding axis.



In this example, we have scaled our simple figure by $s_x = -2, s_y = 3$, resulting in a *reflection* in x (which is said to be *about the* y *axis*).

Shearing

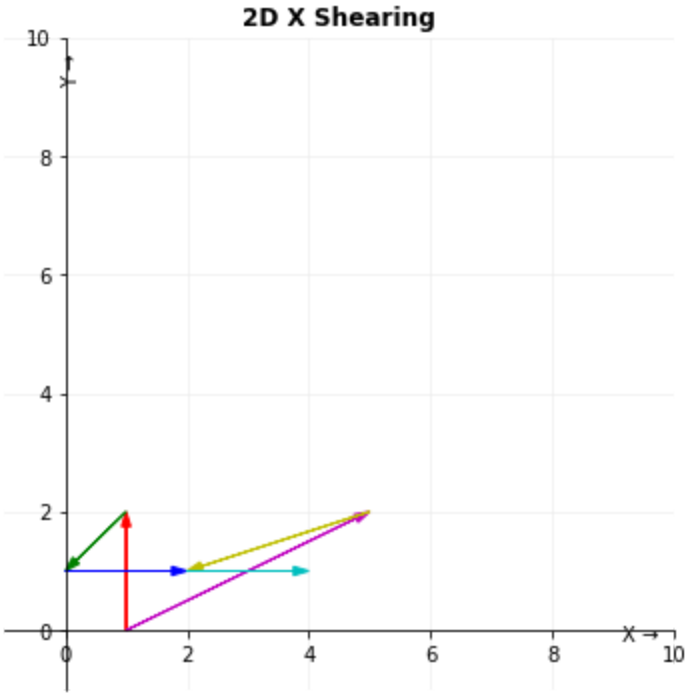
The *shearing* transform is used to proportionately change the values along one axis based on the distance along the other axis. As with the scaling transform, the shearing transform is expressed as a matrix to be multiplied with the point or vector that is being sheared.

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

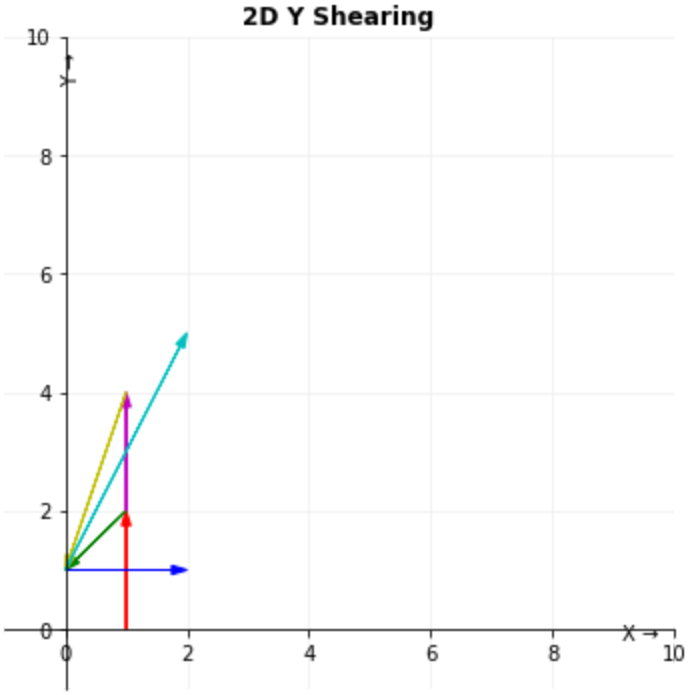
$$\mathbf{p}' = \mathbf{H}_{xy}\mathbf{p} = \begin{bmatrix} 1 & h_x \\ h_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} h_x y + x \\ h_y x + y \end{bmatrix}$$

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, $h_y = 2$.

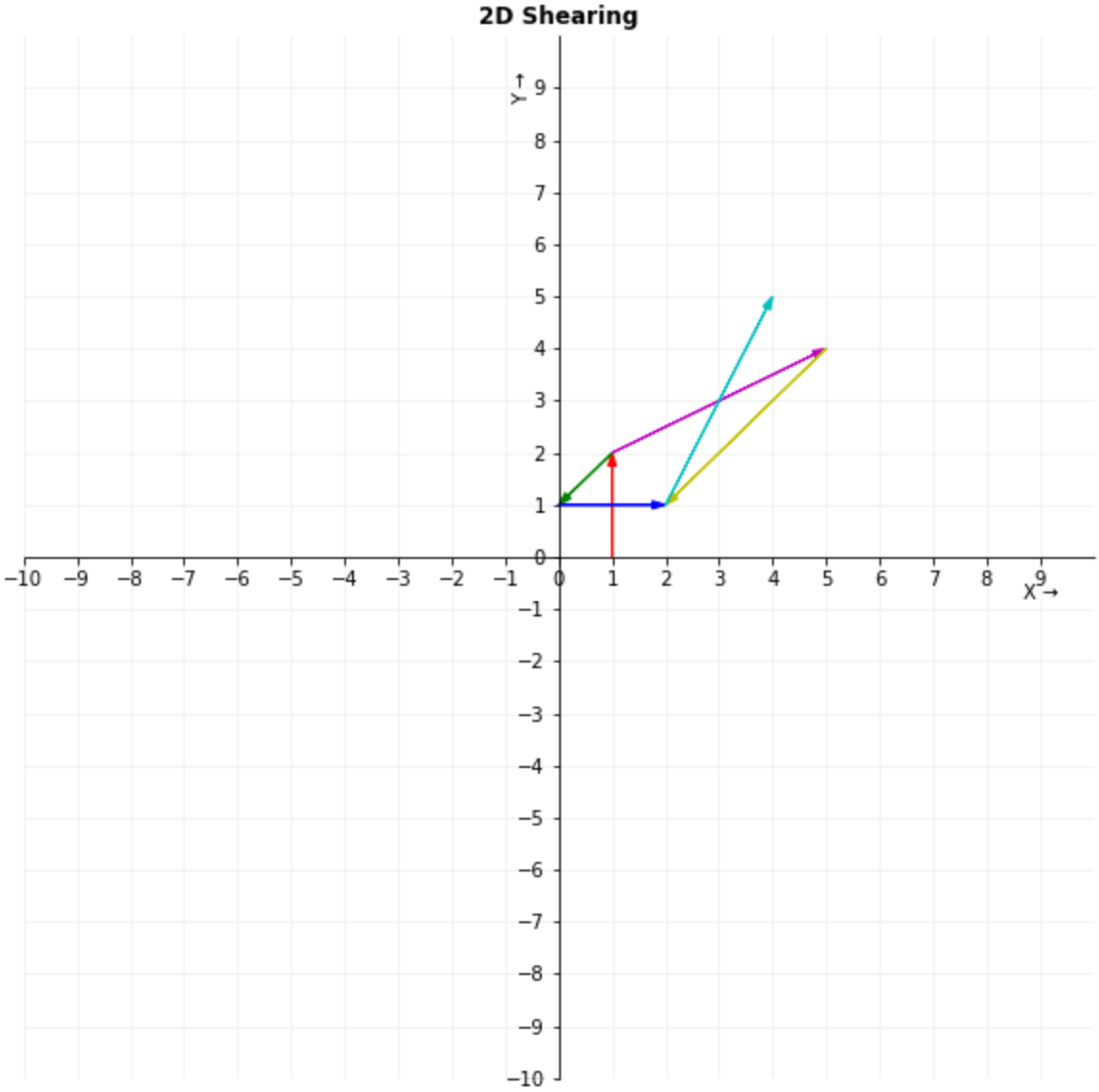


This time the shear is along the y dimension and the shift in y values gets larger as x increases.

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

hx 2.0

hy 2.0



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 ***with respect to 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.

Rotation is *always* with respect to the origin. Je répète, ***rotation is* always *with respect to the origin***. For some reason, many students get confused about this and convince themselves that rotation is done with respect to, e.g., the item's center, its left corner, its right corner, its upper corner, ...

Non. La rotation se fait par rapport à l'origine.

Nein. Die Drehung erfolgt in Bezug auf den Ursprung.

Rotation is always with respect to the origin.

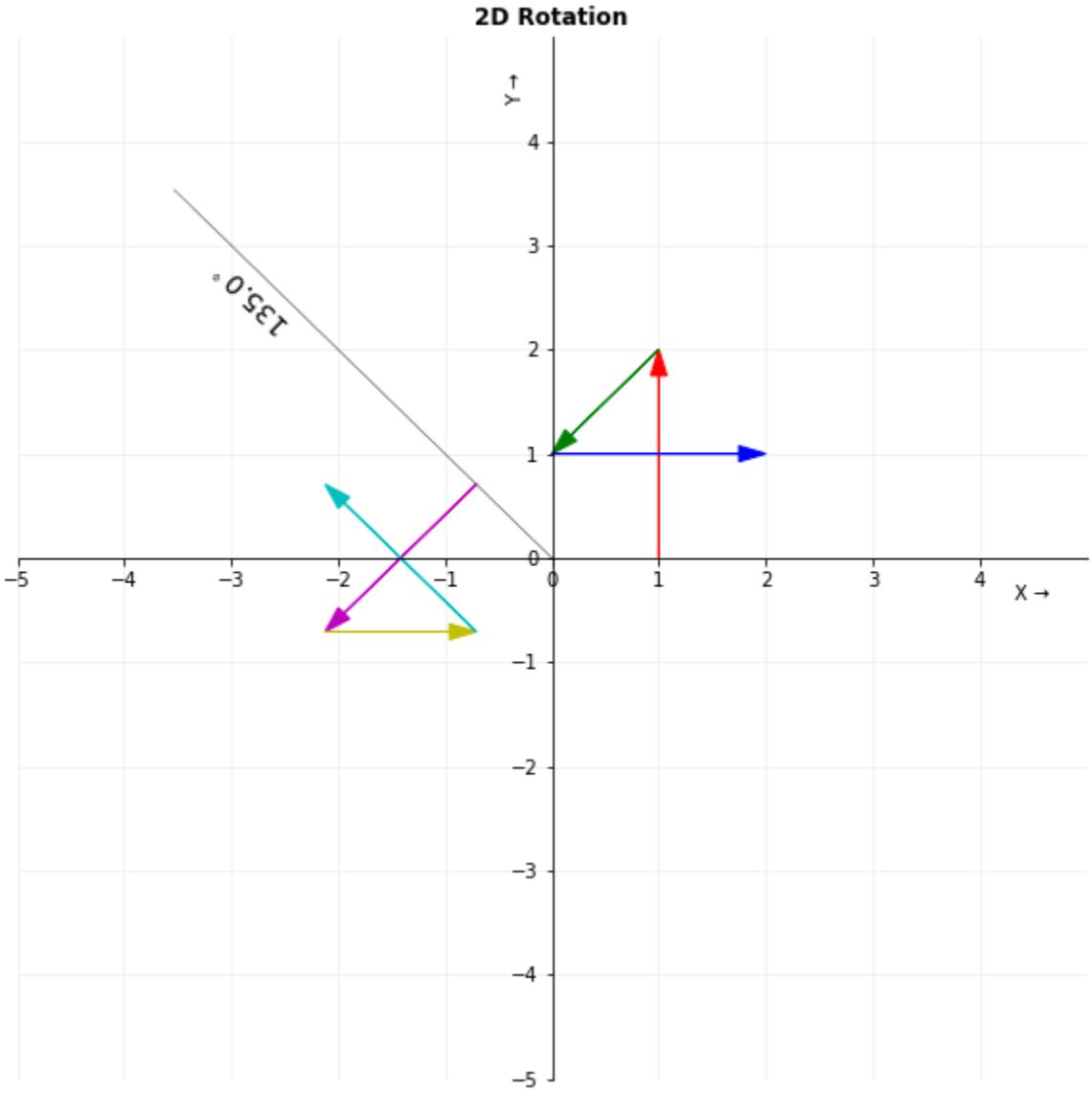
We represent rotation by the angle θ with the matrix $\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$.

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 $\theta = 135^\circ$.

theta 135.0



The orientation of the figure has been turned by 135° with respect to the origin. The size and relationships between the parts of the figure have not been changed.

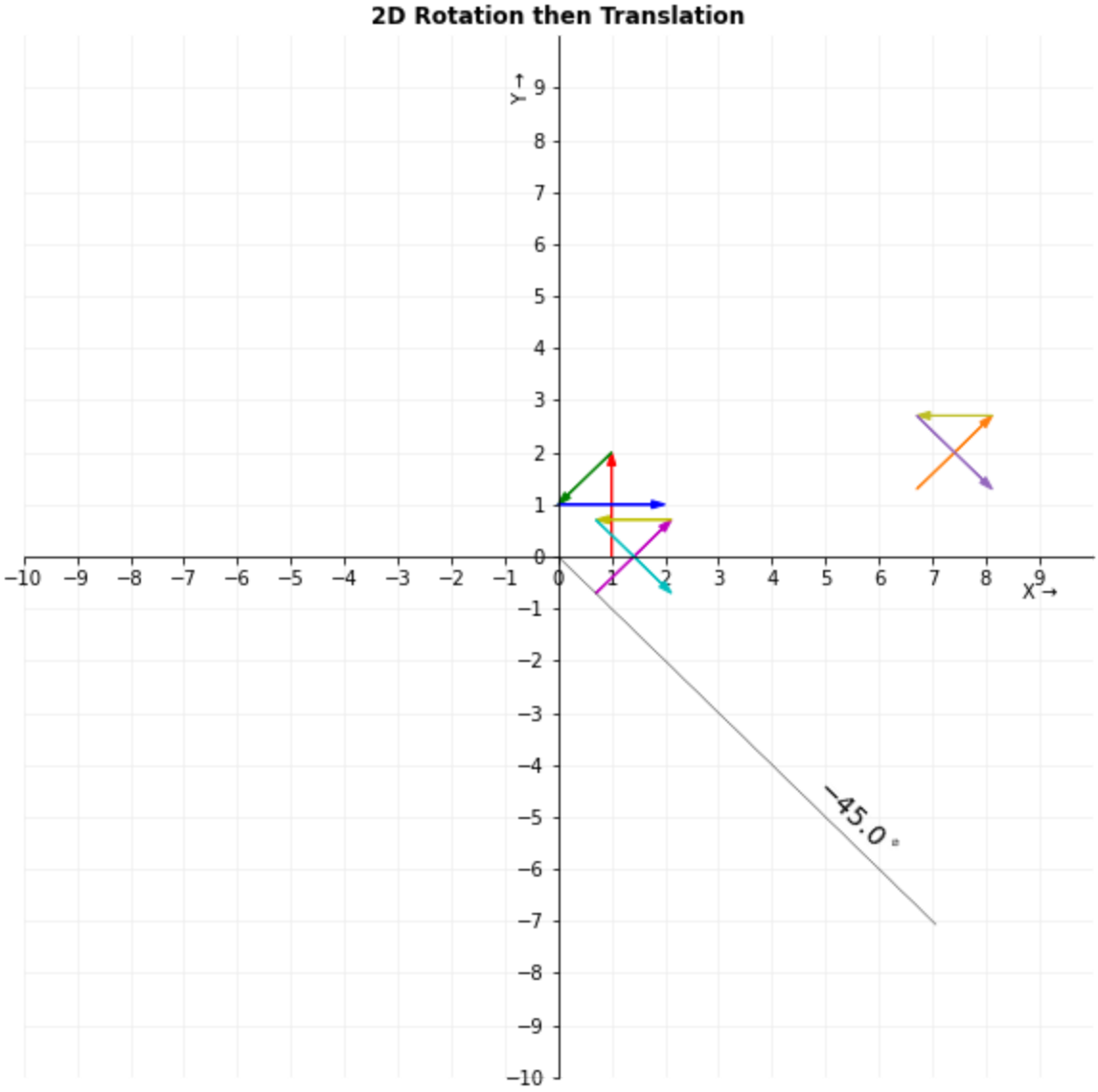
Rotation then Translation

After rotating our simple figure by -45°, we can then translate it by $t_x = 6, t_y = 2$.

theta -45.0

tx 6.0

ty 2.0



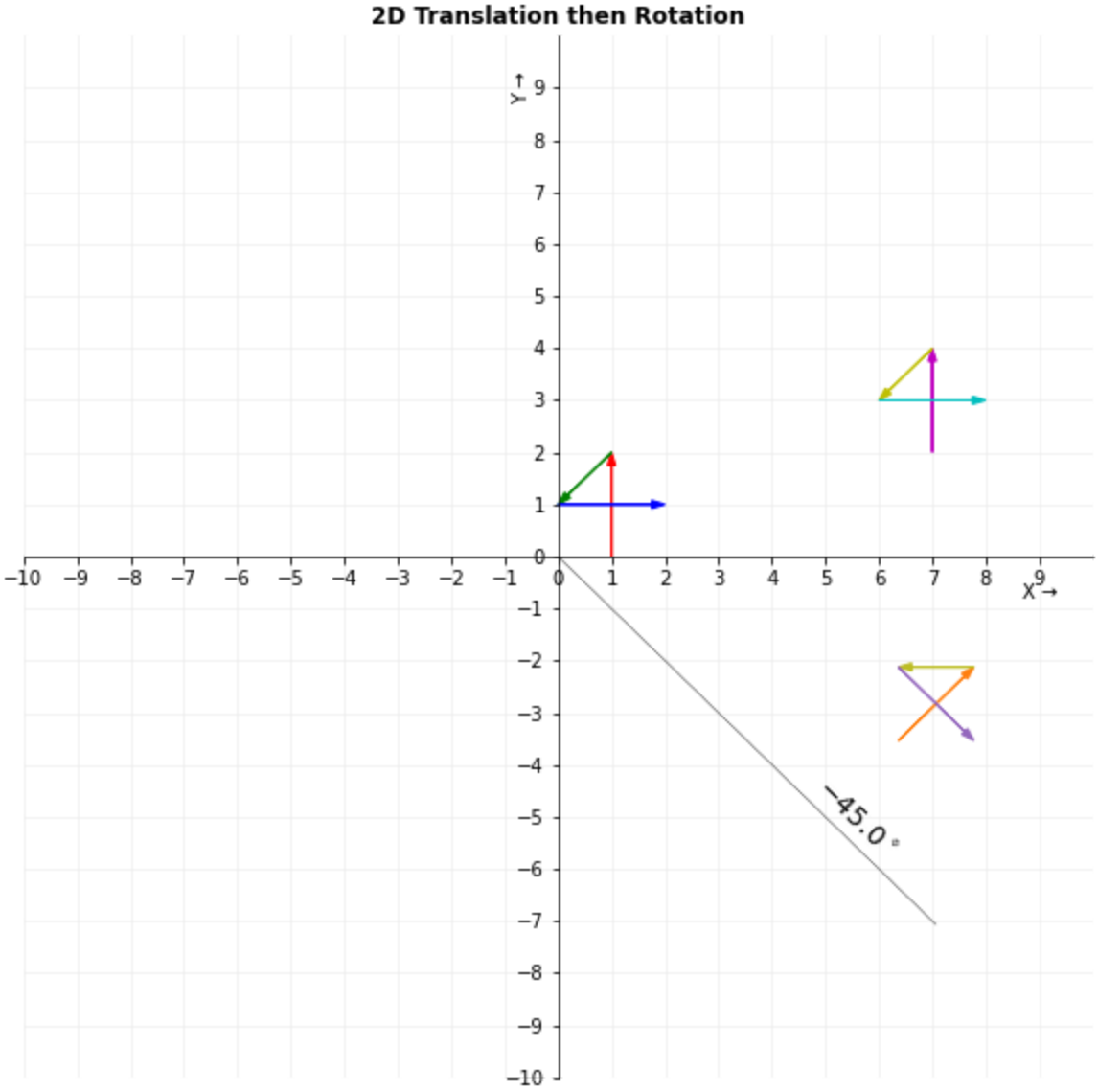
As expected, the figure is turned and then moved over to the right. No big surprises here.

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

tx 6.0

ty 2.0

theta -45.0



Doing the transforms in the opposite order yields a significantly different result. We see therefore that we cannot in general *commute* translation and rotation transforms.

> Why doesn't the line showing the angle of rotation touch the bottom of the figure as it does in the *Rotation then Translation* case? Under what special condition would the line showing the angle of rotation touch the bottom of the figure?

> Though a translation and a rotation do not *in general* commute, under what special conditions is it possible to commute a translation and a rotation and get the same result?

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 transform) 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 $\mathbf{p}_0$ by s_x, s_y, then shear it by h_x, h_y, and finally rotate it by θ, we would just compute

$$\begin{aligned}\mathbf{p}_1 &= \mathbf{S}_{xy}\mathbf{p}_0 \\ \mathbf{p}_2 &= \mathbf{H}_{xy}\mathbf{p}_1 \\ \mathbf{p}' &= \mathbf{R}_\theta\mathbf{p}_2\end{aligned}$$

But there's no reason to break it up into multiple operations on multiple points. We can chain the matrix multiplications together,

$$\mathbf{p}' = \mathbf{R}_\theta(\mathbf{H}_{xy}(\mathbf{S}_{xy}\mathbf{p}_0))$$

The parenthesization shows that we are doing the scaling first, then the shearing, and finally the rotation. Instead of needing the intermediate points $\mathbf{p}_1$ and $\mathbf{p}_2$, we just go directly from $\mathbf{p}_0$ to $\mathbf{p}'$.

Further, we can use the *associativity* of matrix multiplication to rearrange the order in which we do the matrix multiplication to get,

$$\mathbf{p}' = (\mathbf{R}_\theta\mathbf{H}_{xy}\mathbf{S}_{xy})\mathbf{p}_0$$

Here we are first multiplying the three matrices $\mathbf{R}_\theta$, $\mathbf{H}_{xy}$, and $\mathbf{S}_{xy}$ together and then taking that composition and using it to transform the point $\mathbf{p}_0$ into the point $\mathbf{p}'$. This composition has the form,

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

By doing this composition, we have combined three transforms 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 as long as they all needed the same scaling, shearing, and rotation (in that order). This can result in a *vast* increase in efficiency.

You may have noticed that we did only scaling, shearing, and rotation. What does not appear here is the fourth fundamental transform, *translation*. Since translation has been expressed as a matrix *addition* instead of a matrix *multiplication*, it does not readily 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 transforms in a uniform way.

> Homogeneous coordinates were introduced in 1827 by August Möbius in his work on barycentric coordinates, *Der barycentrische Calcül*. Originally, they had nothing to do with computer graphics. (Obviously. :)

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.

A vector has $w = 0$, for reasons we'll explore a bit later.

Translation

Translation was originally 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,

$$\mathbf{T}_{xy}^h = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}.$$

Multiplying that transform matrix by the augmented point, we find

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

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

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

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

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

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

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

Properly transformed, as expected.

Observations

In the homogeneous coordinate representation, each of the four elementary transforms is 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 transform matrix.

You may have noticed that the w component is always 1 in each of the original points, transform matrices, and transformed points. This is true because each of these is an *affine* transform. (See *Appendix: Affine Transform* for more information.)

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 Transforms

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

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 transform matrix $\mathbf{T}_{xyz}^h$ and the original point $\mathbf{p}$. The transform matrix is on the left, the original point on the right.

Out[13]:

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

The matrix multiplication process does the required adding of the proper translation amounts to the corresponding coordinates. Translation is an affine transform so the w element is 1 for the original point, the transform 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 transform matrix $\mathbf{S}^h_{xyz}$ and the original point $\mathbf{p}$. The transform matrix is on the left, the original point on the right.

Out[14]:

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

The matrix multiplication process does the required scaling of the point's coordinates according to the corresponding scaling factor. Scaling is an affine transform so the w element is 1 for the original point, the transform 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^y_x means *shear along the x axis with respect to the y axis*. The five other possibilities are $h^z_x, h^x_y, h^z_y, h^x_z$, and h^y_z.

The three-dimensional homogeneous shearing matrix is $\mathbf{I}_4$ (the 4×4 identity matrix) with the six required shearing factors $h^y_x, h^z_x, h^x_y, h^z_y, h^x_z, h^y_z$ 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 transform matrix $\mathbf{H}_{xyz}^{h}$ and the original point $\mathbf{p}$. The transform matrix is on the left, the original point on the right.

The matrix is as follows.

Out[15]:

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

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 transform so the w element is 1 for the original point, the transform 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 transform matrix $\mathbf{R}_x^h$, $\mathbf{R}_y^h$, or $\mathbf{R}_z^h$ and the original point $\mathbf{p}$. The transform matrix is on the left, the original point on the right.

Out[16]:

Rotation

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

Out[17]:

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

Out[18]:

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

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 transform so the w element is 1 for the original point, the transform matrix, and the resulting transformed point.

Composition of Transforms

As seen in the two-dimensional case, a series of three-dimensional transforms may be combined to reduce the cost of applying the same transform 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 transform would be computed as:

Out[19]:

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

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

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

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

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

Since the rotation in this combined transform 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.)

Out[20]: In the special case where the translation g is the inverse of the translation f (that is, $g = -f$), the transform 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 transform.

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

Out[21]:

Efficiency of Transforms

As mentioned above, once the transform 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 transform

$$(\mathbf{T}_f^{-1})(\mathbf{R}_\theta^z)(\mathbf{T}_f) =$$

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

Combined transform

$$(\mathbf{T}_f^{-1}\mathbf{R}_\theta^z\mathbf{T}_f) =$$

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

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.

Out[22]: 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 transform

$$(\mathbf{T}_f^{-1})(\mathbf{R}_\theta^z)(\mathbf{T}_f) =$$

$$\begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & -4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.866 & -0.5 & 0 & 0 \\ 0.5 & 0.866 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Combined transform

$$(\mathbf{T}_f^{-1}\mathbf{R}_\theta^z\mathbf{T}_f) =$$

$$\begin{bmatrix} 0.866 & -0.5 & 0 & -1.77 \\ 0.5 & 0.866 & 0 & 0.598 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Order of Transforms

The composition of transforms is order dependent as matrix multiplication is *not* commutative. This can be shown by simply reversing the order of the transforms 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.

Out[23]:

Original transform

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

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

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

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

Out[24]:

Reversed transform

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

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

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

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

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 transform that affected the x and y elements is quite different.

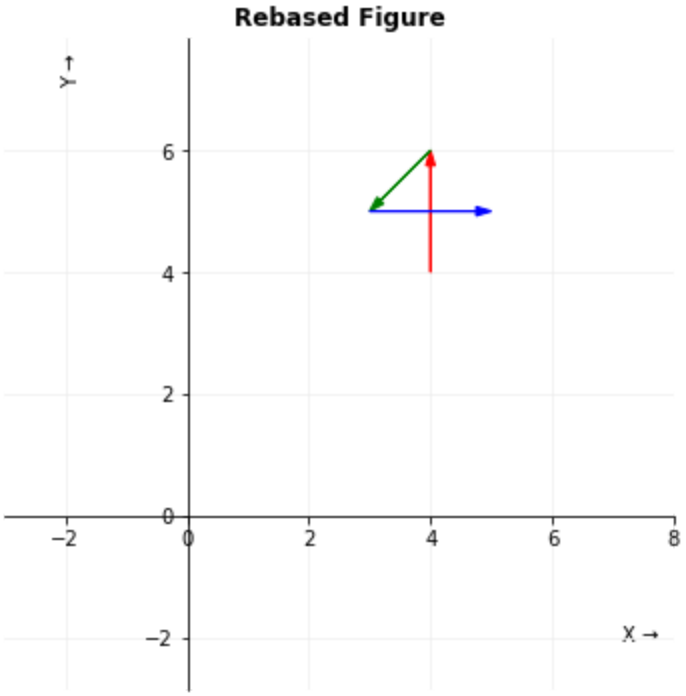
Geometric Meaning of Combined Transform

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

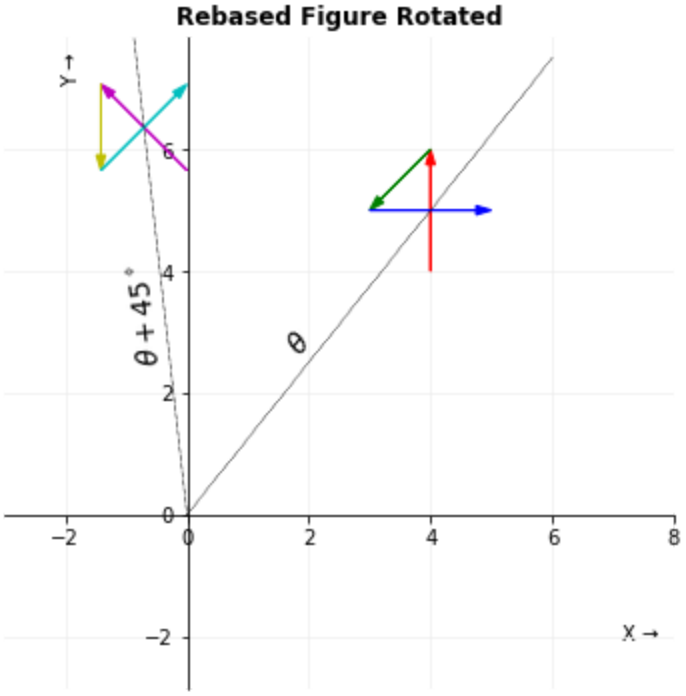
Given the infinite number of possible transforms, it is not possible to make a definitive geomerical statement about all of them, but certain combinations are amenable to intuitive geometrical explanation.

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

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

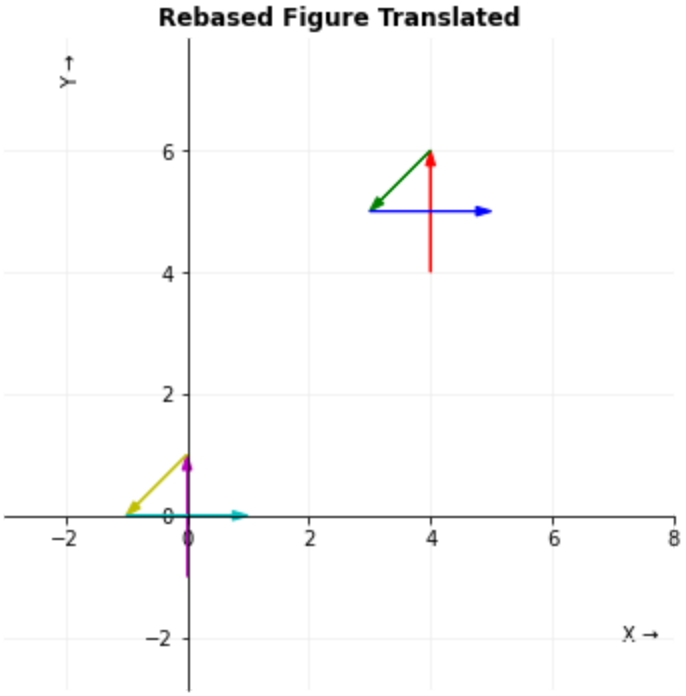


If we now rotate the figure 45°, we obtain the following.

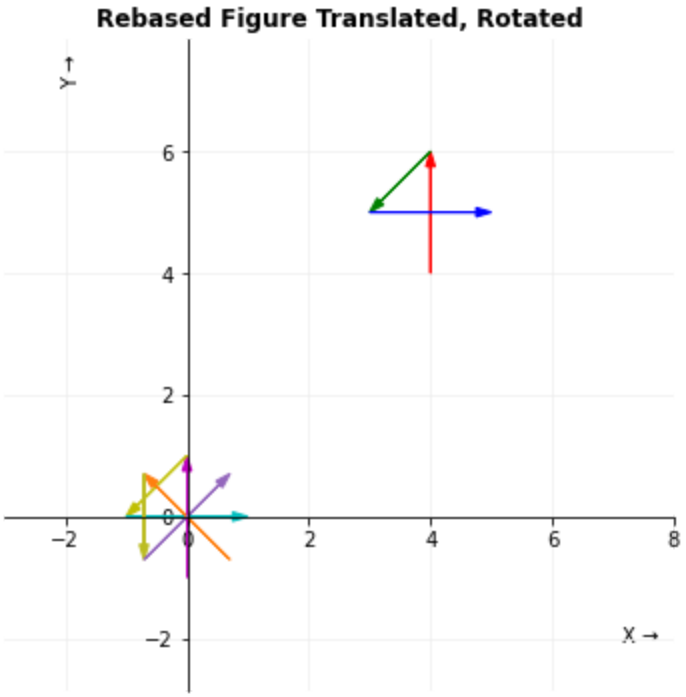


While it is indisputable that the figure has been rotated 45°, the rotation was with respect to the origin, and probably not what was wanted or expected. Intuitively, one thinks of rotation as being with respect to the center of the object being rotated.

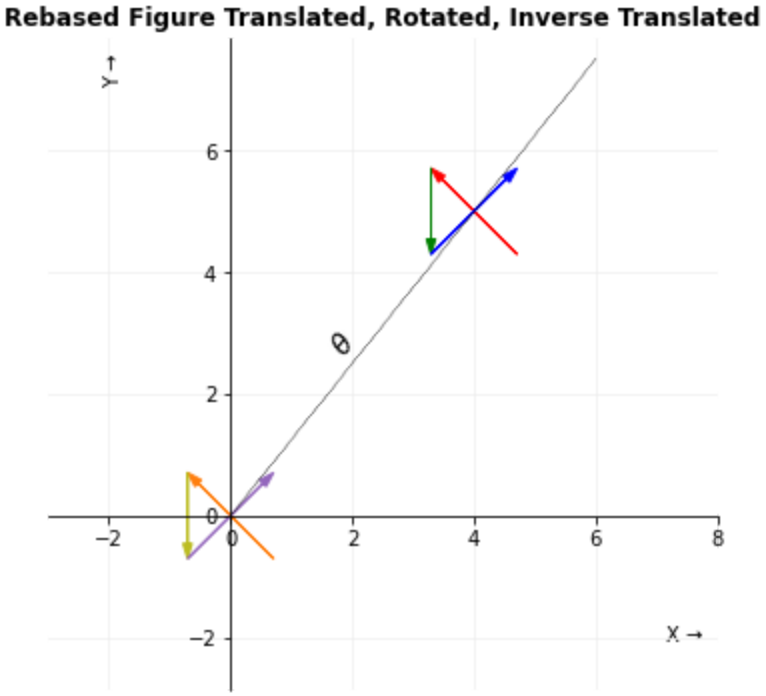
If we want our figure to be rotated about its own center, we must ensure that the center of the object is at the origin when the object is rotated. That's easy enough to do, as we can see in the following.



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.



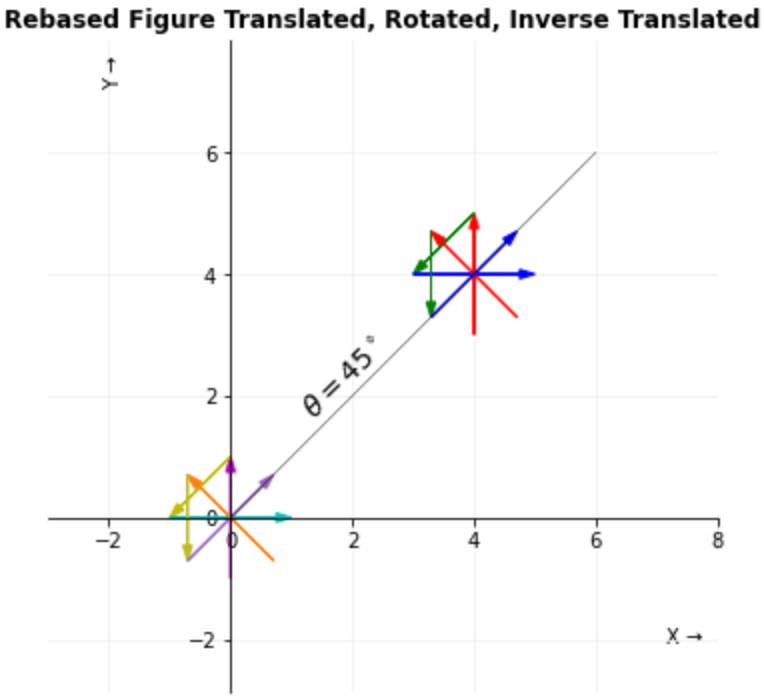
Now that the figure has been rotated about its center for 45°, all that remains is to move it back to its original position. We do this with an inverse translation, that is, a translation that is the negative of the translation that was used to move the object to the origin.



Out[30]: 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 transform, and then returning it to its original position can be used with any of the elementary transforms.

Appendix: Rotation Transform Derivation

A common question that comes up is “Why the heck does the rotation transform have that bizarre form?”

$$
\begin{aligned}
\mathbf{p}' &= \mathbf{R}_\phi \mathbf{p}_0 \\
&= \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \\
&= \begin{bmatrix} x\cos\phi - y\sin\phi \\ x\sin\phi + y\cos\phi \end{bmatrix}
\end{aligned}
$$

While you really ought to be able to figure this out for yourself using basic trigonometry, let's derive this form.

The key observation is that rotation is much easier if you express coordinates in *polar* form instead of *Cartesian* form. That is, instead of representing a point $\mathbf{p}_0$ as (x, y), represent it as (r, θ), where $r = \sqrt{x^2 + y^2}, \theta = \arctan(x, y)$.

> ***Don't read anything further — use the hint of polar form and try to derive this yourself!***

Rotating a point $\mathbf{p}_0 = (r, \theta)$ by the angle ϕ is absolutely trivial. The result is simply $(r, \theta + \phi)$.

OK, great. However, we're doing everything else in Cartesian form, so how do we get x' and y'? Well, using basic trigonometry, we know that to convert a polar representation to Cartesian form, we use $\cos$ and $\sin$ thus,

$$
\begin{aligned}
x' &= r\cos(\theta + \phi) \\
\\
y' &= r\sin(\theta + \phi)
\end{aligned}
$$

Using the trigonometric identities for $\cos$ and $\sin$ of the sums of angles, we find,

$$
\begin{aligned}
x' &= r(\cos\theta\cos\phi - \sin\theta\sin\phi) \\
&= r\cos\theta\cos\phi - r\sin\theta\sin\phi \\
\\
y' &= r(\sin\theta\cos\phi + \cos\theta\sin\phi) \\
&= r\sin\theta\cos\phi + r\cos\theta\sin\phi \\
&= r\cos\theta\sin\phi + r\sin\theta\cos\phi
\end{aligned}
$$

Since our original point was (r, θ), we have the original Cartesian coordinates $x = r\cos\theta$ and $y = r\sin\theta$. Substituting these identities we find,

$$
\begin{aligned}
x' &= r\cos\theta\cos\phi - r\sin\theta\sin\phi \\
&= x\cos\phi - y\sin\phi \\
\\
y' &= r\cos\theta\sin\phi + r\sin\theta\cos\phi \\
&= x\sin\phi + y\cos\phi
\end{aligned}
$$

Or, when expressed in matrix form,

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x\cos\phi - y\sin\phi \\ x\sin\phi + y\cos\phi \end{bmatrix}$$

which is easily recognized as the multiplication,

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

Ta-da!

That was pretty easy, wasn't it?

Appendix: Affine Transform

An *affine* transform is any transform that preserves ① points-on-a-line (that is, any points that are colinear before the transform are still colinear after the transform) and ② proportionality of distances (that is, while the length of a line segment may change after the transform, the proportion of distances of points on that line segment remain constant: the midpoint is still the midpoint, a point one-third of the way from one end of the line segment is still one-third of the way, etc.).

With that definition, we find that all of the four fundamental transforms — *translation*, *scaling*, *shearing*, and *rotation* — are affine.

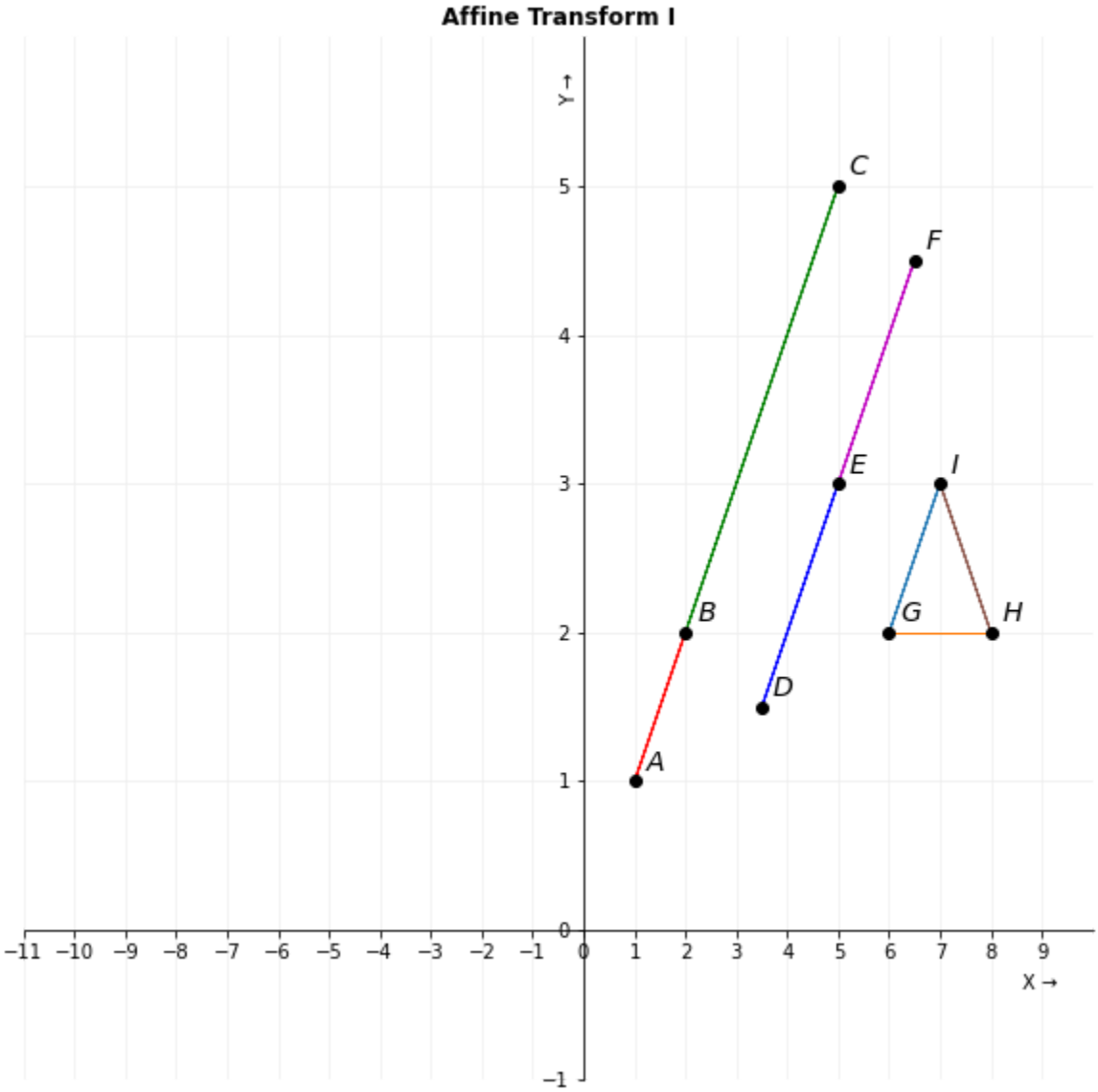
After an affine transform,

- A point is still a point: it doesn't get stretched into a line or a line segment.
- A line or line segment is still a line or a line segment: it doesn't get compressed into a point.
- Two lines or line segments that were parallel are still parallel.
- The proportional positions of points on a line or line segment remain the same. For example, a point that is one-third of the way from one end of a line segment to the other end is still one-third of the way, the midpoint of a line segment is still the midpoint, etc.

On the other hand, there are some properties that are *not* preserved,

- The measure of an angle formed by two lines or line segments may have changed.
- The length of a line segment may have changed.

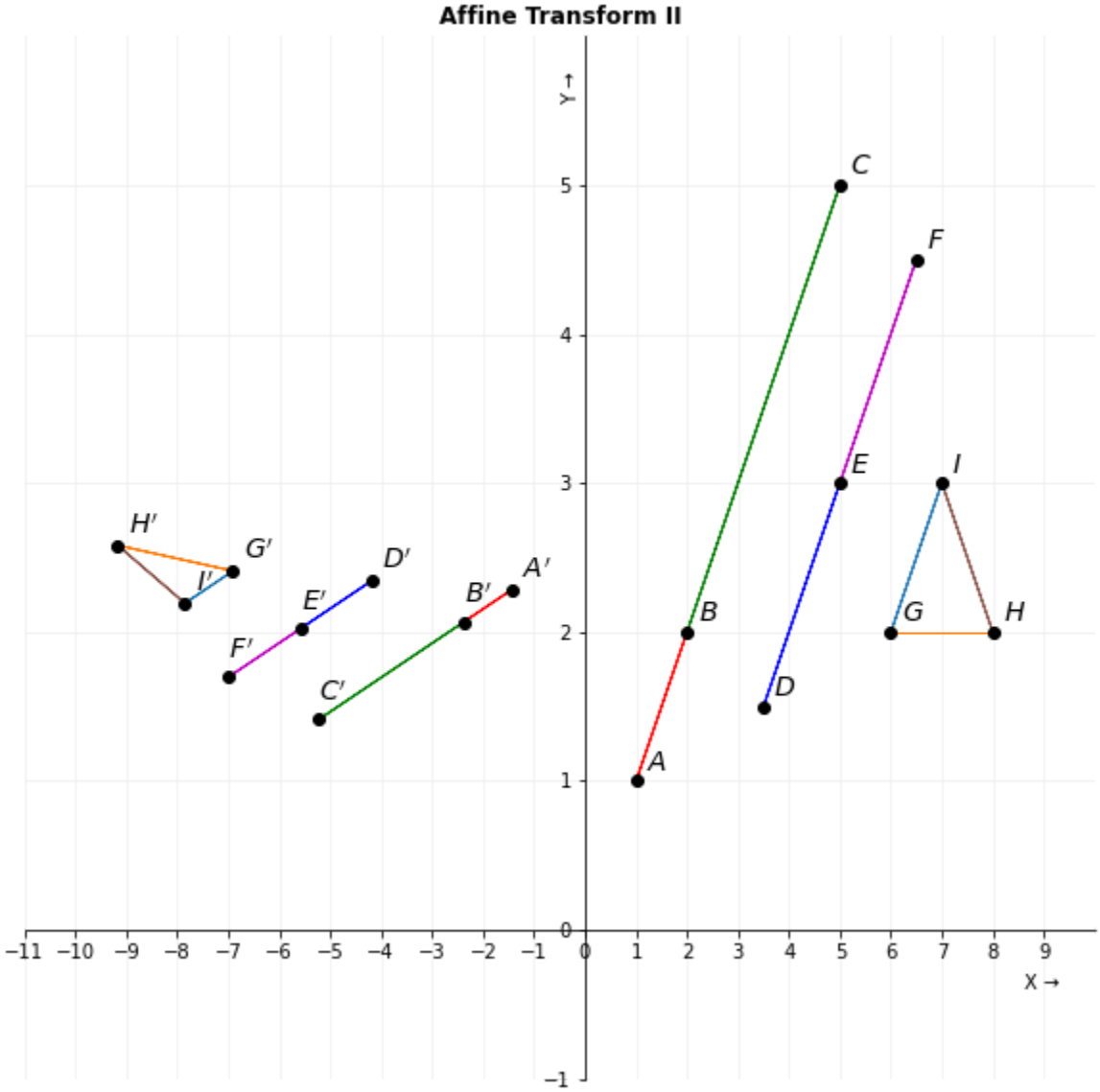
Here's an example,



In this diagram, we see the line segment $\overline{AC}$ with point $\mathbf{B}$ one-quarter of the way from $\mathbf{A}$ to $\mathbf{C}$. Line segment $\overline{DF}$ has point $\mathbf{E}$ as its midpoint. The $\triangle GHI$ is isosceles, with $\angle IGH = \angle IHG$.

> Ever wonder about that weird-looking word “isosceles”? It comes from the ancient Greek word ἰσοσκελής isoskelēs → ἰσοσ isos “equal” + σκελος skelos “leg” + ής ēs (adjective ending) → “equal-legged”.
>
> Euclid applied this term to triangles with *exactly* two equal sides. Today in geometry it means *at least* two equal sides. The difference is that today we consider equilateral triangles also to be isosceles while Euclid would not have.

Now, let's perform a series of affine transforms on this figure: a rotation of $\frac{\pi}{5}, 36°$, then a shearing of $(0.25, 1.10)$, a scaling of $(0.90, 0.50)$, a second rotation now of $\frac{3\pi}{2}, 135°$, and finally a translation of $(-0.5, 2.5)$ (in that order). Our original points are now the transformed points $\mathbf{A}'$, $\mathbf{B}'$, $\mathbf{C}'$, $\mathbf{D}'$, $\mathbf{E}'$, $\mathbf{F}'$, $\mathbf{G}'$, $\mathbf{H}'$, and $\mathbf{I}'$.



In this second diagram, we clearly see that $\mathbf{A}'$, $\mathbf{B}'$, and $\mathbf{C}'$ are just as colinear as $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are, and they form the line segment $\overline{A'C'}$. $\mathbf{B}'$ is one-quarter of the way from $\mathbf{A}'$ to $\mathbf{C}'$ just as $\mathbf{B}$ is one-quarter of the way from $\mathbf{A}$ to $\mathbf{C}$.

Similarly, $\mathbf{D}'$, $\mathbf{E}'$, and $\mathbf{F}'$ are just as colinear as $\mathbf{D}$, $\mathbf{E}$, and $\mathbf{F}$ are, and they form the line segment $\overline{D'F'}$. $\mathbf{E}'$ is the midpoint of line segment $\overline{D'F'}$ just as $\mathbf{E}$ is the midpoint of line segment $\overline{DF}$.

Line segments $\overline{A'C'}$ and $\overline{D'F'}$ are parallel, just as $\overline{AC}$ and $\overline{DF}$ are.

What we *don't* have is preservation of line segment lengths. Clearly, the length of line segment $\overline{A'C'}$ is shorter than that of $\overline{AC}$. Similarly, the length of $\overline{D'F'}$ is shorter than that of $\overline{DF}$.

We also don't have even *similarity* of $\triangle GHI$ with $\triangle G'H'I'$, let alone *congruence*. $\triangle G'H'I'$ is not even isosceles after the transform. In fact, no angle of $\triangle GHI$ is preserved in $\triangle G'H'I'$ and neither is the length of any of its sides.

For those of you who don't trust the diagram (and why should you? :), here are the positions of the points, pre- and post-transform as well as the line segment lengths (all rounded to three decimal places). From this information, you can compute the colinearity, parallelism, and proportionality measures to convince yourself of the truth of the statements made above.

```
Point Positions
  A (   1.000,   1.000 ) ->  A' (  -1.443,   2.283 )
  B (   2.000,   2.000 ) ->  B' (  -2.386,   2.066 )
  C (   5.000,   5.000 ) ->  C' (  -5.215,   1.416 )
  D (   3.500,   1.500 ) ->  D' (  -4.176,   2.347 )
  E (   5.000,   3.000 ) ->  E' (  -5.590,   2.021 )
  F (   6.500,   4.500 ) ->  F' (  -7.005,   1.696 )
  G (   6.000,   2.000 ) ->  G' (  -6.909,   2.410 )
  H (   8.000,   2.000 ) ->  H' (  -9.171,   2.582 )
  I (   7.000,   3.000 ) ->  I' (  -7.852,   2.193 )

Segment Lengths
  AB   1.414 ->  A'B'   0.968
  BC   4.243 ->  B'C'   2.903
  DE   2.121 ->  D'E'   1.451
  EF   2.121 ->  E'F'   1.451
  GH   2.000 ->  G'H'   2.268
  HI   1.414 ->  H'I'   1.375
  IG   1.414 ->  I'G'   0.968
```

Appendix: Rigid Bodies and their Transforms

In “the real world”, a *Rigid Body* is one which doesn't *deform* — that is, it has a constant shape and size. No matter how one applies force to it, its shape and size remain constant. A good example of a real-world rigid body is a brick. No matter how much one tries to squeeze or stretch a brick, it stays the same shape and size. (Until, of course, it disintegrates. :)

Cherokee Brick Company

In CG, we also have the concept of a rigid body except it's a body that we are not *allowed* to deform. Since models are all just bits, we have the *ability* to deform any model any which way we want, but when a model is declared to be a rigid body, we *agree* not to deform it. After all, unless one is making a movie about magic or hallucinations, one doesn't tend to see bricks squeezed and stretched.

Hardly ever happens in Real Life

Given rigid bodies, what transforms are we allowed to perform that would *not* cause deformation?

We define deformation as changing the distance r between any pair of points of the model. Consider two arbitrary points of a model, $p_1 = (x_1, y_1, z_1)^T$ and $p_2 = (x_2, y_2, z_2)^T$. The original distance r between these two points is computed as,

$$\begin{aligned} d_x &= x_1 - x_2 \\ d_y &= y_1 - y_2 \\ d_z &= z_1 - z_2 \\ r &= \sqrt{d_x^2 + d_y^2 + d_z^2} \end{aligned}$$

If we then perform a transform on p_1, p_2, we obtain $p_1' = (x_1', y_1', z_1')^T$ and $p_2' = (x_2', y_2', z_2')^T$. Similar to r, we compute the transformed distance r' as,

$$\begin{aligned} d_x' &= x_1' - x_2' \\ d_y' &= y_1' - y_2' \\ d_z' &= z_1' - z_2' \\ r' &= \sqrt{d_x'^2 + d_y'^2 + d_z'^2} \end{aligned}$$

if $r = r'$ for *all* pairs of points p_1, p_2 of the model, we say the transform does *not* deform and is therefore a legal rigid body transform.

We now consider each of the four fundamental transforms and evaluate their suitability as a rigid body transform.

> ***NOTE***: You might pause for a moment before reading on and try to convince yourself of the suitability of each transform. Try also to visualize the *reason* a particular kind of transform is or is not suitable as a rigid body transform.

Translation

Clearly, *translation* does not change the distance r between p_1 and p_2 since both of p_1 and p_2 are translated the same. That is, no matter what t_x, t_y, t_z values are used, the d_x', d_y', d_z' values remain the same as d_x, d_y, d_z. For example,

$$\begin{aligned} d_x' &= x_1' - x_2' \\ &= (x_1 + t_x) - (x_2 + t_x) \\ &= (x_1 - x_2) + (t_x - t_x) \\ &= x_1 - x_2 \\ &= d_x \end{aligned}$$

Similar arguments follow for d_y', d_z'.

We can also consider a translation of a *model* to be a translation of the *coordinate system* (in the opposite direction). [Do you remember that from *Linear Algebra*?] Thinking of the translation that way, it's pretty easy to visualize that the distance between any two points of the model didn't change.

Scaling

On the other hand, *scaling* does change the distance between two points. (That's kind of the point of the scaling transform, isn't it?)

For example, if we do a scaling s_x in the x dimension, we find,

$$\begin{aligned} d'_x &= x'_1 - x'_2 \\ &= s_x x_1 - s_x x_2 \\ &= s_x(x_1 - x_2) \\ &= s_x d_x \end{aligned}$$

We see that d'_x differs from d_x by a factor s_x. Thus we will get r' that's different from r.

Sharp-thinking readers are already voicing an objection to the previous paragraph. :) “Suppose,” they say, “s_x was equal to 1? That wouldn't change d_x and therefore r wouldn't change, would it?” No, it wouldn't, and in that case scaling is *not* a deformation. On the other hand, having $s_x = s_y = s_z = 1$ is just the *identity* transform and hardly counts as a scaling transform.

With $s_x, s_y, s_z = 1$, $\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ becomes $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \mathbf{I}_4$

Even sharper-thinking readers are about to voice a subtler objection: “What about the case where s_x (and/or s_y, s_z) is equal to -1? That would change d_x, but all it would do is *negate* it. That wouldn't change r, would it?” No, that doesn't change r either. However, remember the effect of having a scaling factor < 0. Instead of just scaling, such a transform also *reflects* about the corresponding axis.

As long as s_x (and/or s_y, s_z) is restricted to ± 1, the distance r' between p'_1 and p'_2 will be the same as the distance r between p_1 and p_2 and the identity / reflection transform would be an acceptable (non-deforming) rigid body transform.

With $s_x = -1$, we have Brick through the Looking Glass

Shearing

Shearing, much like scaling, is *not* a rigid body transform as it most definitely changes the distance between points. (As with the general case of scaling, that's kind of the point of the shearing transform.) We assume the shearing factors are $\neq 0$, as that would result in the identity transform.

For example, if we shear the x axis with respect to the y axis by a factor $h_x^y \neq 0$, we end up with,

$$
\begin{aligned}
x'_1 &= x_1 + h_x^y y_1 \\
x'_2 &= x_2 + h_x^y y_2 \\
\\
d'_x &= x'_1 - x'_2 \\
&= (x_1 + h_x^y y_1) - (x_2 + h_x^y y_2) \\
&= (x_1 - x_2) + (h_x^y y_1 - h_x^y y_2) \\
&= d_x + h_x^y (y_1 - y_2)
\end{aligned}
$$

This result differs from d_x as long as $y_1 \neq y_2$. In the general case, we cannot depend on $y_1 = y_2$ so d'_x in general will not be equal to d_x. Similar arguments follow for d'_y, d'_z.

Rotation

Finally, *rotation*, much like translation, is a rigid body transform. As in the translation case, we can consider the rotation of a model about any of the x, y, or z axes as a rotation of the coordinate system about the same axis (but in the opposite direction). Visualizing a rotation this way makes the classification of the rotation transform as a rigid body transform easy.

While “easy”, this visualization is not all that *satisfying*. Let's *prove* that this is true! To make it simpler for this proof, let's do it in two dimensions; the three-dimensional case follows a similar development.

As a reminder, to rotate a point $\mathbf{p}$ by an angle θ in two dimensions, we compute

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

Therefore,

$$\begin{aligned} x_1' &= x_1\cos\theta - y_1\sin\theta \\ y_1' &= x_1\sin\theta + y_1\cos\theta \\ \\ x_2' &= x_2\cos\theta - y_2\sin\theta \\ y_2' &= x_2\sin\theta + y_2\cos\theta \end{aligned}$$

We first compute the value $r^2 = d_x^2 + d_y^2$ for the original two points, p_1, p_2. (There's no reason to go all the way to $r = \sqrt{d_x^2 + d_y^2}$ if all we're going to be doing is checking if $r = r'$. We can more easily check if $r^2 = r'^2$.)

Let's start with r^2,

$$\begin{aligned} r^2 &= d_x^2 + d_y^2 \\ &= (x_1 - x_2)^2 + (y_1 - y_2)^2 \\ &= x_1^2 - 2x_1x_2 + x_2^2 + y_1^2 - 2y_1y_2 + y_2^2 \end{aligned}$$

Similarly, r'^2 is,

$$\begin{aligned} r'^2 &= d_x'^2 + d_y'^2 \\ &= (x_1' - x_2')^2 + (y_1' - y_2')^2 \\ &= x_1'^2 - 2x_1'x_2' + x_2'^2 + y_1'^2 - 2y_1'y_2' + y_2'^2 \end{aligned}$$

We already know what x_1', y_1', x_2', y_2' are so we can just plug them in and do a bit of algebra.

$$
\begin{aligned}
r'^2 &= (x_1 \cos\theta - y_1 \sin\theta)^2 \\
&\qquad - 2(x_1 \cos\theta - y_1 \sin\theta)(x_2 \cos\theta - y_2 \sin\theta) \\
&\qquad + (x_2 \cos\theta - y_2 \sin\theta)^2 \\
&\quad + (x_1 \sin\theta + y_1 \cos\theta)^2 \\
&\qquad - 2(x_1 \sin\theta + y_1 \cos\theta)(x_2 \sin\theta + y_2 \cos\theta) \\
&\qquad + y_2 \cos\theta)^2 \\
&= x_1^2 \cos^2\theta - 2x_1 y_1 \sin\theta\cos\theta + y_1 \sin^2\theta \\
&\quad + 2x_1 x_2 \cos^2\theta - 2x_2 y_1 \sin\theta\cos\theta - 2x_1 y_2 \sin\theta\cos\theta + 2y_1 y_2 \sin^2\theta \\
&\quad + x_2^2 \cos^2\theta - 2x_2 y_2 \sin\theta\cos\theta + y_2 \sin^2\theta \\
&\quad + x_1^2 \sin^2\theta + 2x_1 y_1 \sin\theta\cos\theta + y_1^2 \cos^2\theta \\
&\quad + 2x_1 x_2 \sin^2\theta + 2x_1 y_2 \sin\theta\cos\theta + 2x_2 y_1 \sin\theta\cos\theta + 2y_1 y_2 \cos^2\theta \\
&\quad + x_2^2 \sin^2\theta + 2x_2 y_2 \sin\theta\cos\theta + y_2^2 \cos^2\theta
\end{aligned}
$$

That looks a bit messy, but if you consider it closely, you'll see that there are a bunch of “equal-but-opposite-in-sign” terms that cancel each other out. One is then left with some terms that differ only in having a $\sin\theta$ or $\cos\theta$ factor. Those terms can be grouped and the result can be arranged as,

$$
\begin{aligned}
r'^2 &= x_1^2 \sin^2\theta + x_1^2 \cos^2\theta - 2x_1 x_2 \sin^2\theta - 2x_1 x_2 \cos^2\theta + x_2^2 \sin^2\theta + x_2^2 \cos^2\theta \\
&\quad + y_1^2 \sin^2\theta + y_1^2 \cos^2\theta - 2y_1 y_2 \sin^2\theta - 2y_1 y_2 \cos^2\theta + y_2^2 \sin^2\theta + y_2^2 \cos^2\theta
\end{aligned}
$$

Factoring, we get,

$$
\begin{aligned}
r'^2 &= x_1^2(\sin^2\theta + \cos^2\theta) - 2x_1 x_2(\sin^2\theta + \cos^2\theta) + x_2^2(\sin^2\theta + \cos^2\theta) \\
&\quad + y_1^2(\sin^2\theta + \cos^2\theta) - 2y_1 y_2(\sin^2\theta + \cos^2\theta) + y_2^2(\sin^2\theta + \cos^2\theta)
\end{aligned}
$$

We now make use of the trigonometric identity $\sin^2\theta + \cos^2\theta = 1$ leaving us with,

$$
\begin{aligned}
r'^2 &= x_1^2 - 2x_1 x_2 + x_2^2 + y_1^2 - 2y_1 y_2 + y_2^2 \\
&= r^2
\end{aligned}
$$

Ta-da!

That was pretty easy, wasn't it?

This derivation may be extended to three dimensions and then applied to rotations about any of the three axes x, y, z. (Do you see how to do this?)

Summary

In summary, the rigid body transforms are *translation*, *reflection* (about any axis), and *rotation* (about any axis). *Shearing* and *scaling* (except when they are *identity* transforms or the scaling is only a *reflection*) are prohibited.

A rigid body transform may change the orientation and/or position of a model, but *not* its size or proportions.

FYI: Tacoma Narrows Bridge Collapse

Here's a pointer to some contemporaneous footage of the collapse of the first Tacoma Narrows Bridge. The bridge was opened to traffic on 1940 July 01 and collapsed into the Puget Sound on 1940 November 07.

> https://www.youtube.com/watch?v=j-zczJXSxnw

More information on the bridge and its collapse can be found at the ever-handy Wikipedia page:

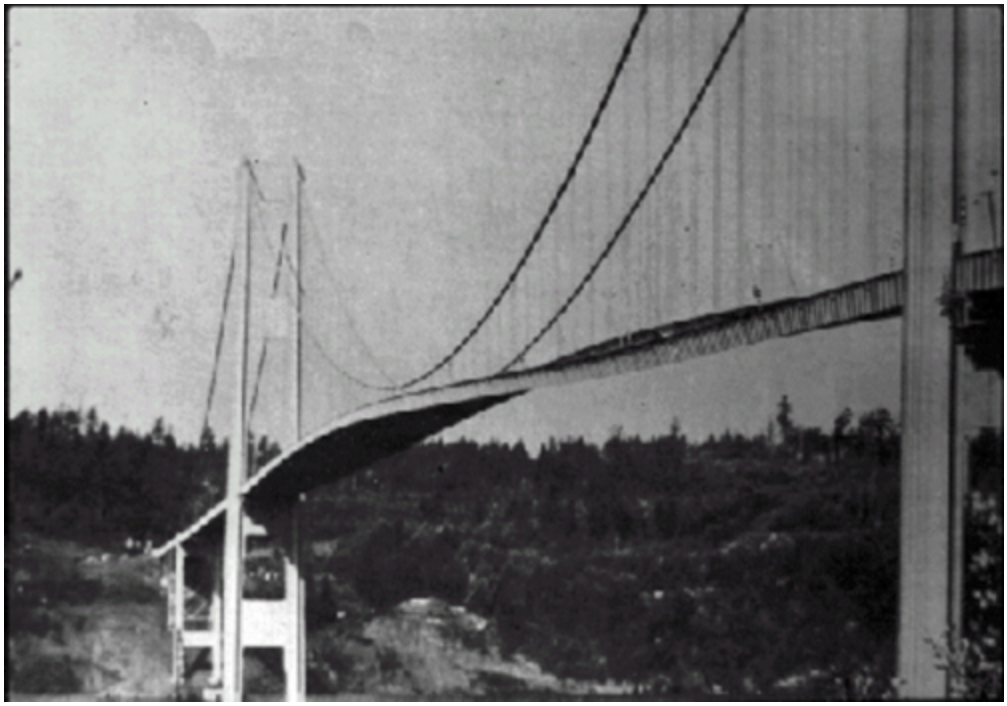
> https://en.wikipedia.org/wiki/Tacoma_Narrows_Bridge_(1940)

In brief summary, from the moment the road deck was laid during the bridge's construction, it moved up and down whenever the wind was strong enough. The movement was so intense that the construction workers nicknamed the bridge “Galloping Gertie”. This motion continued even though damping countermeasures were installed. An engineering study was commissioned to more deeply analyze the situation, but the bridge collapsed before the recommendations of the study could be implemented.

So why this *FYI* in a section on *Rigid Bodies*? Well, we used a brick as the Real World example of a Rigid Body. Anything made of Concrete is also a Rigid Body, as concrete doesn't bend very much at all either.

However, if you review the video at the given link and the picture below, it certainly *looks* as if concrete is bending quite a bit, doesn't it?

What do you think is really happening (given that the concrete is *not* bending)?



Galloping Gertie (during)

Galloping Gertie (after)

Волгоградский мост (Volgograd Bridge)

While Galloping Gertie is from 1940, the phenomena is still seen on bridges today. Here's an image of Волгоградский мост in Волгогра́д, Россия, from 2010 May 20. The effect is not as prominent as in the Tacoma Narrows case, but is still clearly visible.

Волгоградский мост (during)

You can see the full effect here,

> https://www.youtube.com/watch?v=uWP5d2t2JVE

Various individuals claimed that the bridge was heaving vertically “two meters”, but in this video it looks to be less than one meter. Locals nicknamed it the Танцующий мост (“Dancing Bridge”), ha!

It sure looks like concrete is bending, doesn't it? What's really happening?

After a engineering analysis, a dozen “semi-active tuned magnetorheological mass dampers” were installed in the Autumn of 2011. The Волгоградский мост has not collapsed. (At least, not yet. :)

Some more details are available on the Волгоградский мост Wikipedia page here,

> https://en.wikipedia.org/wiki/Volgograd_Bridge