Kathmandu University
Department of Computer Science and Engineering
Dhulikhel, Kavre

COMP 342

Lab Report 4

Submitted by
Ashish kumar khatri
Roll No: 28
CE 3rd Year/2nd Sem

Submitted To,

Mr. Dhiraj Shrestha
Department of Computer Science and Engineering

Programming Language : Javascript

Graphics Library : Opengl through webgl api

Title : 2D Translation, rotation, scaling, reflection, shearing

1. 2D translation:

It refers to the process of moving an object or shape from one position to another in a 2-dimensional coordinate system. It involves shifting the object's coordinates by a specified distance in the x and y directions.The translation can be described using a translation vector (tx, ty), where tx represents the translation distance in the x-axis and ty represents the translation distance in the y-axis. Positive values of tx move the object to the right, while negative values move it to the left. Positive values of ty move the object upwards, while negative values move it downwards.

- Initialize an empty list to store the translated vertices.
- For each vertex V in the object:
    - Let Vx be the x-coordinate of the vertex V.
    - Let Vy be the y-coordinate of the vertex V.
    - Compute the translated vertex coordinates:
        - Translated_x = Vx + tx
        - Translated_y = Vy + ty
    - Add the translated vertex (Translated_x, Translated_y) to the list of translated vertices.
- Return the list of translated vertices.

2. 2D rotation:

2D rotation refers to the process of rotating an object or shape in a 2-dimensional coordinate system. It involves changing the angles of the object's vertices while keeping the distances between them constant.

- Initialize an empty list to store the rotated vertices.
- Convert the angle of rotation from degrees to radians if necessary.

- For each vertex V in the object:
    - Let Vx be the x-coordinate of the vertex V.
    - Let Vy be the y-coordinate of the vertex V.
    - Translate the vertex to the origin (center of rotation):
        - Vx -= cx
        - Vy -= cy
    - Compute the rotated vertex coordinates:
        - Rotated_x = Vx * cos(angle) - Vy * sin(angle)
        - Rotated_y = Vx * sin(angle) + Vy * cos(angle)
    - Translate the vertex back to its original position:
        - Rotated_x += cx
        - Rotated_y += cy
    - Add the rotated vertex (Rotated_x, Rotated_y) to the list of rotated vertices.
- Return the list of rotated vertices.

To perform a 2D rotation, you need to specify the angle of rotation and the center of rotation. The center of rotation serves as the anchor point around which the object rotates. The rotation can be clockwise or counterclockwise, depending on the direction of the angle.

3. 2D scaling:

2D scaling refers to the process of resizing an object or shape in a 2-dimensional coordinate system. It involves changing the distances between the object's vertices while maintaining their relative positions.

To perform a 2D scaling, you need to specify the scaling factors for the x and y directions. The scaling factors determine the amount by which the object will be expanded or contracted in each direction.

- Initialize an empty list to store the scaled vertices.
- For each vertex V in the object:
    - Let Vx be the x-coordinate of the vertex V.
    - Let Vy be the y-coordinate of the vertex V.
    - Compute the scaled vertex coordinates:

        - Scaled_x = Vx * sx
        - Scaled_y = Vy * sy
    - Add the scaled vertex (Scaled_x, Scaled_y) to the list of scaled vertices.
- Return the list of scaled vertices.

  This algorithm loops over each vertex of the object and multiplies its x-coordinate by the scaling factor (sx) and its y-coordinate by the scaling factor (sy) to calculate the scaled coordinates. The scaled vertices are stored in a list and returned as the result.

4. 2D reflection:

2D reflection, also known as mirror reflection, refers to the transformation of an object or shape across a mirror line or axis. It produces a mirror image of the original object, preserving its shape and size but changing the orientation.

To perform a 2D reflection, you need to specify the mirror line or axis across which the reflection will occur. The mirror line can be vertical, horizontal, or inclined at any angle.

- ReflectX:
    - reflectXMatrix =

      [1 0]

      [0 -1]
    - If V (x,y) is a vertex, then V(x,y)*reflectXMatrix is the final reflectedX vertex.
- ReflectY:
    - reflectYMatrix =

      [-1 0]

      [0 1]
    - If V (x,y) is a vertex, then V(x,y)*reflectYMatrix is the final reflectedX vertex.

5. 2D shearing:

To perform a 2D shearing transformation, you can use the following matrices depending on the type of shearing you want to apply:

- **Horizontal Shearing:**

  To shear a 2D object horizontally, you can use the following shearing matrix:

  [1 shx]

  [0   1]

- **Vertical Shearing:**

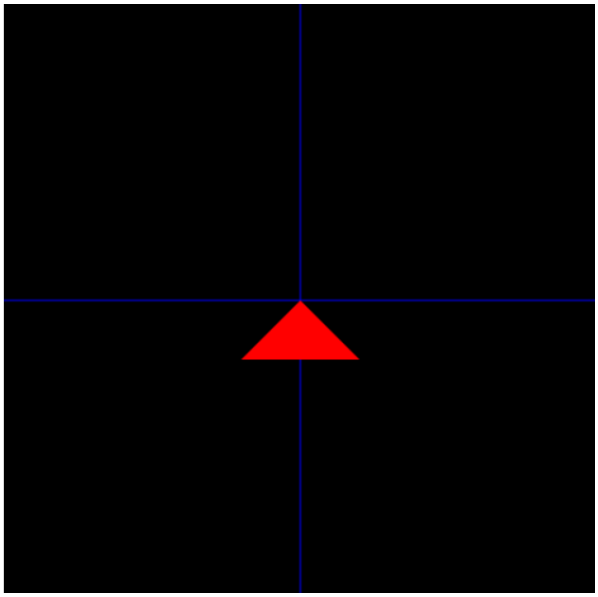
  To shear a 2D object vertically, you can use the following shearing matrix:

  [1   0]

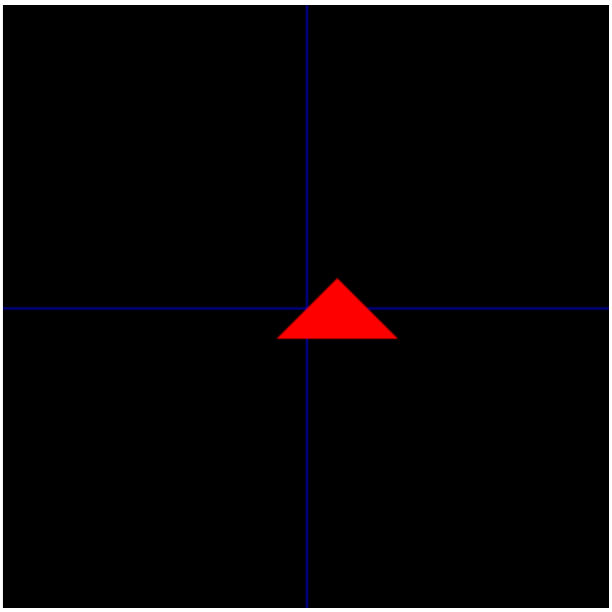
  [shy 1]

ScreenShots:

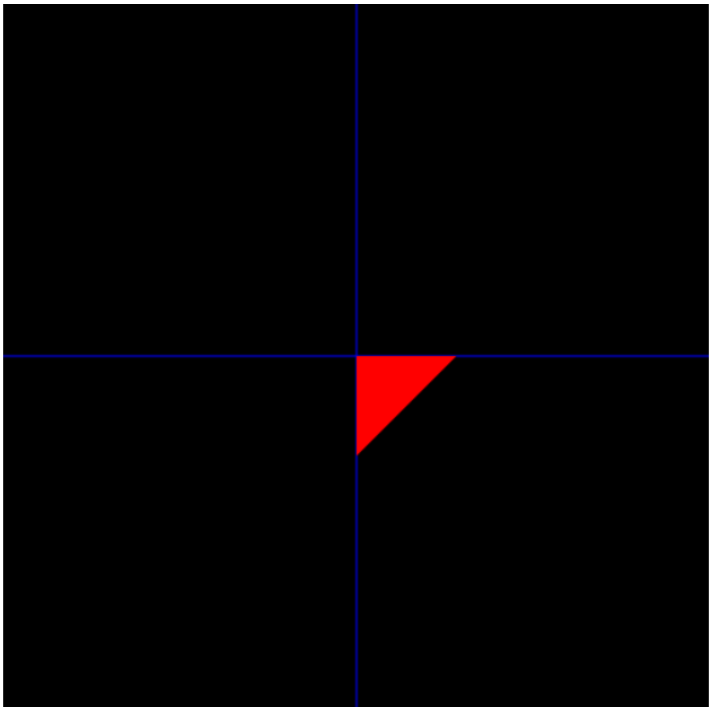
- **Original Position:**



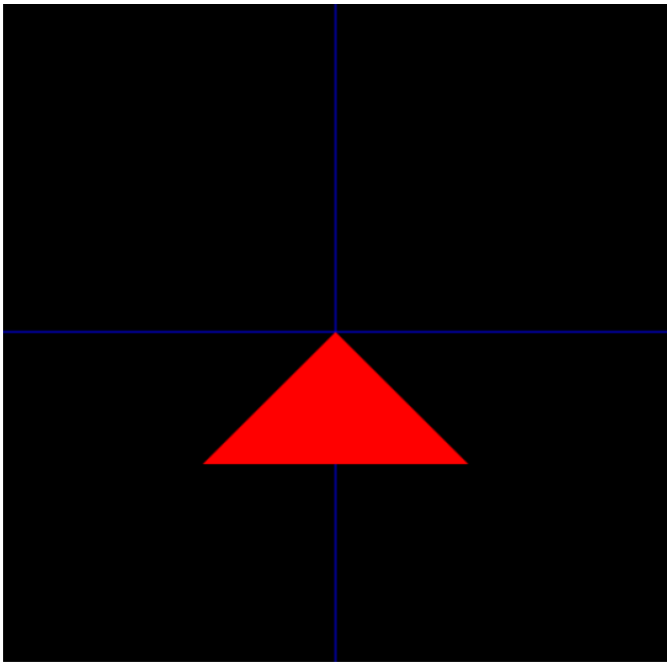
- **Translate by 0.1,0.1**



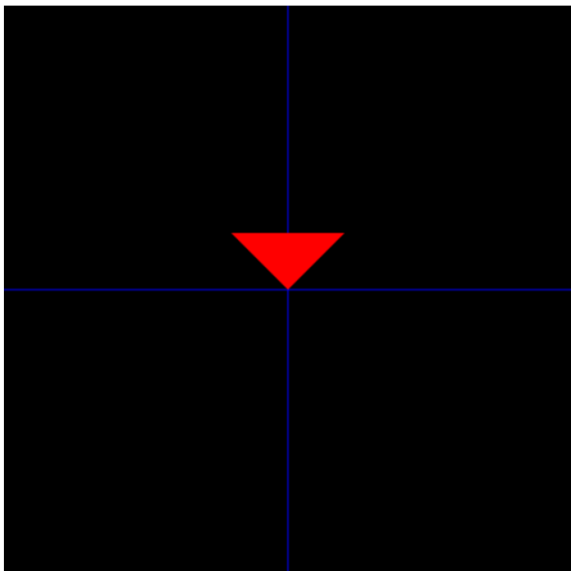
- **Rotate by 45 deg anticlockwise about origin**



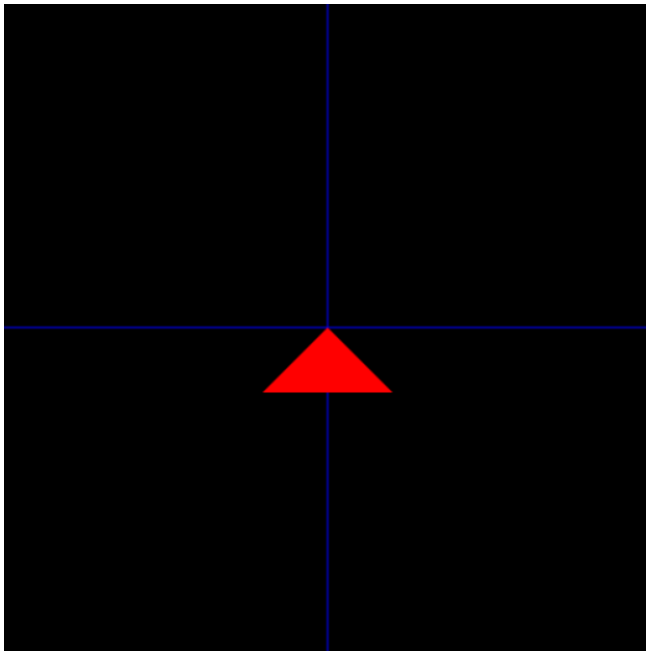
- **Scale by factor (2,2)**



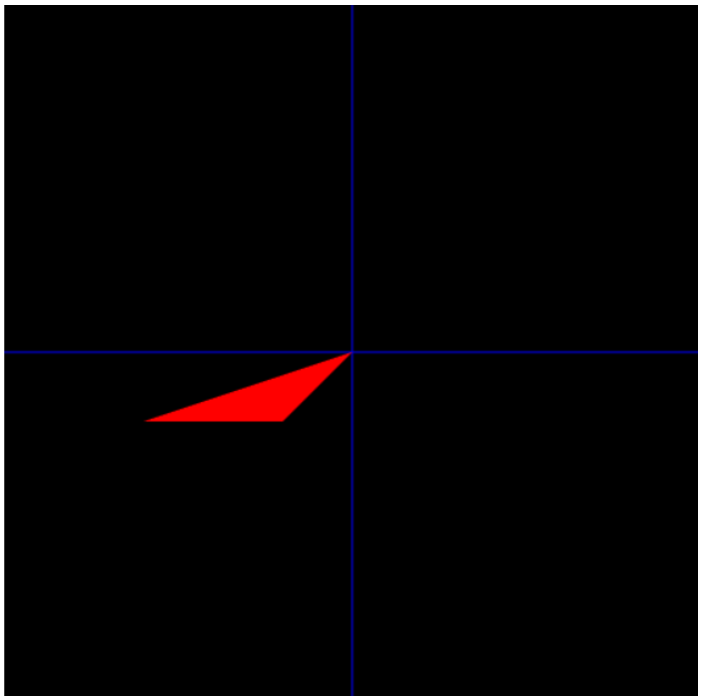
- **reflectX**



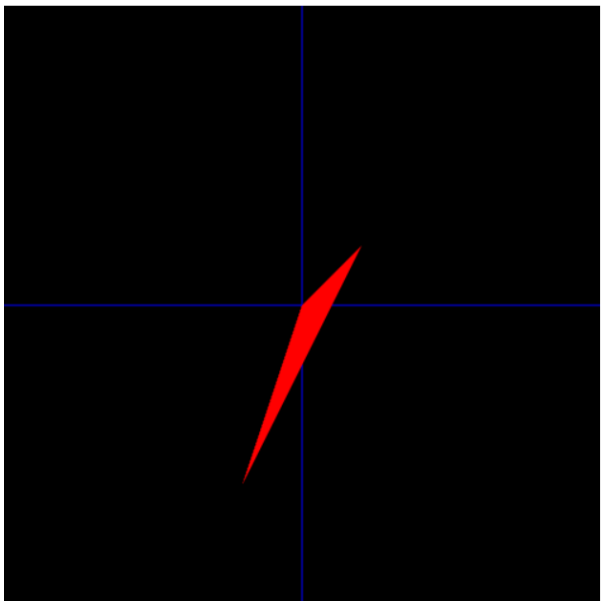
- **reflectY**



- **shearX**



- **shearY**



Conclusion:

So, by using webgl, different 2D transformation algorithm was implemented in lab 4.