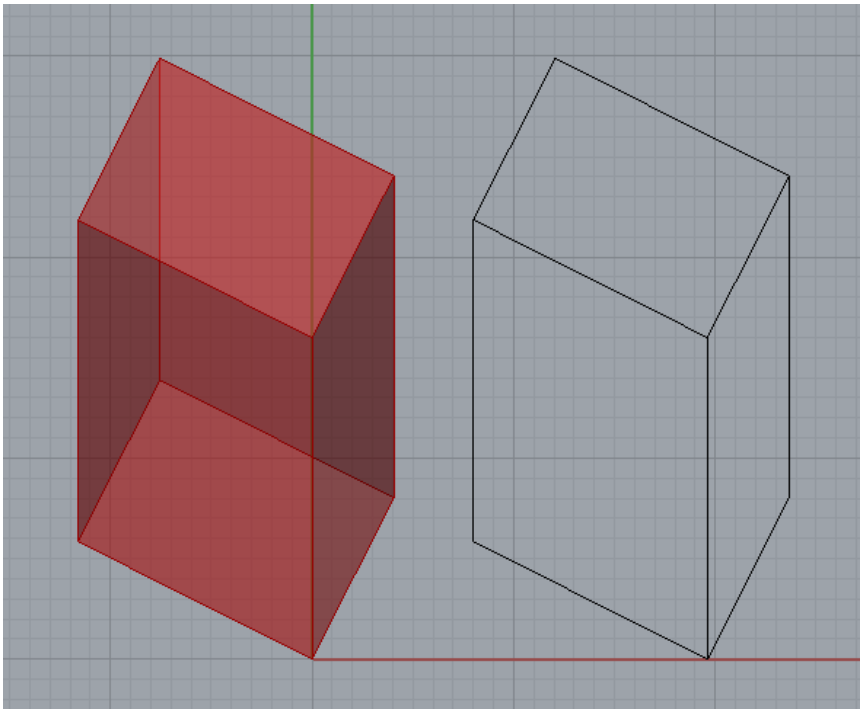
MP2 with GRASSHOPPER & RHINO



What you will learn

1. Create a “military projection” with Grasshopper and Rhino
2. Concatenate transformations
3. Master matrix multiplication with transformation matrices

List of relevant components used

Rotate

Shear

Scale

Move (translate)

Part 1- Create a Military Projection with Grasshopper and Rhino

To build a military projection of a geometry we need to perform three operations: rotation, shear, and projection. In this exercise we will use a Box as starting geometry (you can create the box with a **Box2Pt** component (surface>primitive) that create a box defined by two points).

1.1-Execute the rotation and the shear in Grasshopper with a **rotate** component (transform>euclidean) and a **shear** component (transform>affine), using the appropriate inputs. Check the values of the transformation matrices associated with the rotate and shear components connecting their X output to a transform matrix component (transform>util).

For the projection, instead of using Grasshopper and its component Project - therefore don´t use the Project component - we will use directly Rhinoceros interface with the following operations:

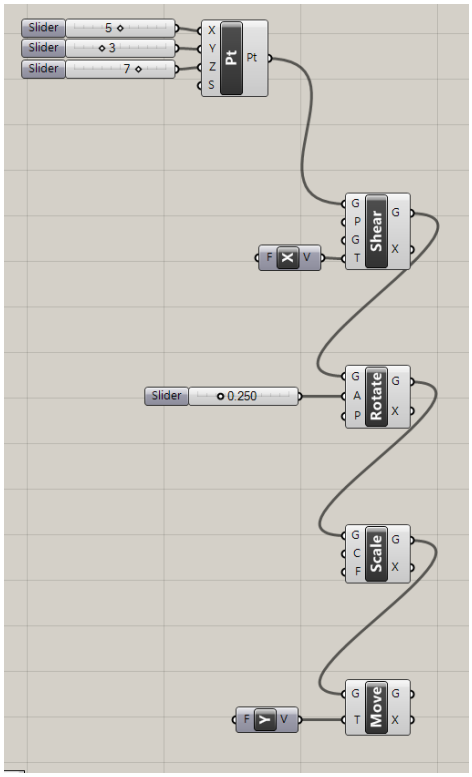
1.2-"Bake" the box after the rotation and the shear (right on the Grasshopper component that contains the transformed geometry>bake). Click the blue "top" button in the Rhino interface to select the top viewport. Type "make2d" in the Rhinoceros command line, and select the geometry “baked” from Grasshopper to obtain the military projection. This operation perform the projection of the geometry on the xy plane. Please note that it is important that the top viewport is selected in order to project the geometry right in the *xy* plane.

Part 2 – Concatenate transformations

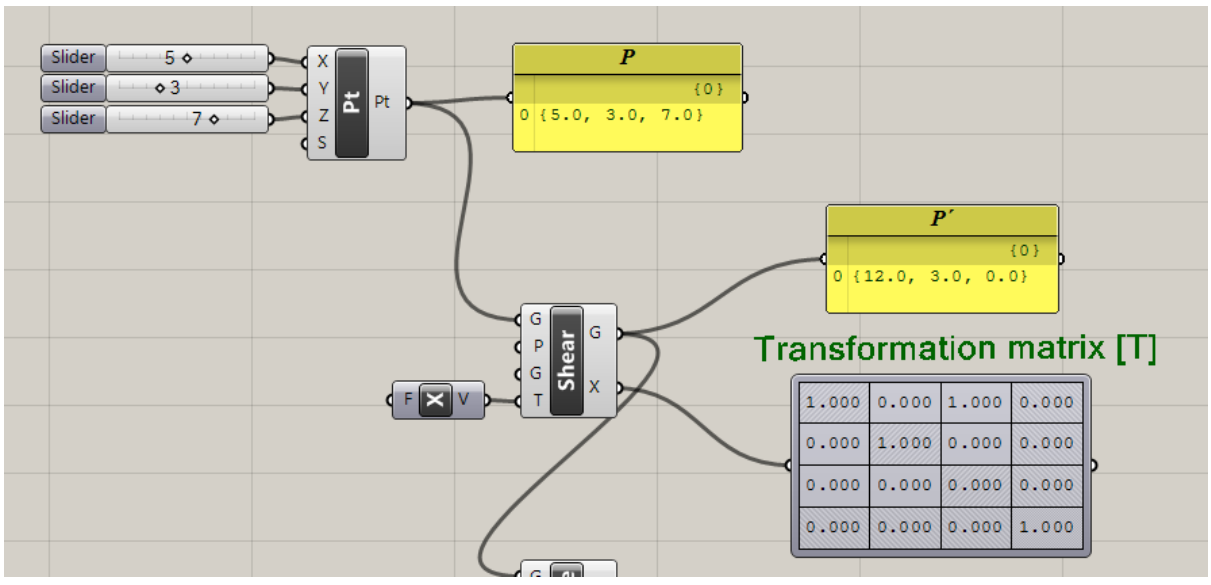
In this part you are asked to concatenate and apply multiple transformations to a geometry in Grasshopper and to execute in parallel the same operations “by hand” with matrix multiplications using the transformation matrices for each transformation.

We will use a point as starting geometry: create one in a random position with the **pointXYZ** component, but be sure that neither of the x y z coordinates are =0.

2.1- Concatenate and apply to the point geometry the following linear transformations one after the other with values of your choice: **shear**, **rotate**, **scale** (transform>affine), **move** (transform>euclidean)



2.2- Now execute the same four transformations this time "by hand" by multiplying the correspondent transformation matrix with the point coordinates before the transformation. The result of the matrix multiplication is the vector with the coordinates of the point after the transformation. In Grasshopper transformation matrices can be visualized for each transformation component by connecting its output X to a matrix component (transform>util). To visualize the coordinates of the point before and after each transformation, use a **Panel** component. Here below is the data you need to perform and check the first matrix multiplication relative to the first transformation:



Note on transformation matrices

In three dimensional space, transformation matrices have a dimension of 4x4, and the 3x1 vector of point coordinates $(x, y, z)^T$ should be represented as a 4x1 vector $(x, y, z, 1)^T$

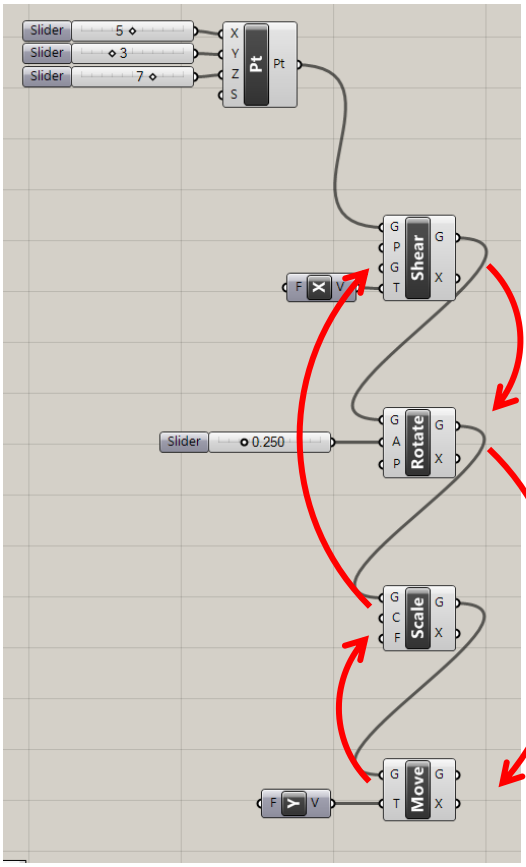
Example: apply a shear transformation matrix to the point P with coordinates P(5,3,7)

$$\begin{bmatrix}1 & 0 & 1 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 0\\0 & 0 & 0 & 1\end{bmatrix} * \begin{bmatrix}5\\3\\7\\1\end{bmatrix} = \begin{bmatrix}1*5+0*3+1*7+0*1\\0*5+1*3+0*7+0*1\\0*5+0*3+0*7+0*1\\0*5+0*3+0*7+1*1\end{bmatrix} = \begin{bmatrix}12\\3\\0\\1\end{bmatrix}$$

The new point position is P´(12,3,0)

Proceed similarly with your data for all the four transformations.

2.3- Create a new concatenation of linear transformations in grasshopper, using the same transformations components and inputs, but changing the order you apply the transformations. Keep besides the previous concatenation too.



Now compare the end position of the point after the four transformations are applied with the end position of the point of the previous concatenation, made of the same transformation components but applied in a different order. Does the end position of the point is the same, regardless the order you apply the fours transformations? Can you give an explanation why?