EXPERIMENT 6B

COMPUTER GRAPICS AND MULTIMEDIA

Aim

Write C Programs for the implementation of 2D and 3D transformations.

Syeda Reeha Quasar
14114802719
3C7

EXPERIMENT 6B

AIM:

Write C Programs for the implementation of 2D and 3D transformations.

THEORY:

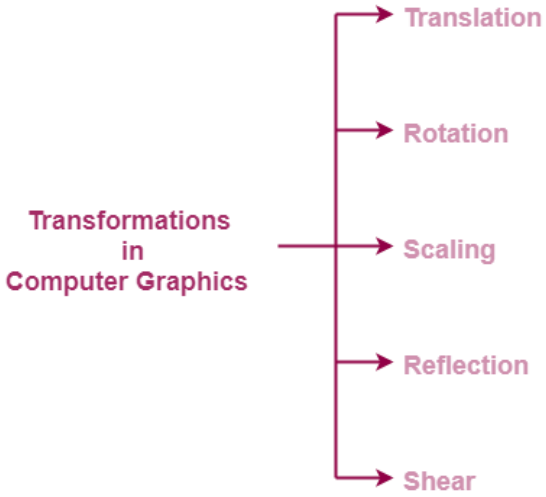
2D TRANSFORMATIONS

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.

Transformations play an important role in computer graphics to reposition the graphics on the screen and change their size or orientation.

In computer graphics, various transformation techniques are-

1. Translation
2. Rotation
3. Scaling
4. Reflection
5. Shear



In this experiment, we will discuss about 2D Reflection and Shearing in Computer Graphics.

2D Reflection in Computer Graphics-

- Reflection is a kind of rotation where the angle of rotation is 180 degree.
- The reflected object is always formed on the other side of mirror.
- The size of reflected object is same as the size of original object.

Consider a point object O has to be reflected in a 2D plane.

Let-

- Initial coordinates of the object O = (X_{old}, Y_{old})
- New coordinates of the reflected object O after reflection = (X_{new}, Y_{new})

Reflection On X-Axis:

This reflection is achieved by using the following reflection equations-

- $X_{new} = X_{old}$
- $Y_{new} = -Y_{old}$

In Matrix form, the above reflection equations may be represented as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

Reflection Matrix

(Reflection Along X Axis)

For homogeneous coordinates, the above reflection matrix may be represented as a 3 x 3 matrix as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ 1 \end{bmatrix}$$

Reflection Matrix

(Reflection Along X Axis)

(Homogeneous Coordinates Representation)

Reflection On Y-Axis:

This reflection is achieved by using the following reflection equations-

- $X_{new} = -X_{old}$
- $Y_{new} = Y_{old}$

In Matrix form, the above reflection equations may be represented as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

Reflection Matrix

(Reflection Along Y Axis)

For homogeneous coordinates, the above reflection matrix may be represented as a 3 x 3 matrix as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ 1 \end{bmatrix}$$

Reflection Matrix

(Reflection Along Y Axis)

(Homogeneous Coordinates Representation)

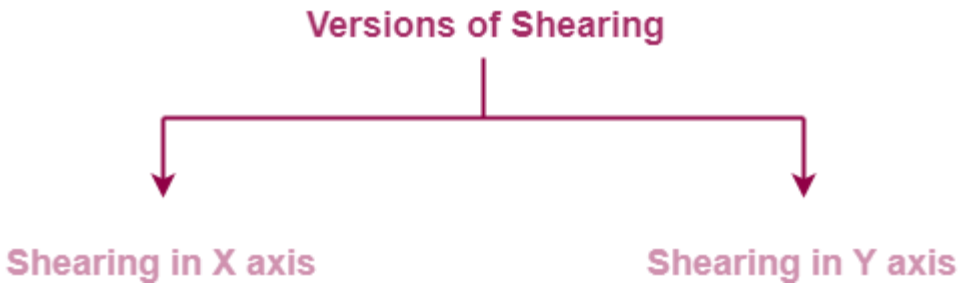
2D Shearing in Computer Graphics-

In Computer graphics,

2D Shearing is an ideal technique to change the shape of an existing object in a two dimensional plane.

In a two dimensional plane, the object size can be changed along X direction as well as Y direction.

So, there are two versions of shearing-



1. Shearing in X direction
2. Shearing in Y direction

Consider a point object O has to be sheared in a 2D plane.

Let-

- Initial coordinates of the object O = (X_{old}, Y_{old})
- Shearing parameter towards X direction = Sh_x
- Shearing parameter towards Y direction = Sh_y
- New coordinates of the object O after shearing = (X_{new}, Y_{new})

Shearing in X Axis-

Shearing in X axis is achieved by using the following shearing equations-

- $X_{new} = X_{old} + Sh_x \times Y_{old}$
- $Y_{new} = Y_{old}$

In Matrix form, the above shearing equations may be represented as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 1 & Sh_x \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

Shearing Matrix

(In X axis)

For homogeneous coordinates, the above shearing matrix may be represented as a 3 x 3 matrix as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ Sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ 1 \end{bmatrix}$$

Shearing Matrix

(In X axis)

(Homogeneous Coordinates Representation)

Shearing in Y Axis-

Shearing in Y axis is achieved by using the following shearing equations-

- $X_{new} = X_{old}$
- $Y_{new} = Y_{old} + Sh_y \times X_{old}$

In Matrix form, the above shearing equations may be represented as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ Sh_y & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

Shearing Matrix

(In Y axis)

For homogeneous coordinates, the above shearing matrix may be represented as a 3 x 3 matrix as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & Sh_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ 1 \end{bmatrix}$$

Shearing Matrix

(In Y axis)

(Homogeneous Coordinates Representation)

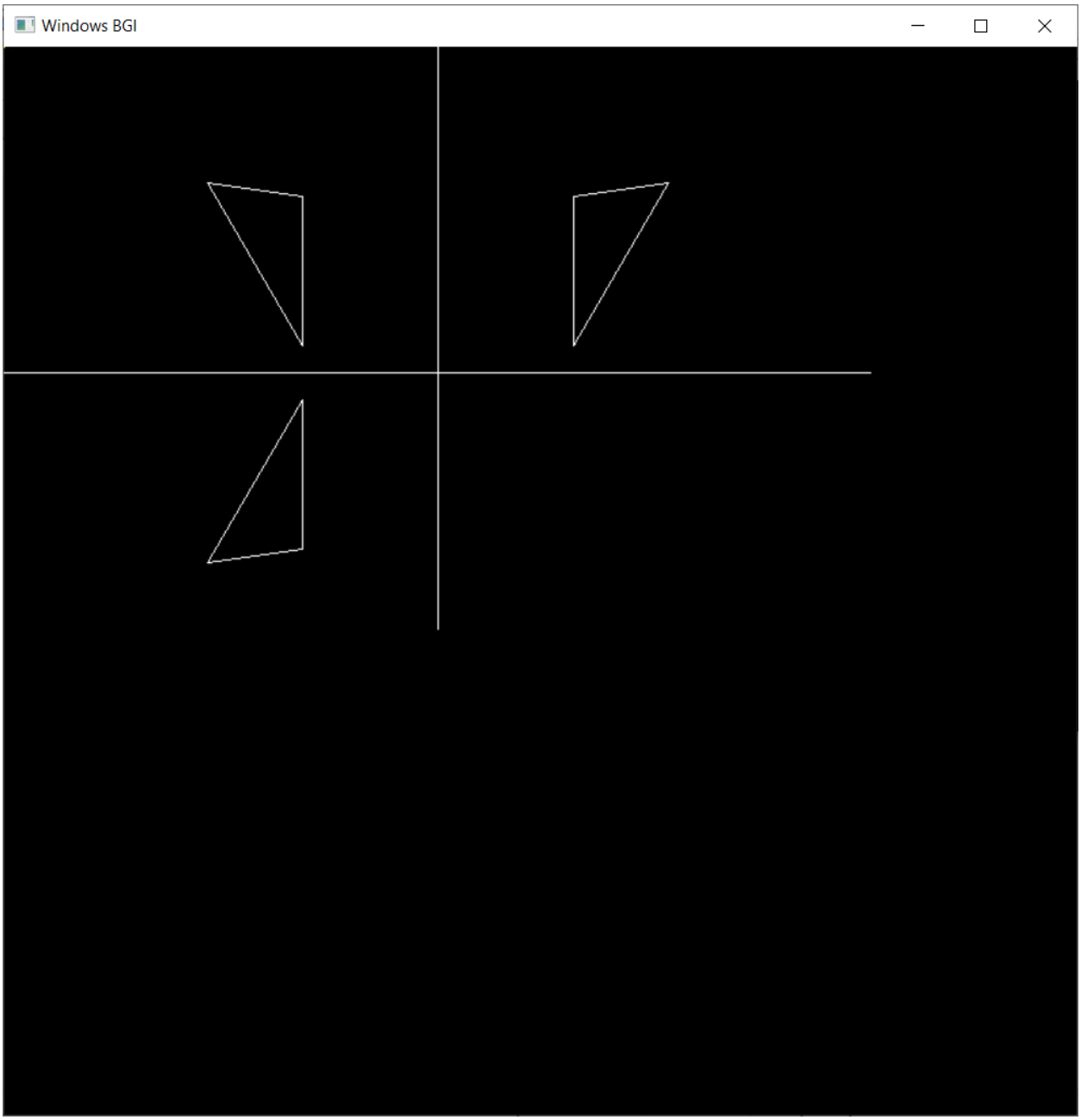
Reflection:

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<stdlib.h>

void refx(int x1,int x2,int x3,int y1,int y2,int y3){
        line(320,0,320,430);
        line(0,240,640,240);
        x1=(320-x1)+320;
        x2=(320-x2)+320;
        x3=(320-x3)+320;
        line(x1,y1,x2,y2);
        line(x2,y2,x3,y3);
        line(x3,y3,x1,y1);
}

void refy(int x1,int x2,int x3,int y1,int y2,int y3){
        line(320,0,320,430);
        line(0,240,640,240);
        y1=(240-y1)+240;
        y2=(240-y2)+240;
        y3=(240-y3)+240;
        line(x1,y1,x2,y2);
        line(x2,y2,x3,y3);
        line(x3,y3,x1,y1);
}
```

```
int main()
{
	int x1,y1,x2,y2,x3,y3;
	initwindow(800, 800);

	//axis
	line(320,0,320,430);
	line(0,240,640,240);

	x1=150;y1=100;
	x2=220;y2=220;
	x3=220;y3=110;

	// triangle
	line(x1,y1,x2,y2);
	line(x2,y2,x3,y3);
	line(x3,y3,x1,y1);

	refx(x1,x2,x3,y1,y2,y3);

	refy(x1,x2,x3,y1,y2,y3);
	getch();

	closegraph();
	return 0;
}
```

OUTPUT:



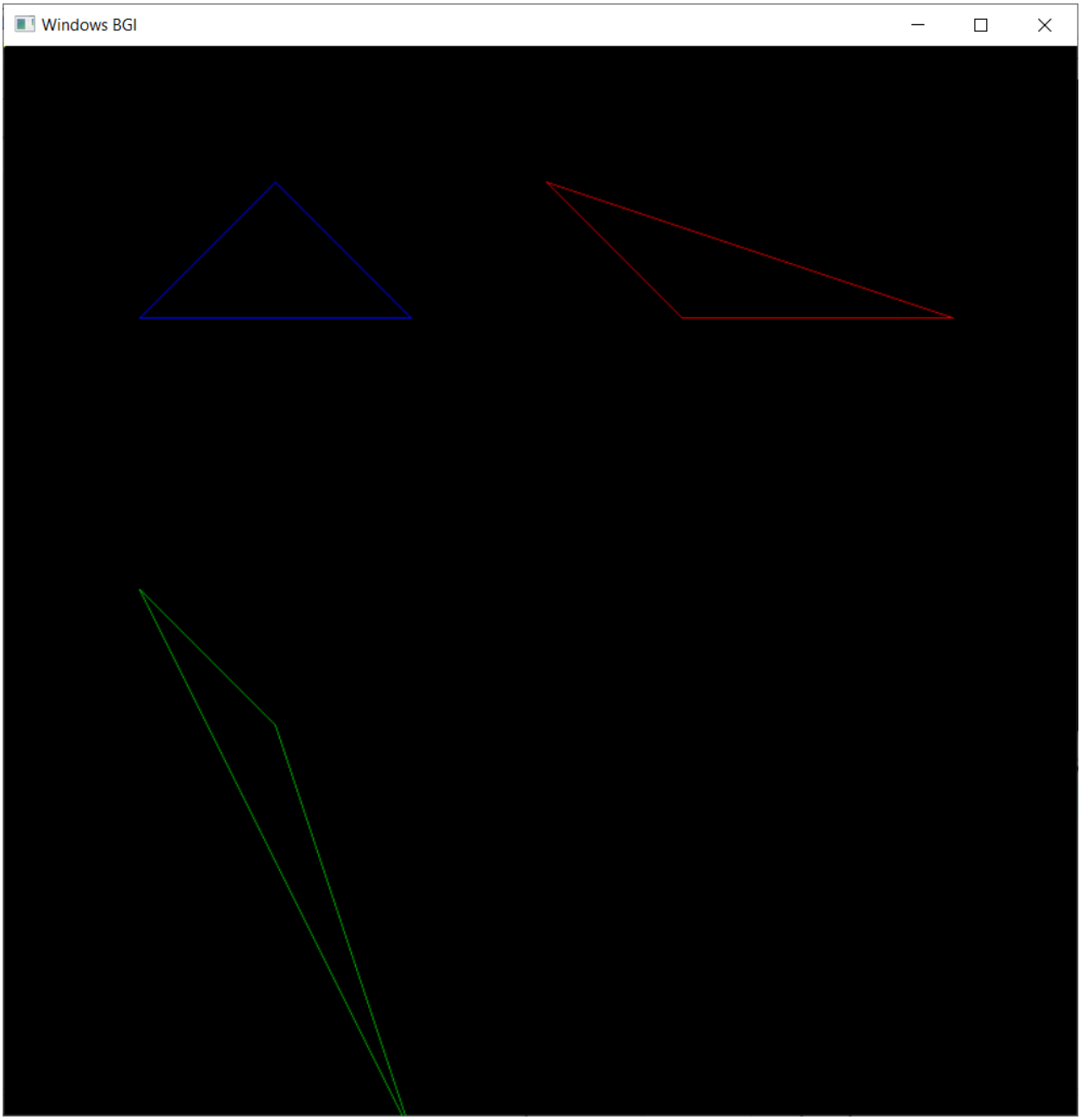
Shearing:

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<stdlib.h>

void shearx(int x,int x1,int x2,int y,int y1,int y2,int shearf){
        setcolor(RED);
        x=x+ y*shearf;
        x1=x1+ y1*shearf;
        x2=x2+ y2*shearf;
        line(x,y,x1,y1);
        line(x1,y1,x2,y2);
        line(x2,y2,x,y);
}

void sheary(int x,int x1,int x2,int y,int y1,int y2,int shearf){
        setcolor(GREEN);
        y=y+ x*shearf;
        y1=y1+ x1*shearf;
        y2=y2+ x2*shearf;
        line(x,y,x1,y1);
        line(x1,y1,x2,y2);
        line(x2,y2,x,y);
```

```
}

int main()
{
        int x,y,x1,y1,x2,y2,shearf;
        initwindow(800,800);

        x=100;y=200;
        x1=200;y1=100;
        x2=300;y2=200;
        shearf=2;
        setcolor(BLUE);
        line(x,y,x1,y1);
        line(x1,y1,x2,y2);
        line(x2,y2,x,y);

        shearx(x,x1,x2,y,y1,y2,shearf);
        sheary(x,x1,x2,y,y1,y2,shearf);
        getch();

        closegraph();
        return 0;
}
```

OUTPUT:



THEORY:

3D TRANSFORMATIONS

The Move Tool

Moves selected object or components by dragging the transform manipulator.

The move tool is located in the toolbox. The Move tool has a manipulator that will become visible when an object is selected. The manipulator consists of arrows on the three axes which are color coded (Red = X, Green = Y, Blue = Z) and a yellow box at the center of the manipulator. To use it, first choose an object you need to move. Upon selecting it, you will see a controller appeared with four handles. They are used to move the object along each axis and one to move along the planes. Also, the colour changes to indicate that it is active when you select a handle.

The Rotate Tool

Rotates selected object or components by dragging the rotation manipulator.

The rotate tool is used to rotate objects in all the three axes. Select the rotate tool available in the toolbox and select the object you need to rotate. Now you will notice four rings colour coordinated to XYZ axes. A virtual sphere is also displayed along with the rings. You will know the selected ring by the change in colour. To perform constrained rotations, use X, Y, Z rings. In order to rotate according to the view, use the outer ring. When you start rotating the object, the application rotates it based on the object's bounding box. If you need to rotate it in fixed increments, then you can make use of the snap option. For e.g., if you set the rotation degree to 15, then you can easily rotate the objects in order of 30, 45, 60, and 90-degree positions for better symmetry. The snapping can be only used from the manipulator's axis handle.

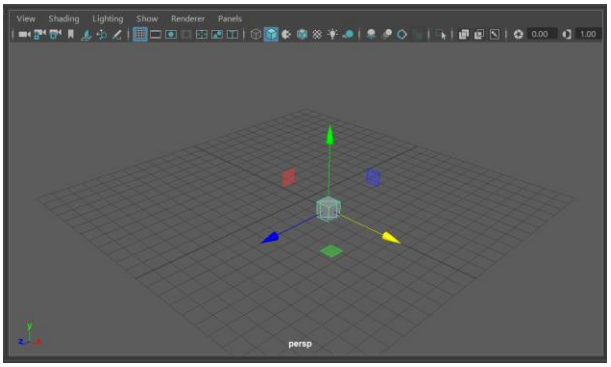
The Scale Tool

Scales selected object or components by dragging the scale manipulator.

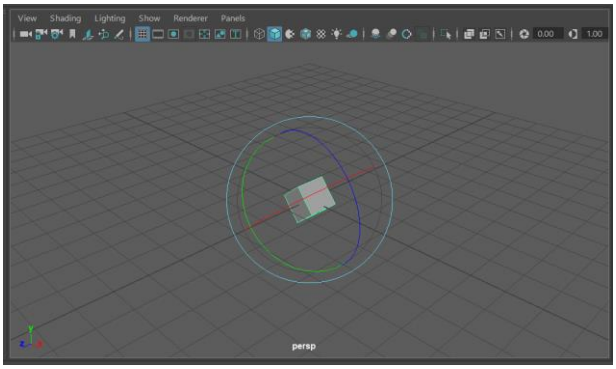
Utilize the scale tool to alter the dimension of the items by scaling uniformly in each of the three measurements. You can likewise scale unevenly in one dimension at any given moment. Snap the scale tool symbol in the tool compartment and choose the item you need to scale. Maya shows a scale controller comprising of four handles.

3D TRANSFORMATIONS:

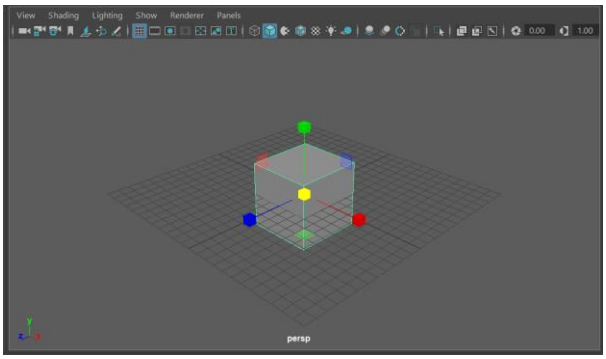
Translation



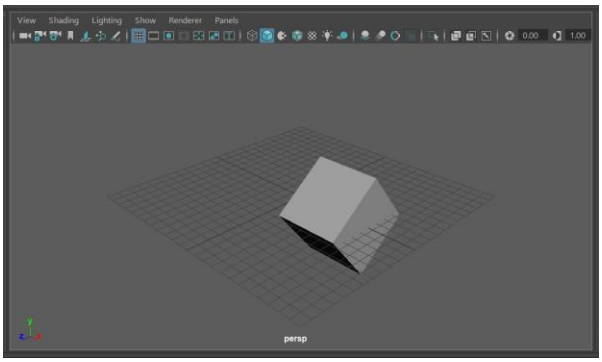
Rotation



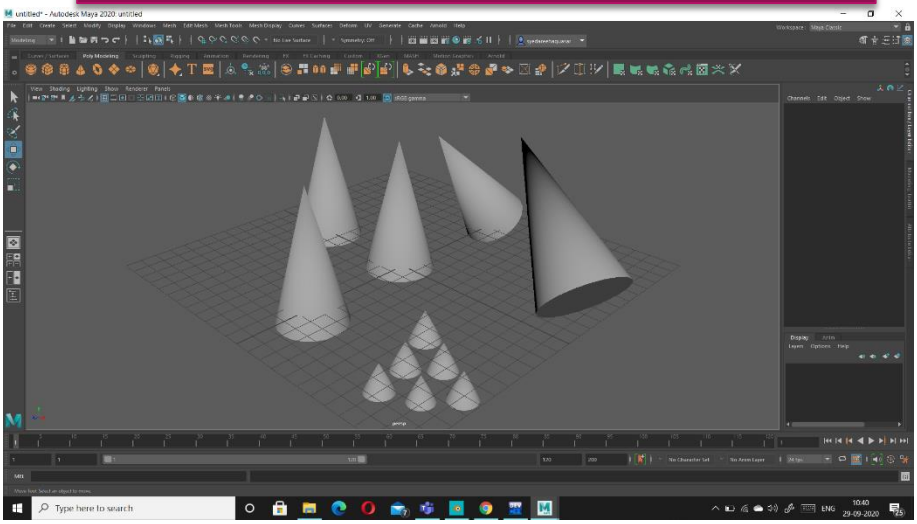
Scalling



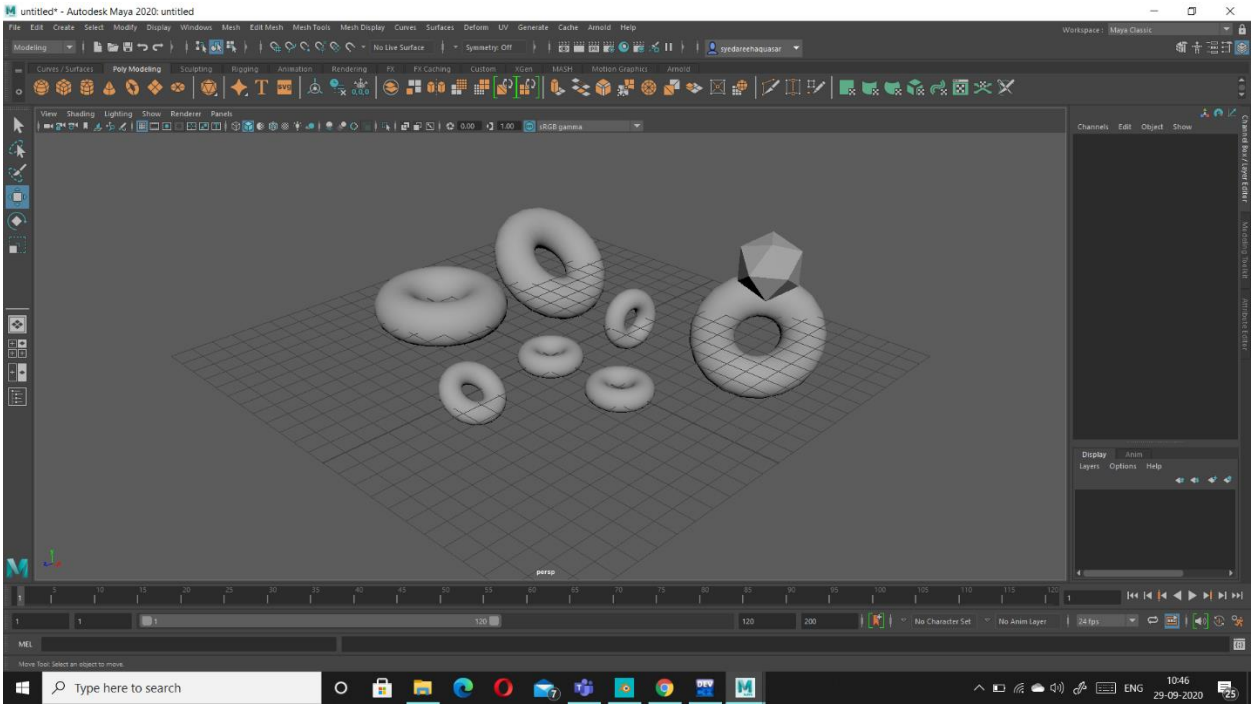
ALL 3 TRANSFORMATIONS



Transformations on Cone



Transformations on Polygon



Transformations on different 3D objects