**Practical No. 8**

|  |
| --- |
| ***Title:-*** Write a C program for 2D Translation and Scaling. |

***Course outcome*** :- Perform and demonstrate basic and composite graphical transformations on given object

***Resources Required (Hardware & Softwares):-***

1. A Desktop PC/ Laptop
2. Ansi C/ Turbo C/ (Any distribution) installed

***Theory:-***

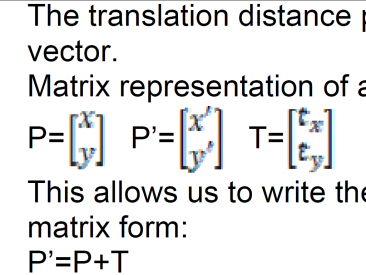
***Translation***

Translation is a process of changing the position of an object in a straight-line path from one coordinate location to another. We can translate a two dimensional point by adding translation distance tx and ty to the original coordinate position(x,y) to move the point to a new position(x’,y’) as shown

x’=x+tx

y’=y+ty

The translation distance pair(tx,ty) is called a translation vector or shift vector. Matrix representation of above equations:



This allows us to write the two dimensional translation equations in the matrix form:

**P’=P+T**

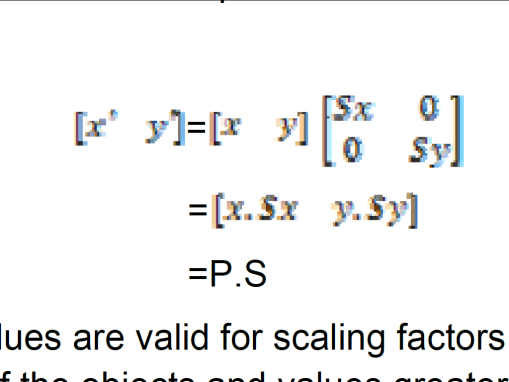
***Scaling***

A scaling transformation changes the size of an object. This operation can be carried out for polygons by multiplying the coordinate values(x,y) of each vertex by scaling factors Sx and Sy to produce the transformed coordinates(x’,y’)

x’=x.Sx

y’=y.Sy

Scaling factor Sx scales object in the x direction and scaling factor Sy scales object in the y direction. The above equations can be written in the matrix form as given below:



Any positive numeric values are valid for scaling factors Sx and Sy. Values less than 1 reduce the size of the objects and values greater than 1 produce an enlarged object. For both Sx and Sy values equal to 1, the size of object does not change. To get uniform scaling it is necessary to assign same value for Sx and Sy. Unequal values for Sx and Sy result in a differential Scaling.

***Algorithm:-***

1. **Translation:-**

1. Read the coordinates of the polygon to be translated about origin.

2. Read the translation factor tx and ty.

3. Draw the original polygon.

4. Sum the original coordinates with Translation Matrix to obtain transformed coordinates.

5. Draw the polygon using transformed coordinates.

6. Stop.

1. **Scaling:-**

1. Read the coordinates of the polygon to be scaled about origin.

2. Read the scaling factor Sx and Sy.

3. Draw the original polygon.

4. Multiply the original coordinates with Scaling Matrix to obtain transformed coordinates.

5. Draw the polygon using transformed coordinates.

6. Stop.

**Program for Translation and Shearing**

#include <graphics.h>

#include <conio.h>

void translate(int x, int y, int tx, int ty) {

x += tx;

y += ty;

line(x, y, x + 50, y + 50); // Draw translated object (e.g., line)

}

void scale(int x, int y, float sx, float sy) {

x \*= sx;

y \*= sy;

rectangle(x, y, x + 100 \* sx, y + 50 \* sy); // Draw scaled object

}

int main() {

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:\\Turboc3\\BGI");

int x = 100, y = 100;

line(x, y, x + 50, y + 50); // Original line

translate(x, y, 50, 50); // Translate by (50, 50)

scale(100, 100, 1.5, 2); // Scale by 1.5x, 2x

getch();

closegraph();

return 0;

}

**Output:- ( Paste your own Output )**

***Conclusion:-***

Thus,we have written & implemented Program for 2 D transformations **Translation and Scaleing.**