Applied Science Periodical **ISSN 0972-5504**
Volume - XXVIII, No. 1, February 2026
Journal website: https://internationaljournalsiwan.com/applied-science.php
ORCID Link: https://orcid.org/0009-0008-5249-8441
International Impact Factor: **9.0** https://impactfactorservice.com/home/journal/2296
Google Scholar: https://scholar.google.com/citations?user=BRweiDcAAAAJ&hl=en
Refereed and Peer-Reviewed Quarterly Periodical

Application of Linear Algebra in Computer Graphics: 2D and 3D Transformations

by **Sarita Agarwal, Gaurav Sharma, Laxmi Narain, Tanishq Negi, Aehsaas Jha and Kartik**

Department of Mathematics,
Acharya Narendra Dev College,
University of Delhi, India

(Received: January 27, 2026; Accepted: February 20, 2026;
Published Online: February 28, 2026)

Abstract:

Our research investigates the role of linear algebra in computer graphics, specifically focusing on its application in 2D and 3D transformations of objects within environments like Blender. This report explains the core linear algebra concepts that are, vectors and matrices, detailing their mathematical representations and operational principles for translation, rotation, scaling, shearing, in 2D and 3D objects.

1. Introduction:

1.1 The Role of Linear Algebra in Computer Graphics:

Linear algebra is used to perform various tasks such as:

(i) Representing Geometric Objects:

In computer graphics, objects are often represented using vectors and matrices, fundamental constructs of linear algebra. Points in 3D space are represented as vectors, while transformations such as rotations, translations, and scaling are achieved using matrices. For example, a 3D point (x, y, z) can be manipulated by multiplying it with transformation matrices to change its position or orientation within the scene.

(ii) Transformations and Operations:

Linear algebra enables various geometric transformations essential for creating dynamic and interactive graphics:

- **Translation:** Moving an object from one position to another using vector addition.
- **Rotation:** Rotating objects around an axis using rotation matrices.
- **Scaling:** Adjusting the size of objects through scaling matrices.
- **Shearing:** Distorting the shape of an object in a specific direction.

These transformations are combined using matrix multiplication, ensuring computational efficiency and precision.

(iii) Camera and Perspective Projections:

Simulating the behaviour of a camera in a 3D scene involves linear algebra. Perspective projection, which mimics the way humans perceive depth, is implemented using projection matrices. These matrices transform 3D coordinates into 2D screen coordinates, ensuring objects closer to the viewer appear larger, while distant objects appear smaller.

(iv) Rendering and Shading:

Rendering is the process of generating a 2D image from a 3D scene, and it heavily relies on linear algebra. Techniques such as ray tracing and rasterization use vector operations to calculate intersections, reflections, and refractions of light. Shading models, including phong and gouraud shading, depends on dot products and cross products to determine light intensity and surface normal, enabling realistic lighting and shadow effects.

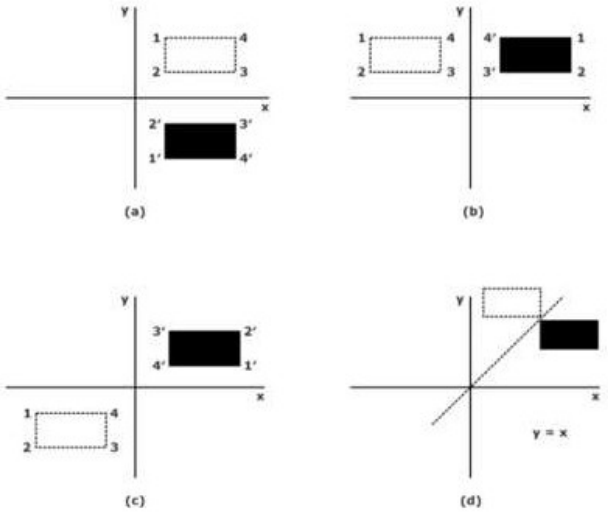
(v) Animations and Simulations:

Animating objects in computer graphics often involves interpolating between states using linear algebra. Key frame animation uses linear transformations to transition objects smoothly from one frame to another. Similarly, simulations of physical phenomena like cloth movement or fluid dynamics rely on solving linear systems of equations to model realistic behaviour. (Outlet, 2025)

1.2 Geometric Transformation of 2D and 3D space:

Transformation means changing some graphics into something else by applying rules. We can have various types of transformations such as **translation, scaling up or down, rotation, shearing,** etc. When a transformation takes place on a 2D plane, it is called 2D transformation. When happen in 3D environment, it is called 3D transformation.

Transformations play an important role in computer graphics to reposition the graphics on the screen and change their size or orientation. These modifications are achieved by applying specific mathematical operations to the coordinates of an object's points or vertices, thereby achieving the desired visual changes.



Common transformations of rectangle in 2D plane.
(Dobrushkin, V. (n.d.))

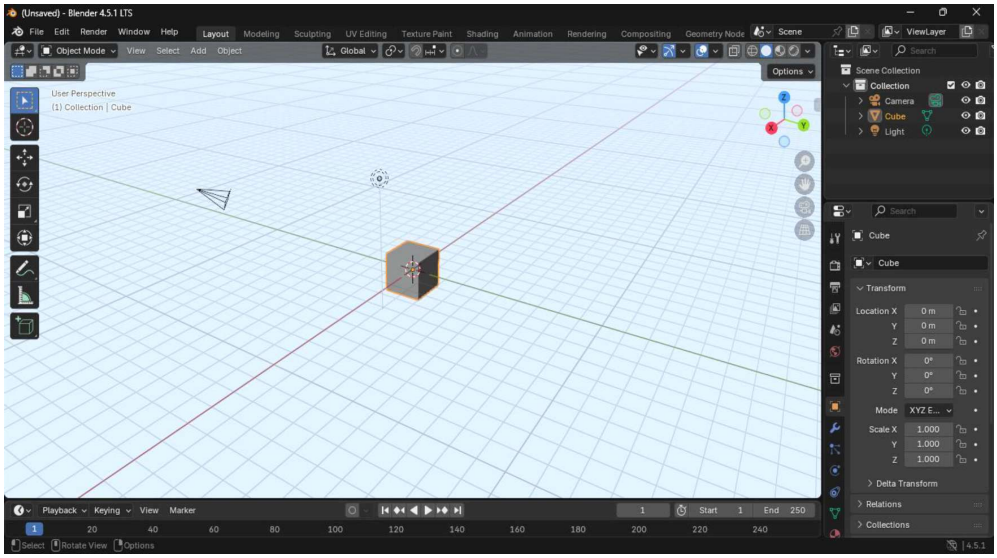
1.3 Homogeneous Coordinates:

Homogeneous coordinates in computer graphics are an extended coordinate system that adds an extra dimension to standard cartesian coordinates, enabling all geometric transformations- including translation, rotation, scaling, and perspective projection to be represented as matrix multiplications. This approach greatly simplifies the implementation and combination of transformations in both 2D and 3D graphics.

In 2D Computer graphics, homogeneous coordinates are used to represent transformations such as rotations, translations, and scaling. The transformation matrices are represented as 3×3 matrices, where the last column represents the translation. (Number Analytics, 2023)

1.4 Implementation of 2D and 3D transformation in Blender:

Blender, a widely adopted open-source 3D creation suite, serves as a prime example of a practical application environment that extensively leverages linear algebra for its core functionalities. Its capabilities span 2D and 3D transformation, animation, rigging, and rendering, with linear algebra forming the mathematical backbone for each of these processes.



Basic Concepts of Linear Algebra for Graphics

2. Vectors: Foundation for Transformations:

- A vector is defined as **"A quantity that has both magnitude and direction."**

In the field of computer graphics, **vectors** are a fundamental mathematical construct used to represent both geometric objects and the transformations applied to them. A vector can be conceptualized in two primary ways: as a **position vector** that defines the location of a point in space relative to an origin, and as a **displacement vector** that represents a change in position, possessing both a specific direction and magnitude. This dual functionality is what makes vectors indispensable for spatial manipulation.

2.1 Translation via Vector Addition:

The most straightforward application of vectors is **translation,** the process of shifting an object from one location to another without altering its size or orientation. This transformation is achieved by performing vector addition. To translate an object, a **translation vector** is added to the position vector of every point on the object. If a point on an object is represented by the position vector $\boldsymbol{P}$**,** and the desired movement is described by the translation vector $\boldsymbol{T}$**,** the new position $\boldsymbol{P'}$ is determined by the vector sum:

$$\boldsymbol{P' = P + T}$$

For a two-dimensional point, this operation is performed component-wise. For example, if a point is at $\boldsymbol{P = (x, y)}$ and the translation vector is $\boldsymbol{T = (tx, ty)}$**,** the new position $\boldsymbol{P'}$ is calculated as:

$$\boldsymbol{P' = (x + tx, y + ty)}$$

This simple yet powerful operation is applied uniformly to all vertices of an object, effectively moving the entire object to a new location within the virtual environment.

3. The Inefficiency of Multiple Operations and the Introduction of Homogeneous Coordinates:

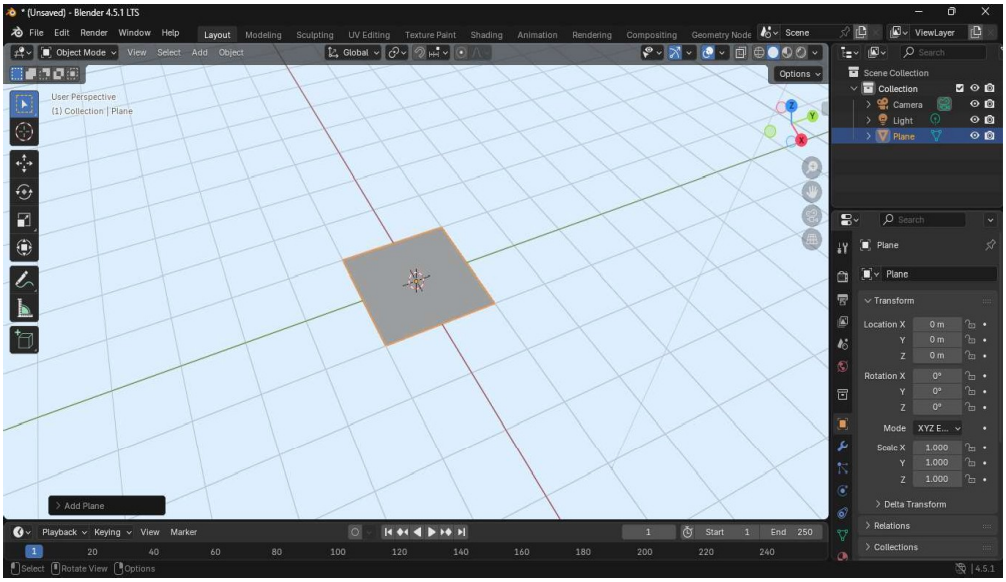
While vector addition is an intuitive method for translation, a more robust and unified system is required for a complete range of graphical transformations.

Other common operations, such as **rotation** and **scaling,** are mathematically performed using vector-matrix multiplication. This creates an inefficiency where translation is a vector addition, while other transformations are a matrix multiplication. This disparity makes it difficult to combine multiple transformations into a single, efficient operation.

To resolve this, computer graphics employs **homogeneous coordinates.** This system extends the vector by one dimension, allowing all transformations-including translation to be represented as a single matrix multiplication. By using a 4×4 transformation matrix for 3D graphics (or a 3×3 for 2D), multiple transformations can be combined into one **composite transformation matrix,** which can be applied to an object's vertices in a single, highly efficient step. This is a critical concept that underpins the performance and flexibility of modern graphics rendering pipelines. The subsequent section will delve into the details of this unified framework.

3.1 2D Geometric transformations:

- Geometric transformations involve altering the spatial characteristics of an image.
- Points and direction in 2D space are represented by vector.
- Geometric objects are collection of these points and are represented by vectors and matrix.
- Type of transformations are
    - Translation
    - Rotation
    - Scaling
    - Shearing
    - Reflection



(Costa, 2023)

- **Translation:**

It involves moving an object from one position to another, it is an affine transformation but can be used in the form of homogeneous coordinates.

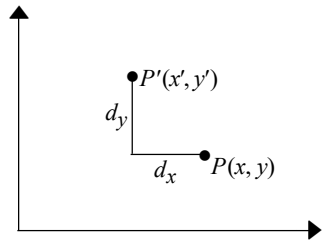
Translation moves an object to a different position on the screen. We can translate a point in 2D by adding translation coordinate (t_x, t_y) to the original coordinate (X, Y) to get the new coordinate (X, Y).

In 2 × 2 matrix form, it can be represented as

$$x' = x + d_x$$
$$y' = y + d_y$$

in matrix form

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



If $T = \begin{bmatrix} d_x \\ d_y \end{bmatrix}$

Then $P' = P + T$

However, this limit us to not allowing to unifying other transformations into single matrix, as other transformation are linear transformations and can be represented by matrix multiplication.

So, in 2D coordinates (x, y), we use **homogeneous coordinates** by adding the extra dimension 'w or 1', so the translation vector, translation matrix, whole transformation becomes

$$\text{Translation vector, } P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

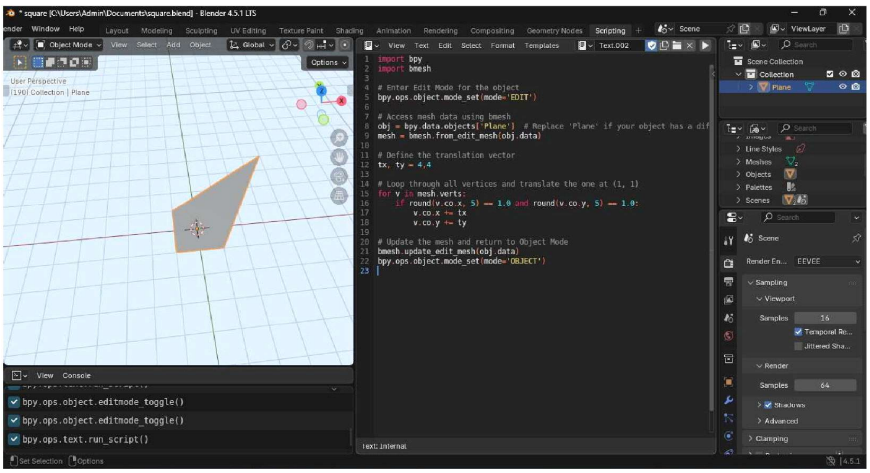
$$\text{Translation matrix, } T = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{Translated point, } \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = P' = P * T = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix}$$

Step-by-step calculation:

- $x' = (1.x) + (0.y) + (T_x.1) = x + T_x$
- $y' = (0.x) + (1.y) + (T_y.1) = y + T$
- $1 = (0.x) + (0.y) + (1.1) = 1$

Example: Translation of point $(1,1)$ of square to $(4,4)$



- **Rotation:**

To rotate a point (x, y) by angle θ. It happens in counterclockwise direction around origin.

This operation changes the orientation of the image. While preserving it shape and size.

In 2 × 2 matrix form, transformation is written as,

$IOP'I = IOPI = l$

$x' = IOP'I \cos(\alpha + \theta) = l \cos(\alpha + \theta)$

$= l \cos\alpha \cos\theta - l \sin\alpha \sin\theta$

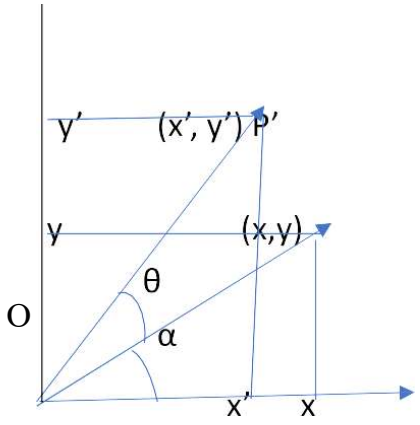
$= x \cos\theta - y \sin\theta$

$y' = IOP'I \sin(\alpha + \theta) = l \sin(\alpha + \theta)$

$= l \cos\alpha \sin\theta + l \sin\alpha \cos\theta$

$= x \sin\theta + y \cos\theta$

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



Here, $R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

$P' = RP$

In homogeneous coordinate form, matrix becomes

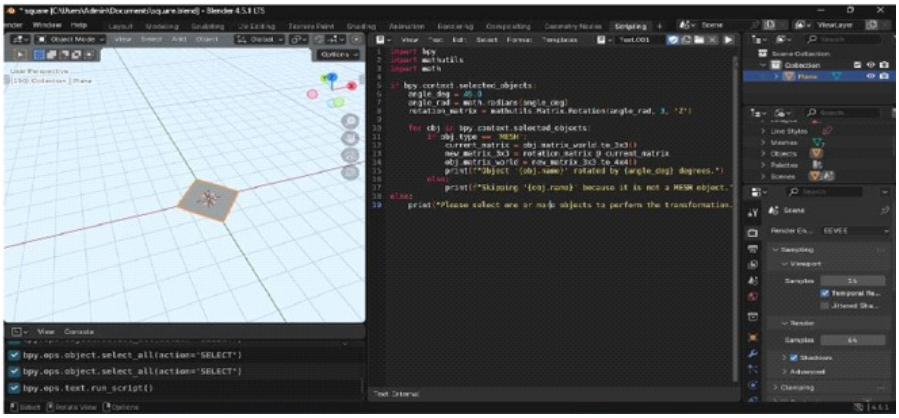
Rotation matrix $R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Rotation transformation $R' = R * P = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

Step-by-step calculation:

- $x' = (x.\cos\theta) - (y.\sin\theta) + (0 \times 1) = x\cos\theta - y\sin\theta$
- $y' = (x.\sin\theta) + (y.\cos\theta) + (0 \times 1) = x\sin\theta + y\cos\theta$
- $1 = (0.x) + (0.y) + (1 \times 1) = 1$

Example: Rotation of square in 45 degrees in counterclockwise direction.



Using the point (1,1,1) (in homogenous coordinates)

$$P = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\theta = 45°$$

$$R = \begin{bmatrix} \cos 45 & -\sin 45 & 0 \\ \sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R' = R * P$$

$$= \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ \sqrt{2} \\ 1 \end{bmatrix}$$

- **Scaling:**

  It refers to stretching or compressing of an object along with its dimension. This is inherently a linear transformation.

  Points can be scaled (stretched) by *Sx* along the *x*-axis and by *Sy* along the *y*-axis into the new points by the multiplications.

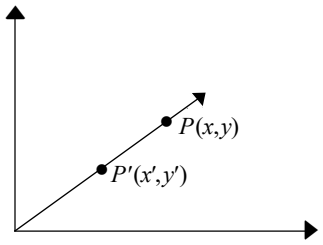
  One can specify how much bigger or smaller by means of a **"scale factor"** (*Sx*, *Sy*).

  In 2 × 2 matrix form, transformation written as

  $$x' = S_x x$$
  $$y' = S_y y$$

  in matrix form

  $$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

  Here, $S = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$

  And $P' = SP$

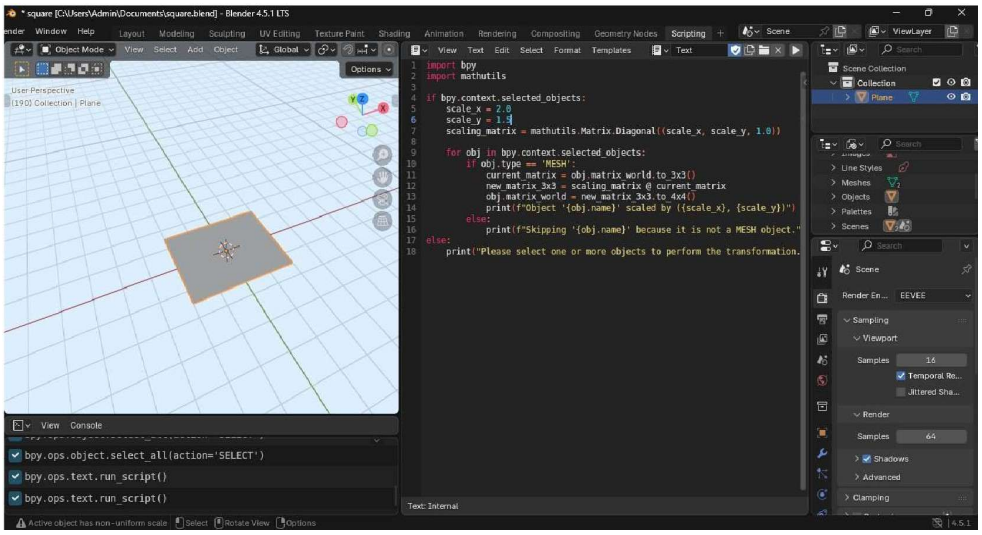
  In homogeneous coordinates system, the matrix will be

  Scaling matrix $S = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$

  Scaled point $P' = S * P = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

  $$x' = (S_x.x) + (0.y) + (0 \times 1) = S_x.x$$
  $$y' = (0.x) + (S_y.y) + (0 \times 1) = S_y.y$$
  $$1 = (0.x) + (0.y) + (1 \times 1) = 1$$

Example: Scaling of square 2 times along *x*-axis and 1.5 times along *y*-axis.



Using the corner of the square at (1, 1, 1) as our initial point vector.

$$P = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$S = \begin{bmatrix} 2.0 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1.0 \end{bmatrix}$$

$$S' = S * P$$

$$= \begin{bmatrix} 2.0 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1.0 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 1.5 \\ 1.0 \end{bmatrix}$$

- **Shearing:**

  It is a transformation that distorts the shape of an object along a certain direction.

  It shifts the coordinates of an object along one axis based on the values of the other axis.

X - shear: shifts point horizontally on y-coordinate
Y - shear: shifts point vertically based on their x-coordinate

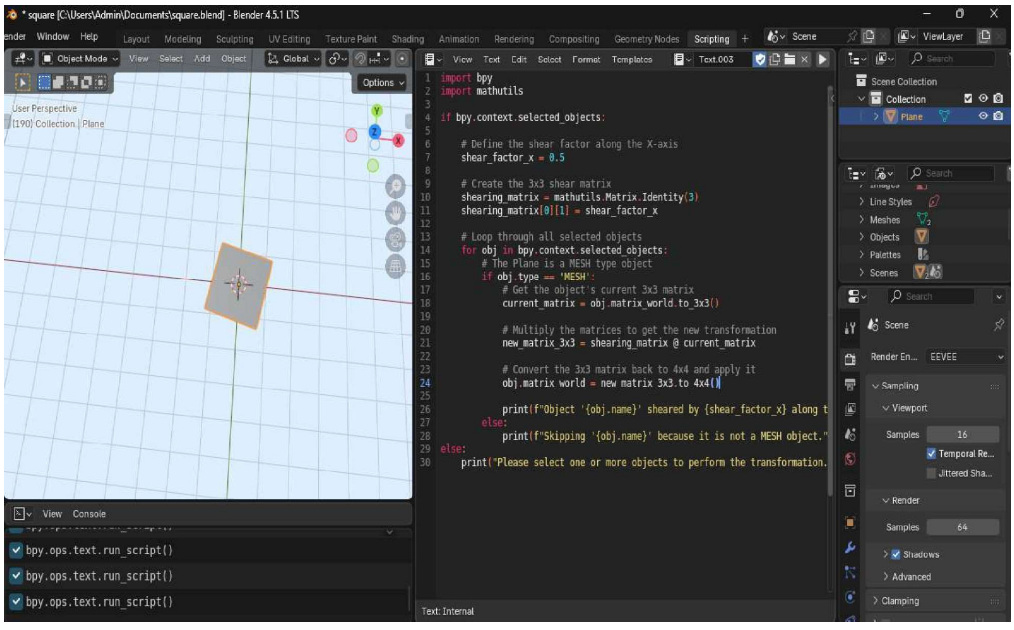
- To shear a point (x, y) by a factor sh_x along x-axis

$$x' = x + sh_x \cdot y$$
$$y' = y$$
$$SH_x = \begin{bmatrix} 1 & shx & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- To shear a point (x, y) by a factor sh_y along y-axis

$$y' = y + sh_y \cdot x$$
$$x' = x$$
$$SH_y = \begin{bmatrix} 1 & 0 & 0 \\ shy & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Example: Shearing of square by factor 0.5 along x-axis



Using the corner of the square at (1, 1, 1) as initial point vector. (in homogeneous coordinate)

$$P = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$SH_x = \begin{bmatrix} 1 & 0.5 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$SH_{x'} = SH_x * P$$

$$= \begin{bmatrix} 1 & 0.5 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1.5 \\ 1 \\ 1 \end{bmatrix}$$

- **Reflection:**

  Reflection is a transformation that produces mirror image of object.

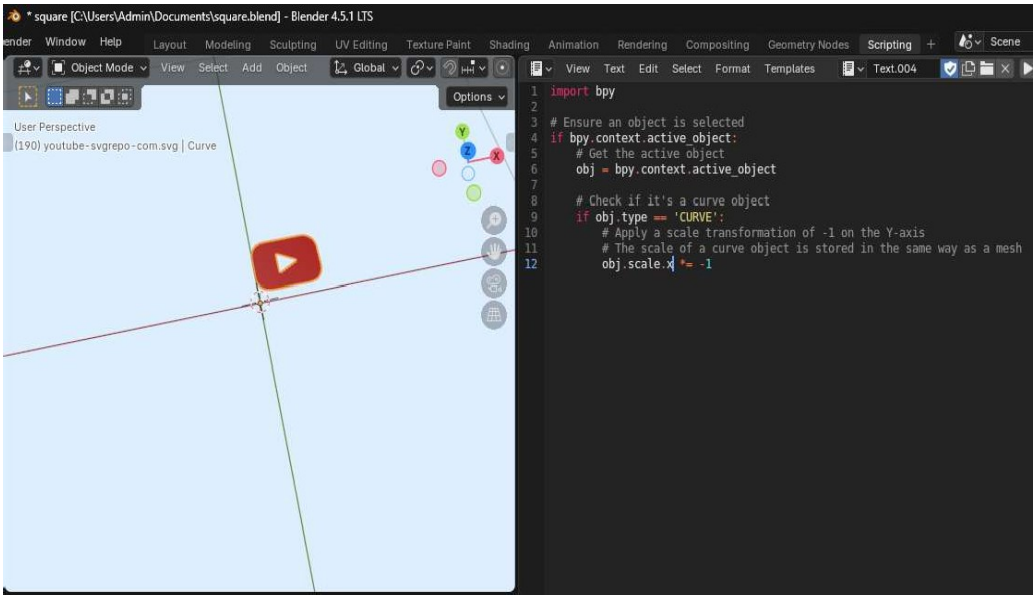
  The mirror image for a two-dimensional reflection is generated relative to an axis of reflection by rotating the object 180 degree about the reflection axis.

  For a reflection about the *x*-axis:

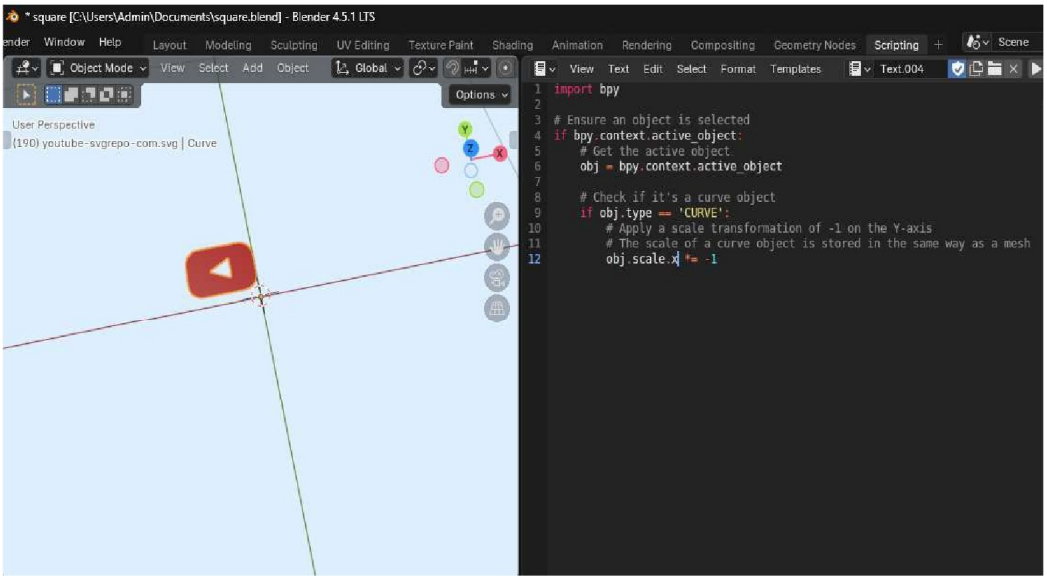
$$\mathrm{Re}f_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{Re}f_y = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Example: "YouTube" logo at origin



After applying transformation on point (1, 1) around *y*-axis



$$P = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\text{Re}f_y = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{Re}f_{y'} = \text{Re}f_y . P$$

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

$$= \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}$$

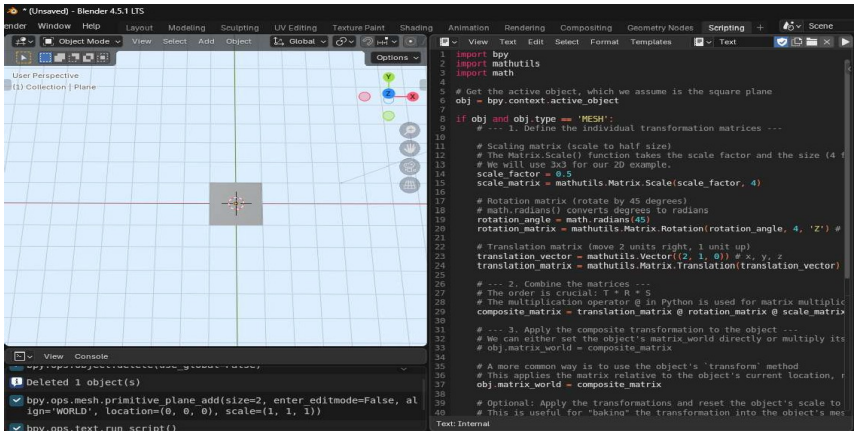
3.2 Composite Transformations:

Composite transformation is the process of applying two or more transformations in a sequence to an object. This approach offers significant advantages in terms of efficiency and simplicity, especially when dealing with complex objects or numerous transformations.

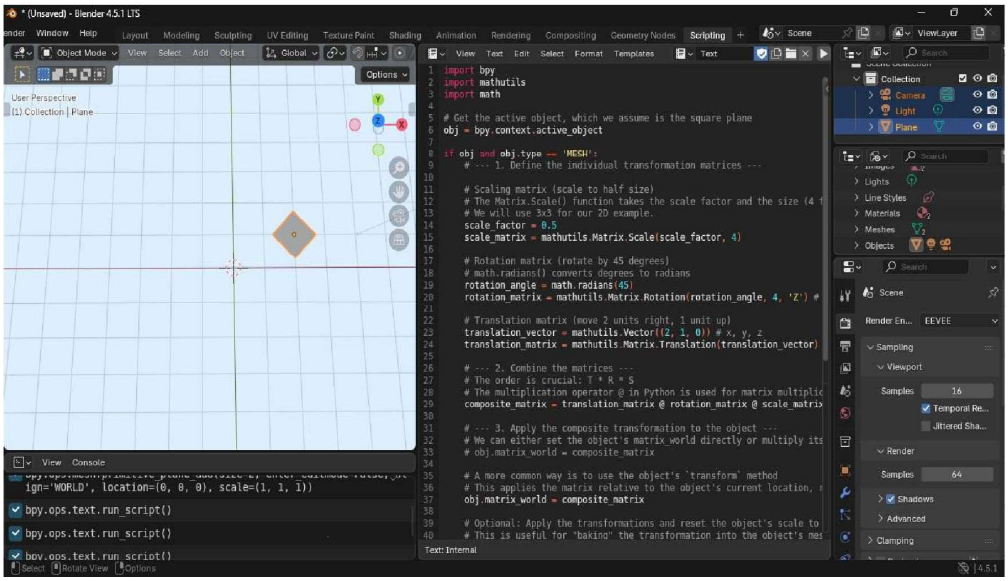
Example: Sequences of transformations on square in Blender:

1. **Scale** the square to 0.5 its size on both the X and Y axes ($S_x = 0.5$, $S_y = 0.5$).
2. **Rotate** the scaled square by 45 degrees around its own centre.
3. **Translate** the rotated square to 2 units to x-axis and 1 unit to y-axis from the origin, so $T_x = 2$, $T_y = 1$. Single matrix, $M = S * R * T$

1) Square at origin:



2) After Transformation:



Initial Point Vector:

Scaling Matrix (S)

$$S = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} \cos 45 & -\sin 45 & 0 \\ \sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Composite applied by first scaling, then rotation, then translation.

$$S * R = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/2\sqrt{2} & -1/2\sqrt{2} & 0 \\ 1/2\sqrt{2} & 1/2\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Now multiply the result by T,

$$M = S * R * T = \begin{bmatrix} 1/2\sqrt{2} & -1/2\sqrt{2} & 0 \\ 1/2\sqrt{2} & 1/2\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/2\sqrt{2} & -1/2\sqrt{2} & 2 \\ 1/2\sqrt{2} & 1/2\sqrt{2} & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

If we apply the composite matrix M to the point $P = (1, 1, 1)$ of square at initial position. The transformed point P' is calculated as

$$P' = M * P$$

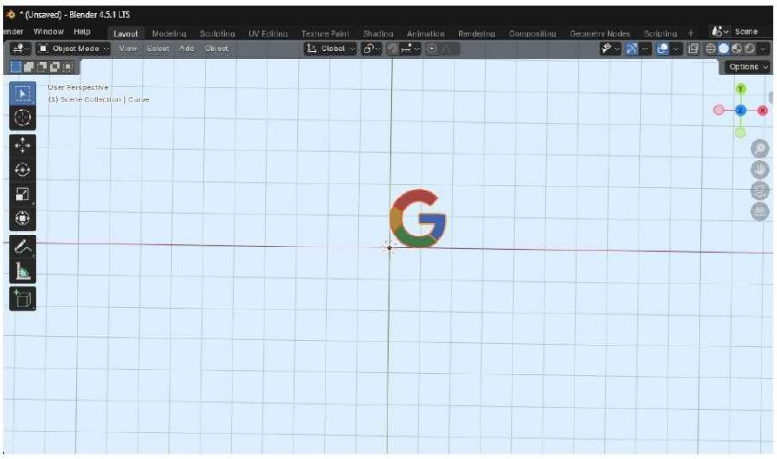
$$P' = \begin{bmatrix} 1/2\sqrt{2} & -1/2\sqrt{2} & 2 \\ 1/2\sqrt{2} & 1/2\sqrt{2} & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$P' = \begin{bmatrix} 1/2\sqrt{2}.1 + (-1/2\sqrt{2}).1 + 2.1 \\ 1/2\sqrt{2}.1 + 1/2\sqrt{2}.1 + 2.1 \\ 0.1 + 0.1 + 1.1 \end{bmatrix}$$

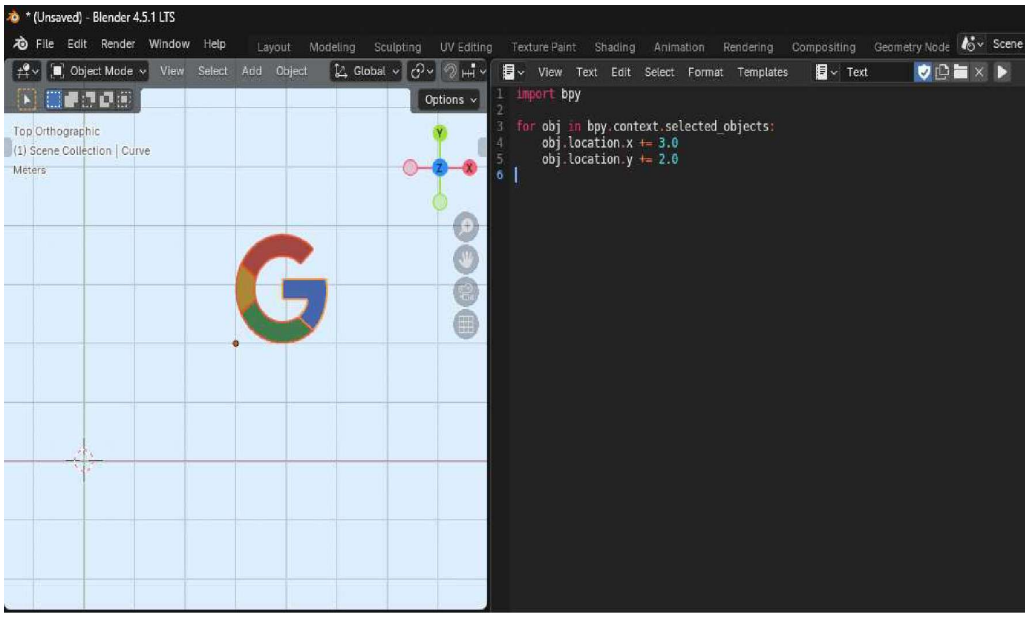
$$P' = \begin{bmatrix} 2 \\ \frac{\sqrt{2}+2}{2} \\ 1 \end{bmatrix}$$

3.3 Example of 2D Transformations:

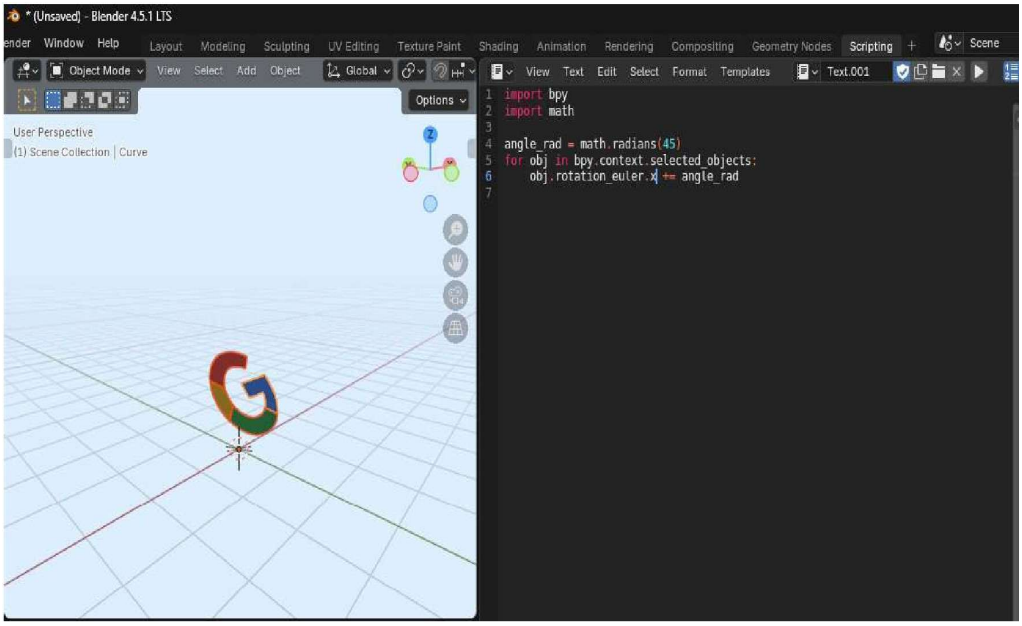
1) Google logo:



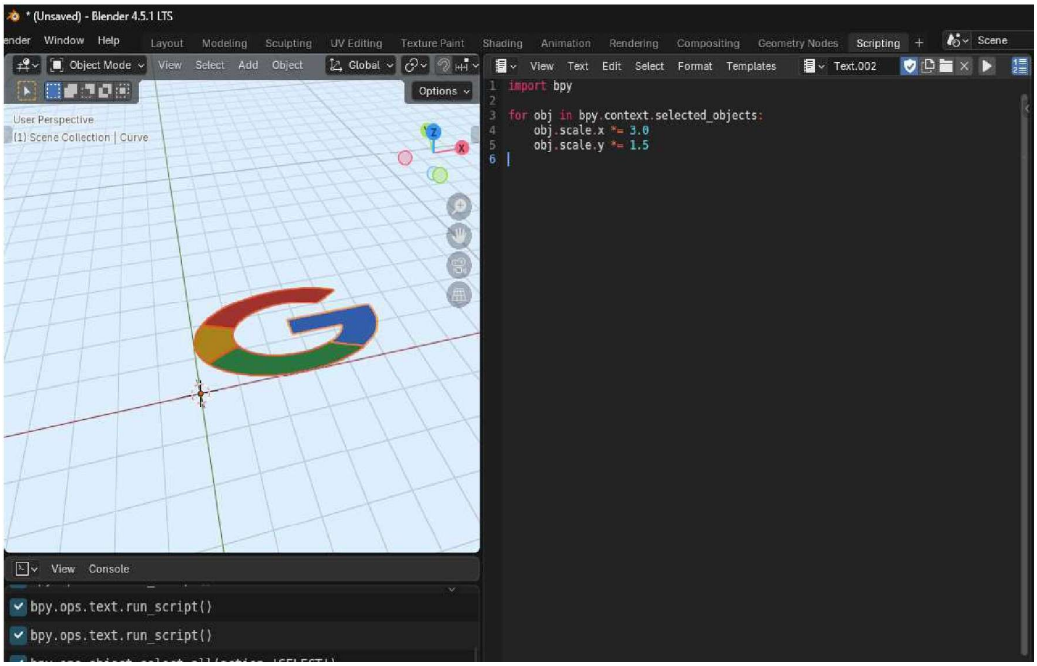
i) **Translation:** Moving object $T_x = 3$ units and $T_y = 2$ units



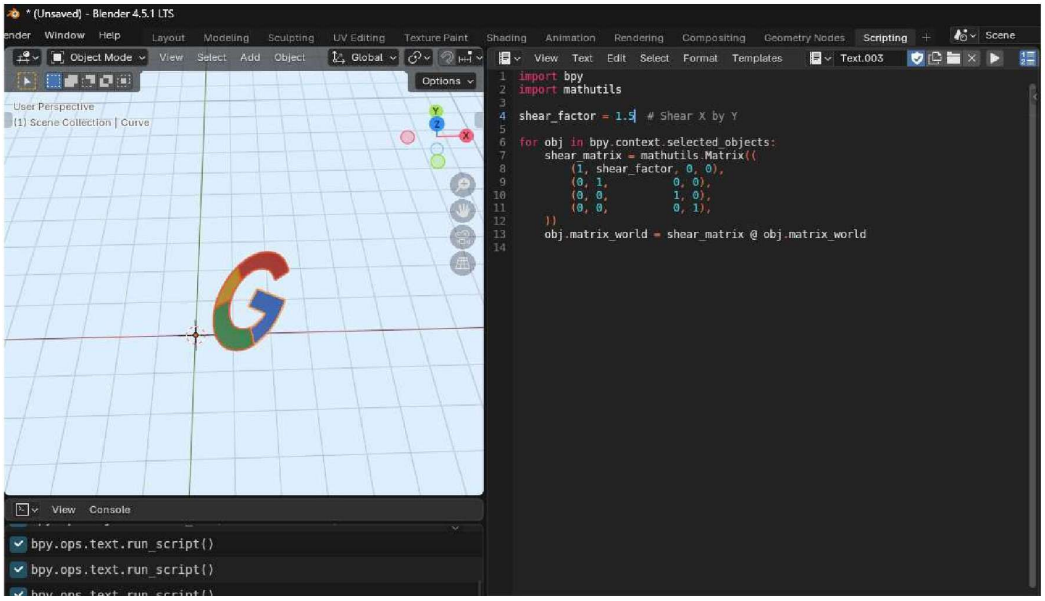
ii) **Rotation:** Rotating 45 degrees in x-axis



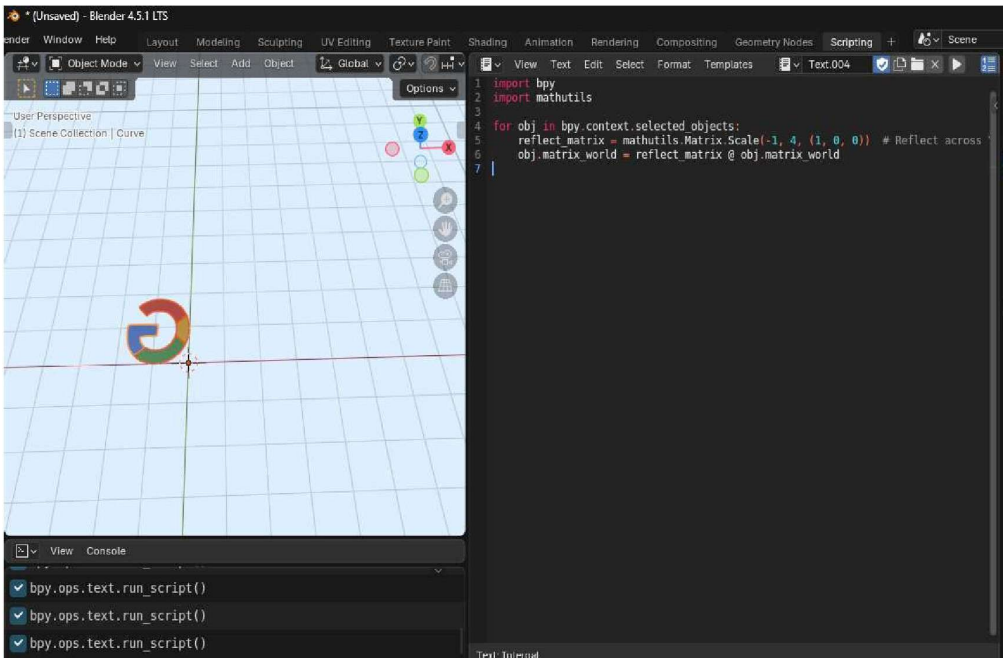
iii) Scaling: Scaling, $S_x = 3.0$ units and $S_y = 1.5$ units



iv) Shearing: Shearing it by factor SH_x

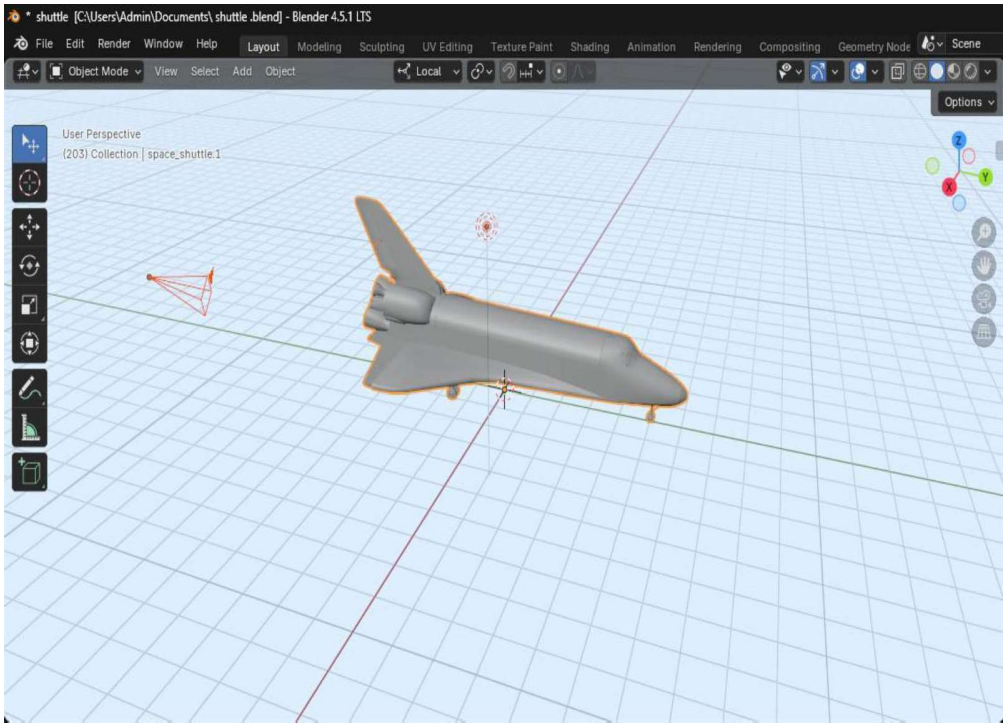


v) **Reflection:** Reflecting object around *y*-axis



4. 3D Geometric Transformations:

- These transformations are for tasks like moving objects, changing their size and shape, and viewing them from different directions.
- They are represented using matrices and applied to the coordinates of the object's vertices.
- In the left-handed system the *x*-axis increases going to the right, the *y*-axis increases going up, and the *z*-axis increases going into the page/screen.
- The right-handed system is the same but with the *z*-axis pointing in the opposite direction.
- Types of transformations are
    - Translation
    - Rotation
    - Scaling



(Pravin, n.d.)

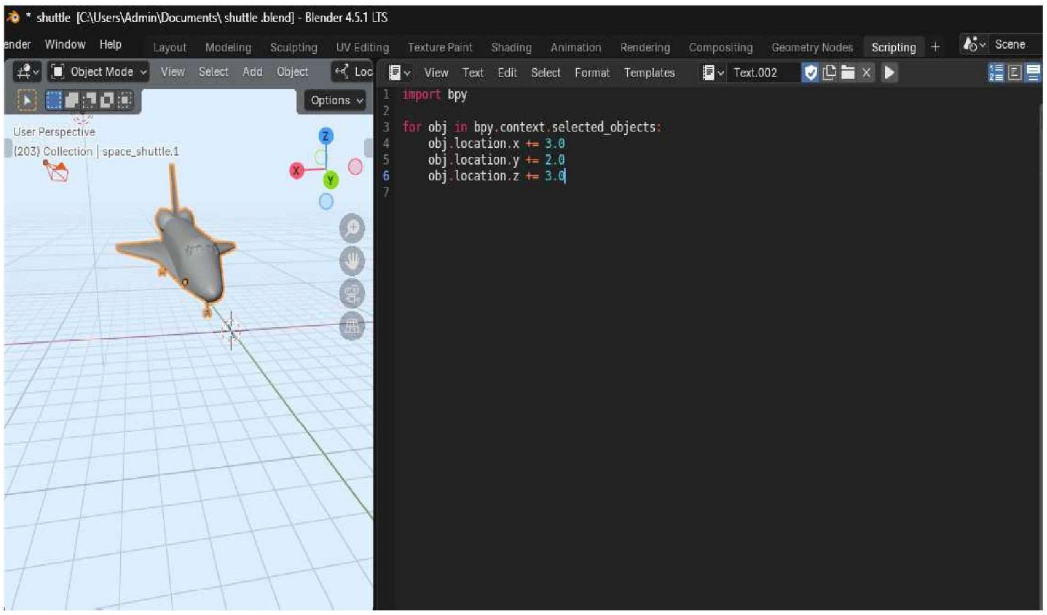
- **Translation:**

  In 3D, each point has three coordinates, x, y and z. Therefore, the translation distances are also of three coordinates.

  They are T_x, T_y and T_z.

  In matrix form, this is represented using a 4 × 4 homogeneous coordinate matrix:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & Tx \\ 0 & 1 & 0 & Ty \\ 0 & 0 & 1 & Tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



- **Rotation:**

  This transformation changes the orientation of an object around a fixed point or axis.

  3D rotation requires specifying both the angle of rotation and the axis of rotation (x, y, or z).

  The order of rotation is important when applying all rotation.

  If angle is Φ then

  $x = r\cos\Phi$ and $y = r\sin\Phi$

  If rotated by θ then:

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

  And

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

  Replacing $r\cos\Phi$ with x and $r\sin\Phi$ with y, we have:

$x' = x\cos\theta - y\sin\theta$

$y' = x\sin\theta + y\cos\theta$

and $z' = z$ (if rotate around z-axis)

Rotation around *X*-axis:

Rotation in the yz-plane

$x' = x$

$y' = y\cos\theta - z\sin\theta$

$z' = y\sin\theta + z\cos\theta$

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

Rotation around *Y*-axis:

Rotation in the xz-plane

$x' = z\sin\theta + x\cos\theta$

$y' = y$

$z' = z\cos\theta - x\sin\theta$

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

Rotation around *Z*-axis:

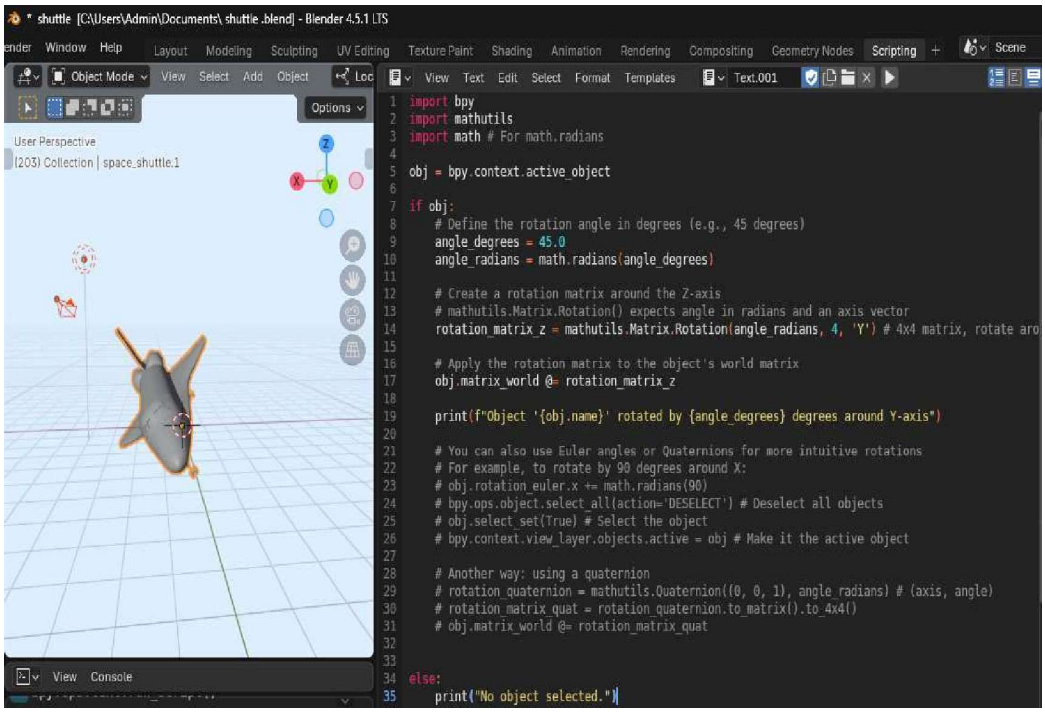
Rotation in the xy-plane

$x' = x\cos\theta - y\sin\theta$

$y' = x\sin\theta + y\cos\theta$

$z' = z$

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

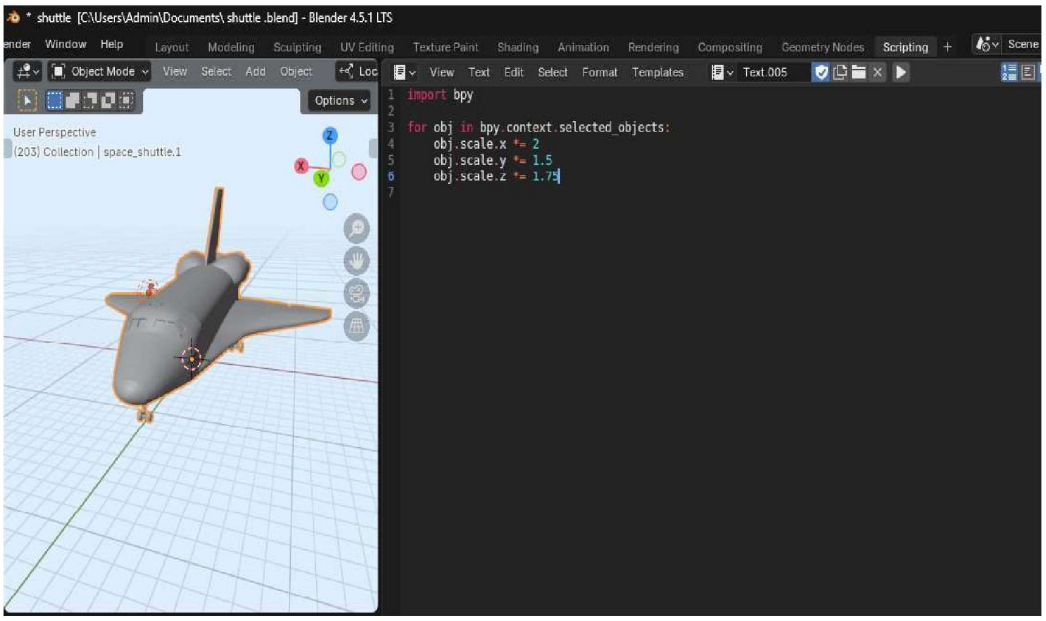


- **Scaling:**

  Scaling changes the size of an object with respect to origin.

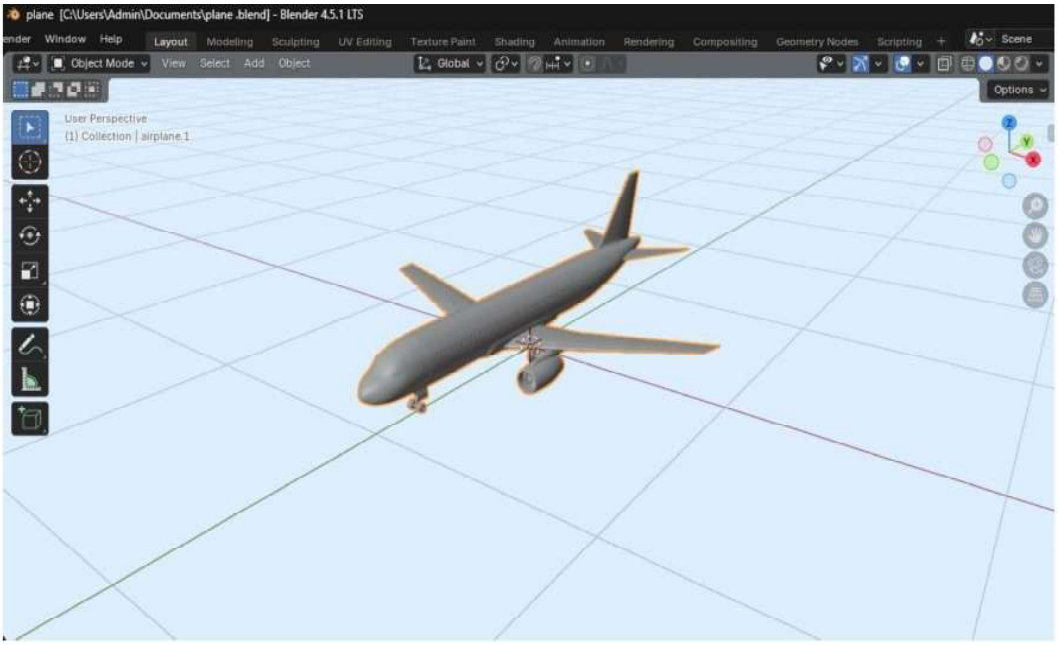
  Scaling can be uniform (keeping the original shape of object) $S_x = S_y = S_z$.

  And it can be non-uniform (does not keep the shape of original object) $S_x <> S_y <> S_z$.

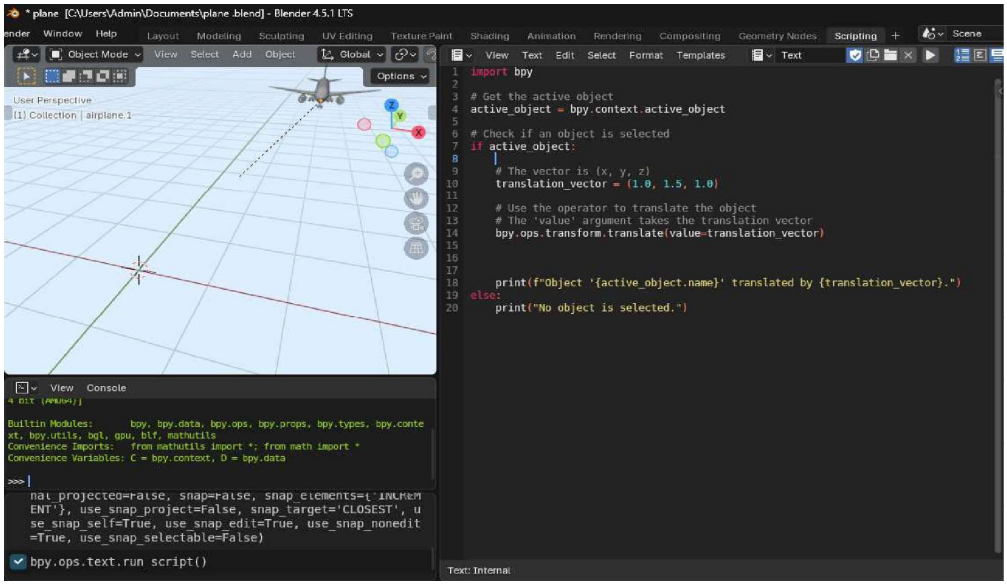


Examples of 3D Transformations:

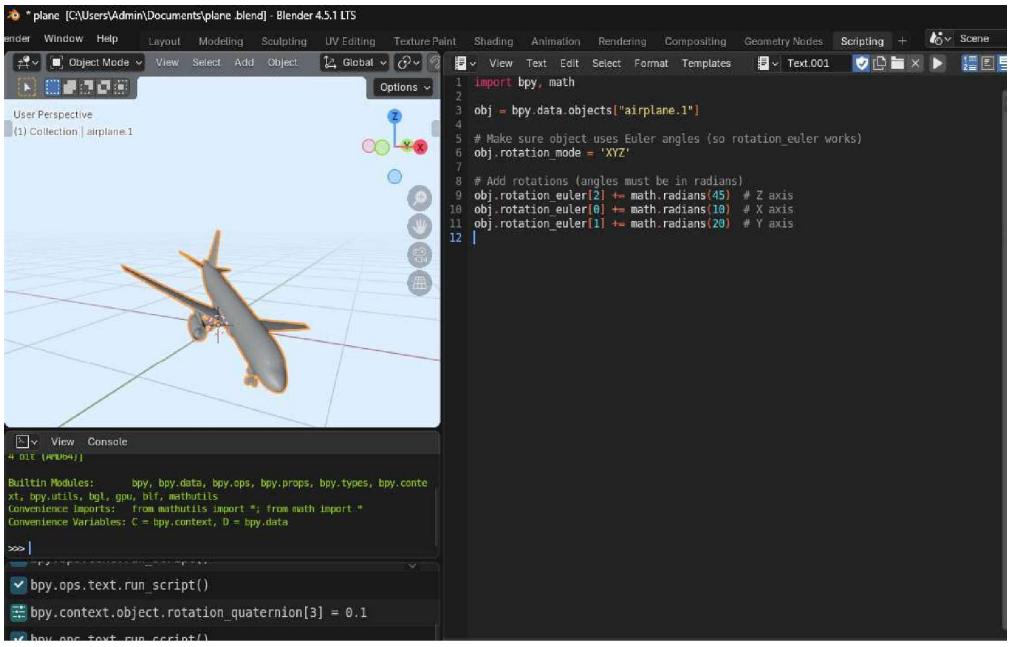
A) Aeroplane:



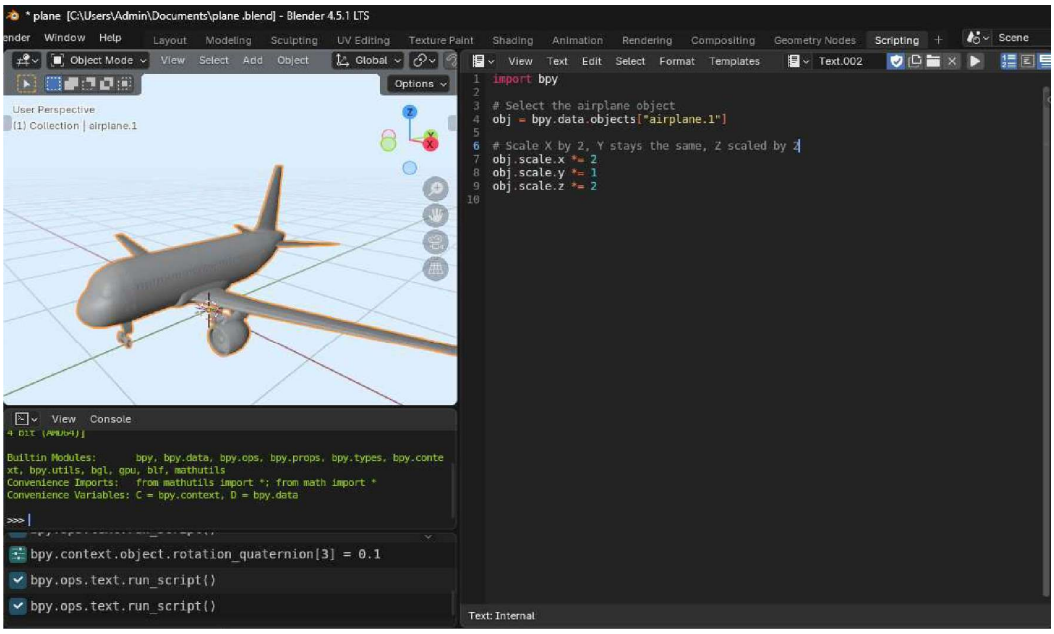
1) Translation:



2) Rotation:

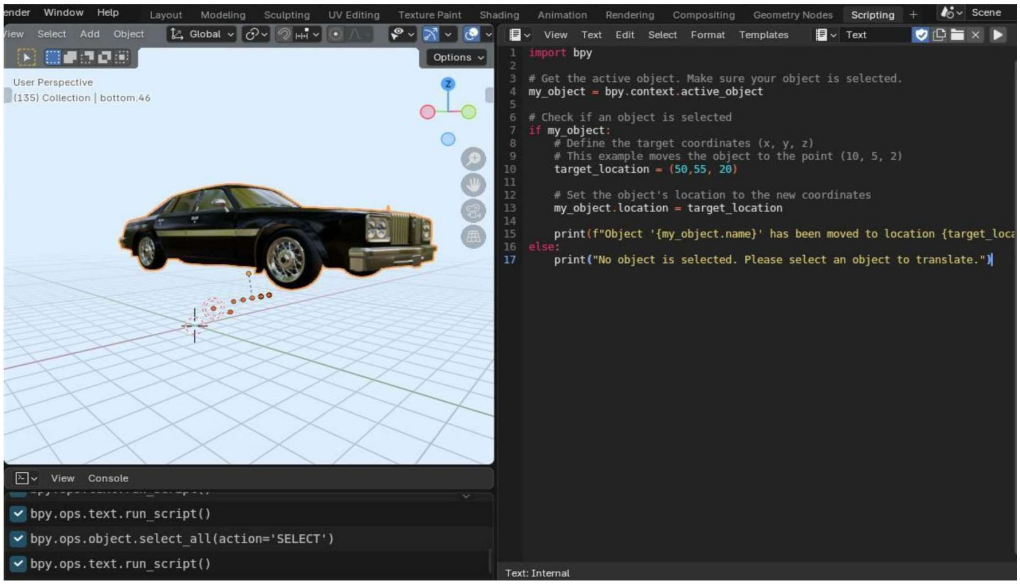


3) Scaling:

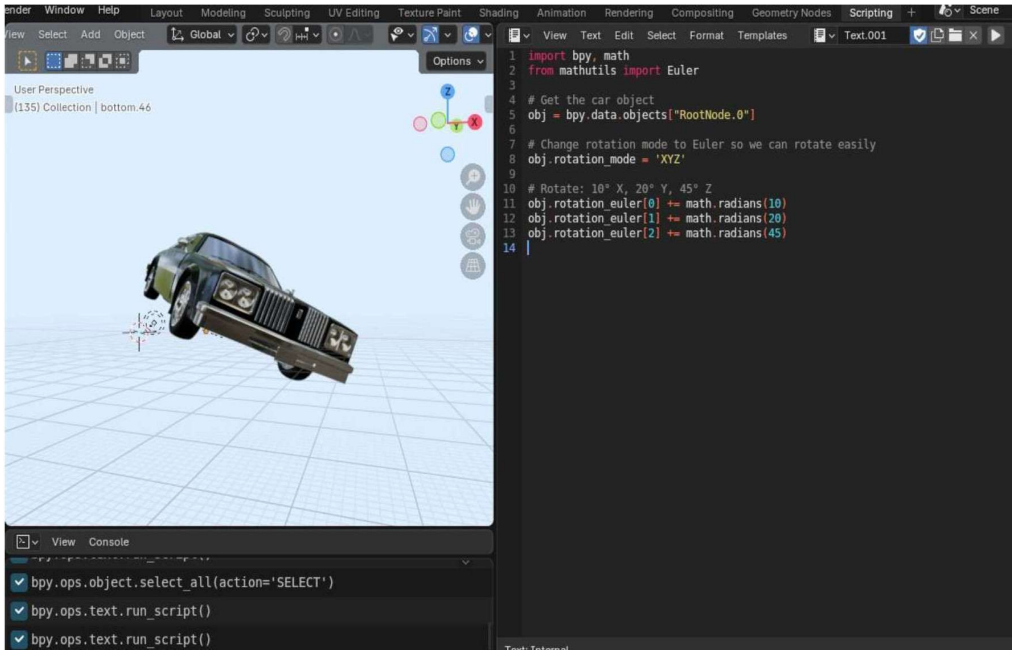


B) Car :

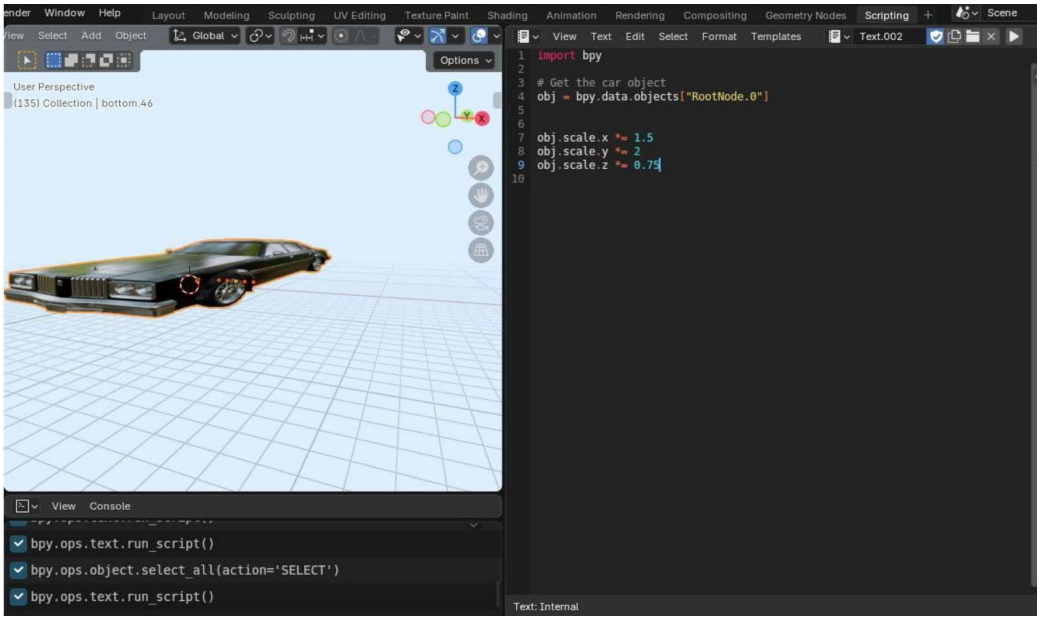
1) Translation:



2) Rotation:



3) Scaling:



(Snyder, 2021)

Acknowledgement:

We would like to express our sincere gratitude to everyone who contributed to the successful completion of **ELITE** research project, **"Application of Linear Algebra in Computer Graphics: 2D And 3D Transformations"**.

We are thankful to **Principal Sir, Prof. Ravi Toteja** for their guidance, encouragement, and feedback. Their expertise was helpful in shaping our understanding and bringing this research to completion.

Conclusion:

Applications of Linear algebra in computer graphics focusing on 2D and 3D transformations has provided concise understanding of principles of digital rendering. Basic concept of Linear algebra (vectors, matrices and transformations) proves to be essential for representation and manipulation of objects in 2D and 3D space. This demonstrates the role of matrix operations in computer graphics. These matrices govern the scaling, translation, rotation, reflection. This investigation has

suggested that the modern computer graphics possible from simple 2D games to complex 3D animations. The successful use of these techniques in games, visual effect, product design and scientific transformation.

References:

- Costa, O.R. (2023, December 8) : Beginner's guide to Linear Transformations and it's applications in computer science. *Medium*, p. n.pag. Retrieved from https://medium.com/@otavioribeiromoreiradacosta/1inear-transformations-applications-in-computer-science-cd85dfa356cb

- Dobrushkin, V. (n.d.) : Computer Graphics (Chapter/Section in Linear Algebra with Mathematica). In V. Dobrushkin, *Linear Algebra with Mathematica* (p. n.pag.). Providence, RI: Brown University, Department of Applied Mathematics. Retrieved from https://www.cfm.brown.edu/people/dobrush/cs52/Mathematica/Part7/graphics.html

- Number Analytics, T. (2023, July 4) : Ultimate Guide to Homogeneous Coordinates in Computer Graphics. *Number Analytics Blog,* p. n.pag. Retrieved from https://www.numberanalytics.com/blog/ultimate-guide-homogeneous-coordinates-computer-graphics#:~:text=Introduction%20to%20Homogeneous%20Coordinates,and%20manipulation%20of%20geometric%20transformations

- Outlet, T.M. (2025, January 9) : Importance of Linear Algebra in Computer Graphics. *The Mathematics Outlet,* p. n. pag.

- Pravin. (n.d.) : 2D and 3D Transformations. *Pravin Hub*.

- Snyder, W. (2021) : Computer Graphics. In W. Snyder, *Linear Algebra, Geometry, and Computation* (p. n. pag.). Boston, MA: Boston University, Computer Science Department. Retrieved from https://www.cs.bu.edu/fac/snyder/cs132-book/L13ComputerGraphics-Spring2021.htm1