Experiment No:07

Experiment Name: Implementation of 2D Transformation in Computer Graphics.

(Shearing)

Description:

Shearing is a type of 2D geometric transformation that distorts the shape of an object such that the transformation shifts one coordinate (x or y) proportionally to the other. It causes the shape to "slant" either horizontally, vertically, or both, without changing the dimensions of the object.

In 2D graphics, shearing is used to produce effects like italicizing text, slanting images, or simulating perspective. This transformation is represented using a shear matrix, which, when multiplied with a point or shape, changes its geometry accordingly.

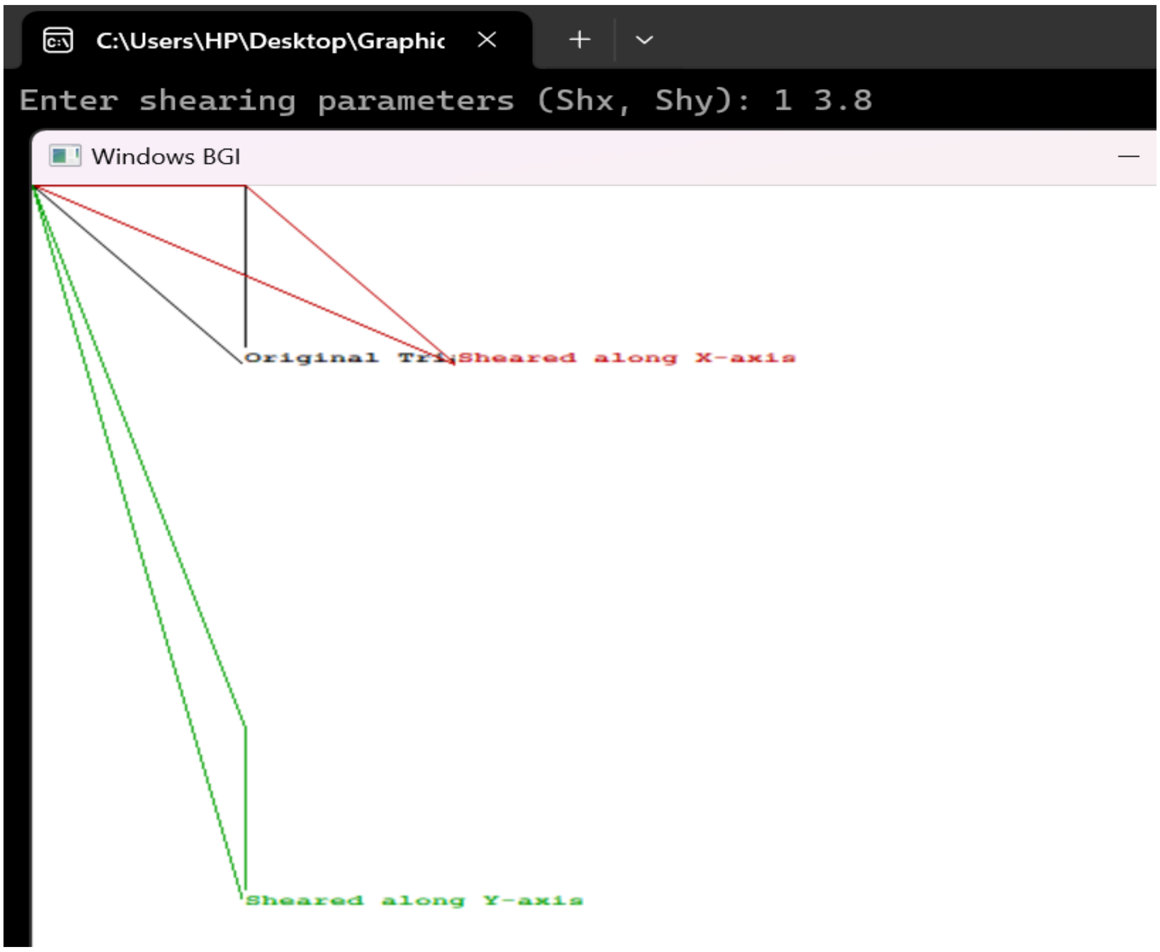
Algorithm:

1. **Start**
2. Define the original coordinates of the object (e.g., square or triangle).
3. Input the shear factor(s): shx for x-shear or shy for y-shear.
4. Apply the appropriate shearing transformation:
    - For X-shear: x' = x + shx * y, y' = y
    - For Y-shear: x' = x, y' = y + shy * x
5. Store the new transformed coordinates.
6. Draw the original shape.
7. Draw the sheared shape using the new coordinates.
8. **End**

Source Code:

```
#include <graphics.h>

#include <iostream>

using namespace std;

void shearX(int &x, int &y, int Shx) {

   // Shearing along X-axis: X' = X + Shx * Y

   int newX = x + Shx * y;

   int newY = y;

   x = newX;

   y = newY;

}

void shearY(int &x, int &y, int Shy) {
   // Shearing along Y-axis: Y' = Y + Shy * X
   int newX = x;
   int newY = y + Shy * x;
   x = newX;
   y = newY;
}int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "");
       setbkcolor(WHITE);
   cleardevice();
// Original coordinates of the triangle
   int x1 = 1, y1 = 1;
   int x2 = 0, y2 = 0;
   int x3 = 1, y3 = 0;
int Shx, Shy;
   cout << "Enter shearing parameters (Shx, Shy): ";
   cin >> Shx >> Shy;
// Original triangle (drawn first)
   setcolor(BLACK);
   line(x1 * 100, y1 * 100, x2 * 100, y2 * 100);
   line(x2 * 100, y2 * 100, x3 * 100, y3 * 100);
   line(x3 * 100, y3 * 100, x1 * 100, y1 * 100);
   outtextxy(x1 * 100, y1 * 100 - 10, "Original Triangle");
// Shearing in X-axis
   int x1_x = x1, y1_x = y1;
   int x2_x = x2, y2_x = y2;
   int x3_x = x3, y3_x = y3;
   shearX(x1_x, y1_x, Shx);
   shearX(x2_x, y2_x, Shx);

   shearX(x3_x, y3_x, Shx);

// Draw sheared triangle after shearing in X-axis

   setcolor(RED);

   line(x1_x * 100, y1_x * 100, x2_x * 100, y2_x * 100);

   line(x2_x * 100, y2_x * 100, x3_x * 100, y3_x * 100);

   line(x3_x * 100, y3_x * 100, x1_x * 100, y1_x * 100);

   outtextxy(x1_x * 100, y1_x * 100 - 10, "Sheared along X-axis");

// Shearing in Y-axis

   int x1_y = x1, y1_y = y1;

   int x2_y = x2, y2_y = y2;

   int x3_y = x3, y3_y = y3;

shearY(x1_y, y1_y, Shy);

   shearY(x2_y, y2_y, Shy);

   shearY(x3_y, y3_y, Shy);

// Draw sheared triangle after shearing in Y-axis

   setcolor(GREEN);

   line(x1_y * 100, y1_y * 100, x2_y * 100, y2_y * 100);

   line(x2_y * 100, y2_y * 100, x3_y * 100, y3_y * 100);

   line(x3_y * 100, y3_y * 100, x1_y * 100, y1_y * 100);

   outtextxy(x1_y * 100, y1_y * 100 - 10, "Sheared along Y-axis");

getch();

   closegraph();

   return 0;

}
```

Output:



```
Enter shearing parameters (Shx, Shy): 1 3.8
```

Figure name: Shearing of triangle.

Conclusion:

In this experiment, we successfully implemented 2D shearing transformation in computer graphics. By applying horizontal and vertical shearing matrices to a 2D object, we observed how its shape could be distorted without changing its size or orientation. This helped in understanding the concept of **non-uniform transformation**, where only one axis is affected. Such transformations are important in computer graphics for creating visual effects and perspective projections.