ST. ANNE'S

COLLEGE OF ENGINEERING AND TECHNOLOGY

ANGUCHETTYPALAYAM, PANRUTI – 607 110

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LAB MANUAL

CS6513-COMPUTER GRAPHICS LABORATORY

Regulation 2013

Year / Semester: III / V

Jun 2017 – Dec 2017

PREPARED BY

Ms. V.VARALAKSHMI, M.E.,
Assistant Professor /CSE

LIST OF EXPERIMENTS

IMPLEMENT THE EXERCISES USING C / OPENGL / JAVA

1. Implementation of Algorithms for drawing 2D Primitives – Line (DDA, Bresenham) – all slopes Circle (Midpoint)
2. 2D Geometric transformations – Translation, Rotation Scaling, Reflection Shear, Window-Viewport
3. Composite 2D Transformations
4. Line Clipping
5. 3D Transformations - Translation, Rotation, Scaling.
6. 3D Projections – Parallel, Perspective.
7. Creating 3D Scenes.
8. Image Editing and Manipulation - Basic Operations on image using any image editing software, creating gif animated images, Image optimization.
9. 2D Animation – To create Interactive animation using any authoring tool.

REFERENCE:

spoken-tutorial.org

SOFTWARE :C, C++, Java, OpenGL

TABLE OF CONTENTS

EX.NO: 1 IMPLEMENTATION OF ALGORITHMS FOR DRAWING 2D PRIMITIVES

DATE:

a) DDA ALGORITHM FOR DRAWING LINE

AIM:
To write a C program for draw a line using DDA line Algorithm.

ALGORITHM:

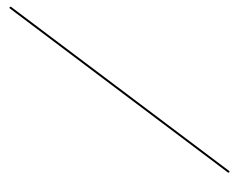
1. Start the program.
2. Enter the starting and ending point of the line.
3. Call the initgraph() function.
4. Invoke the function draw, and calculate the absolute value of dx and dy and check if abs(dx)>abs(dy).
5. If true assign step size as abs(dx), or else assign as abs(dy).
6. Calculate the x in c, y in c and plot the start point use the loop, until k is less than or equal to step size.
7. Calculate the x and y for each steps and plot the corresponding pixels and let the line be displayed.
8. Stop the graphics driver.
9. Stop the program.

PROGRAM:

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

void draw(int x1,int y1,int x2,int y2);
int main(void)
{
        int x1,y1,x2,y2;
        int gdriver=DETECT,gmode,gerror;
        printf("\n Enter the x and y value for starting point:\n");
        scanf("%d%d",&x1,&y1);
        printf("\n Enter the x and y value for ending point:\n");
        scanf("%d%d",&x2,&y2);
        clrscr();
        initgraph(&gdriver,&gmode,"E:\\TC\\BGI\\");
        draw(x1,y1,x2,y2);
        getch();
        return 0;
}

void draw(int x1,int y1,int x2,int y2)
{
        float x,y,xinc,yinc,dx,dy;
        int k,step;
        dx=x2-x1;
        dy=y2-y1;
        if(abs(dx)>abs(dy))
                step=abs(dx);
        else
                step=abs(dy);

        xinc=dx/step;
        yinc=dy/step;
        x=x1;
        y=y1;
        putpixel(abs(x),abs(y),111);
        for(k=1;k<=step;k++)
        {
                x=x+xinc;
                y=y+yinc;
                putpixel(abs(x),abs(y),111);
        }
}
```

OUTPUT:

```
Enter the x and y value for starting point: 50 60
Enter the x and y value for ending point: 70 90
```



RESULT:

Thus the program in C to draw the line by using DDA algorithm was written and executed successfully.

b) BRESENHAM'S ALGORITHM FOR LINE DRAWING

AIM:

To write a program in C to draw a line using Bresenham's algorithm.

ALGORITHM:

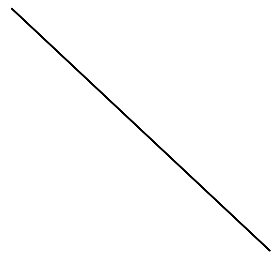
1. Start the program.
2. Initialize the variables.
3. Call the initgraph() function.
4. Input the two line end-points and store the left end-point in (x_0, y_0).
5. Plot the point (x_0, y_0).
6. Calculate the constants Δx, Δy, $2\Delta y$, and ($2\Delta y$ - $2\Delta x$) and get the first value for the decision parameter as:
7. At each x_k along the line, starting at $k = 0$, perform the following test. If $p_k < 0$, the next point to plot is (x_k+1, y_k) and
8. Otherwise, the next point to plot is (x_k+1, y_k+1) and Repeat step 4 (Δx – 1) times.
   $xyp\Delta-\Delta=20yppkk\Delta+=+21xyppkk\Delta-\Delta+=+221$
9. Stop the graphics driver.
10. Stop the program.

PROGRAM:

```
# include <stdio.h>
# include <conio.h>
# include <graphics.h>
void main( )
{
        int x,y,x1,y1,x2,y2,p,dx,dy;
        int gdriver=DETECT,gmode;
        initgraph(&gdriver,&gmode,"C:\\tc\\BGI:");
        printf("\nEnter the x-coordinate of the first point ::");
        scanf("%d",&x1);
        printf("\nEnter the y-coordinate of the first point ::");
        scanf("%d",&y1);
        printf("\nEnter the x-coordinate of the second point ::");
        scanf("%d",&x2);
        printf("\nEnter the y-coordinate of the second point ::");
        scanf("%d",&y2);
        x=x1;
        y=y1;
        dx=x2-x1;
        dy=y2-y1;
        putpixel(x,y,2);
        p=(2dy-dx);
        while(x<=x2)
        {
                if(p<0)
                {
                        x=x+1;
                        p=2*x-dx;
                }
                else
                {
                        x=x+1;
                        y=y+1;
                        p=p+2*dy;
                }
                putpixel(x,y,7);
        }
        getch();
        closegraph();
}
```

OUTPUT:

```
Enter the x-coordinate of the first point ::40
Enter the y-coordinate of the first point ::50
Enter the x-coordinate of the second point ::60
Enter the y-coordinate of the second point ::80
```



RESULT:

Thus the program in C to draw the line by using Bresenham’s algorithm was done successfully.

c) BRESENHAM'S ALGORITHM TO DRAW A CIRCLE

AIM:

To write a program in C to draw a circle using Bresenham's Algorithm.

ALGORITHM:

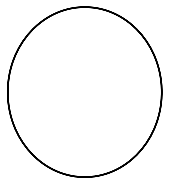
1. Start the program.
2. Initialize the variables.
3. Call the initgraph() function.
4. Input radius *r* and circle centre (x_c, y_c), then set the coordinates for the first point on the circumference of a circle centred on the origin as:
5. Calculate the initial value of the decision parameter as:
6. Starting with $k = 0$ at each position x_k, perform the following test. If $p_k < 0$, the next point along the circle centred on $(0, 0)$ is (x_k+1, y_k) and *rp*–=450),
7. Otherwise the next point along the circle is (x_k+1, y_k-1) and
8. Determine symmetry points in the other seven octants
9. Move each calculated pixel position (x, y) onto the circular path centred at (x_c, y_c) to plot the coordinate values:
10. Repeat steps 3 to 5 until $x >= y$
11. Stop the graphics driver.
12. Stop the program.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void circlepoints(int,int);
void main( )
{
        int x,y,p,r;
        int gdriver=DETECT,gmode;
        initgraph(&gdriver,&gmode,"C:\\tc\\bgi:");
        clrscr();
        printf("Enter the radius");
        scanf("%d",&r);
        x=0;y=r;p=1-r;
        while(x<y)
        {
                x++;
                if(p>0)
                {
                        p=p+2*(x-y)+1;
                        y--;
                }
                else
                        p=p+2*x+1;
                        circlepoints(x,y);
        }
        getch();
        closegraph();
}

void circlepoints(int x,int y)
{
        putpixel(x+300,y+300,8);
        putpixel(x+300,-y+300,8);
        putpixel(-x+300,y+300,8);
        putpixel(-x+300,-y+300,8);
        putpixel(y+300,x+300,8);
        putpixel(y+300,-x+300,8);
        putpixel(-y+300,x+300,8);
        putpixel(-y+300,-x+300,8);
}
```

OUTPUT:

```
Enter the radius 50
```



RESULT:

Thus the program in C to draw the circle by using Bresenham's algorithm was done successfully.

VIVA QUESTIONS AND ANSWERS

1. **Define Computer Graphics.**
   Computer graphics remains one of the most existing and rapidly growing computer fields. Computer graphics may be defined as a pictorial representation or graphical representation of objects in a computer.

2. **Define pixel.**
   Pixel is shortened forms of picture element. Each screen point is referred to as pixel or pel.

3. **What is an output primitive?**
   Graphics programming packages provide function to describe a scene in terms of these basic geometric structures, referred to as output primitives.

4. **Write short notes on lines.**
   A line is of infinite extent can be defined by an angle of slope . and one point on the line P=P(x,y). This can also be defined as Y=mx+C where C is the Y- intercept.

5. **Define Circle.**
   Circle is defined by its center xc, yc and its radius in user coordinate units. The equation of the circle is (x-xc) +(y-yc) = r2.

6. **Define Ellipse.**
   An ellipse can use the same parameters xc, yc ,r as a circle, in addition to the eccentricity e. the eqn of an ellipse is: (x-xc)2/a2 + (y-yc)2/b2 = 1

7. **What are the various attributes of a line?**
   The line type, width and color are the attributes of the line. The line type include solid line, dashed lines, and dotted lines.

8. **What is pixel phasing?**
   Pixel phasing is an antialiasing technique, stair steps are smoothed out by moving the electron beam to more nearly approximate positions specified by the object geometry.

9. **What is a Line cap?**
   Line caps can be used to adjust the shape of the line ends to give a better appearance. There are three types of line caps.

10. **What is an attribute parameter?**
    Any parameter that affects the way a primitive is to be displayed is referred to as an attribute parameter.

APPLICATIONS

- Paint Programs: It used to draw common shapes such as lines, circle and ovals
- Computer games: Vector Graphics may use some programs such as Adobe Illustrator for making computer games
- Interactive graphical user interfaces (Mouse Pointer).
- Graphic Design.

EX.NO: 2 **IMPLEMENTATION OF 2D GEOMETRIC TRANSFORMATIONS**

DATE:

AIM:

To implement the following 2D Geometric Transformations

a. Translation b. Rotation c. Scaling d. Reflection e. Shearing
f. Window-View Port

ALGORITHM:

1. Get the coordinates of triangle (x1, y1, x2, y2, x3, y3).
2. Draw the original triangle.
3. Print the menu for choosing 2D Geometric Transformation.

a. If users choose translation then get the translation factors(x, y) and draw the translated triangle in the following coordinates (x1+x, y1+y, x2+x, y2+y, x3+x, y3+y).

b. (i) If user choose rotation then get the rotation angle (t) and reference point of the rotation (rx, ry).

(ii) Change the t value to t = t * (3.14 / 180) and calculate the rotated coordinates by the following formulae rx1 = rx + (x1 - rx) * cos (t) - (y1 - ry) * sin (t); ry1 = ry + (x1 - rx) * sin (t) + (y1 - ry) *cos (t);

(iii) Similarly calculate the coordinates rx2, ry2, rx3, ry3 and draw the rotated triangle in the following coordinates (rx1, ry1, rx2, ry2, rx3, ry3).

c. If user choose scaling then get the scaling factors(x, y) and draw the scaleded triangle in the following coordinates (x1*x, y1*y, x2*x, y2*y, x3*x, y3*y).

d. If user choose reflection then rotate the triangle in 1800 at (x2, y2) and draw the rotated triangle (which is reflected triangle).

e. If user choose shearing then get the shear value and draw the sheared triangle in the following coordinates (x1, y1, x2+x, y2, x3, y3).

f. (i) if user choose window-view port then draw the rectangle in window port coordinates (w1, w2, w3, w4) and draw the original triangle.

(ii) Calculate the x, y and view port coordinates by following formulae

```
x = (v3 - v1) / (w3 - w1);
y = (v4 - v2) / (w4 - w2);
vx1 = v1 + floor (((x1 - w1) * x) + 0.5);
vy1 = v2 + floor (((y1 - w2) * y) + 0.5);
```

(iii) Similarly calculate the coordinates vx2, vy2, vx3, vy3

(iv) Draw the rectangle in view port coordinates (v1, v2, v3, v4) and draw the triangle in view Port by the following coordinates (vx1, vy1, vx2, vy2, vx3, vy3)

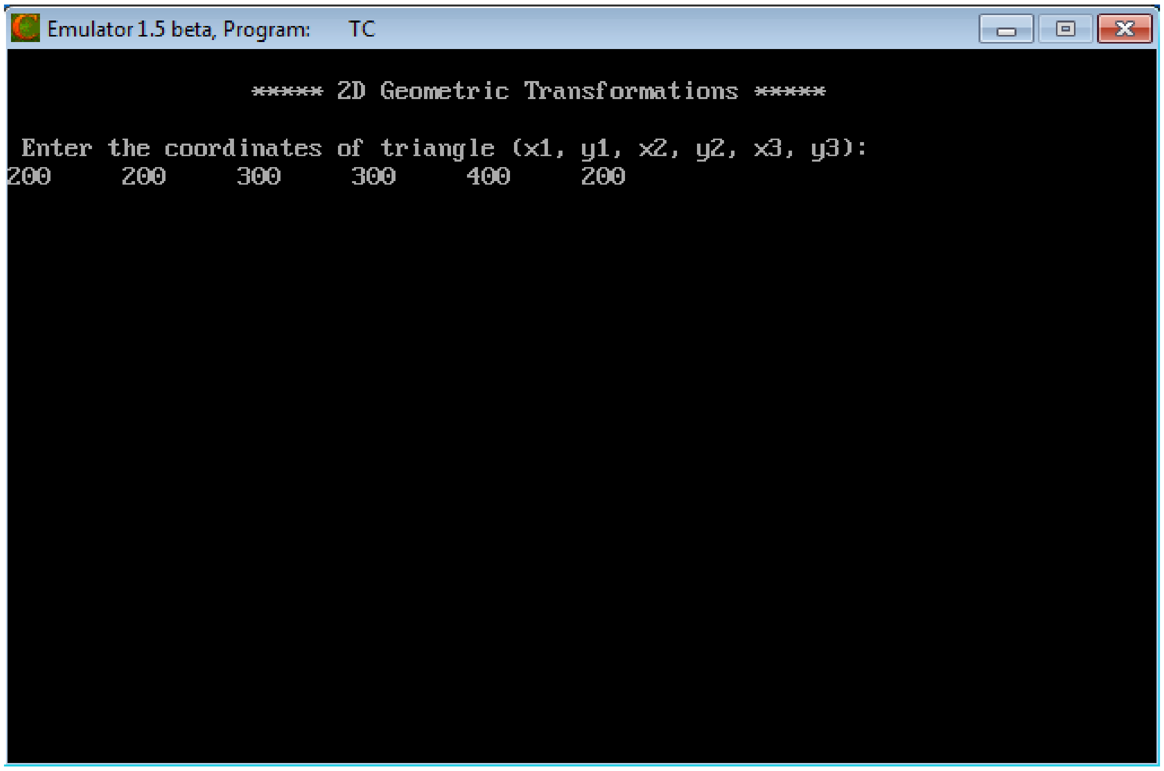
PROGRAM:

```
#include <graphics.h>
#include <iostream.h>
#include <conio.h>
#include <math.h>
float x1, y1, x2, y2, x3, y3, x, y, rx1, ry1, rx2, ry2, rx3, ry3, t, vx1, vy1, vx2, vy2, vx3, vy3;
floatw1 = 5, w2 = 5, w3 = 635, w4 = 465, v1 = 425, v2 = 75, v3 = 550, v4 = 250;
intgd, gm, ch;
void original ();
void triangle (float, float, float, float, float, float);
void rotate (float, float, float);
void main ()
{
        clrscr ();
        cout<< "\n\t\t ***** 2D Geometric Transformations *****";
        cout<< "\n\n Enter the coordinates of triangle (x1,y1,x2,y2,x3, y3): \n";
        cin>> x1 >> y1 >> x2 >> y2 >> x3 >> y3;
        original ();
        closegraph ();
        do
        {
                cout<< "\n\n Choose any one Transformation : ";
                cout<< "\n\t 1.Translation \n\t 2.Rotation \n\t 3.Scaling \n\t 4.Reflection";
                cout<< "\n\t 5.Shearing \n\t 6.Window-Viewport";
                cout<< "\n\n Enter your choice : \t";
                cin>>ch;
                switch (ch)
                {
                        case 1:
                                cout<< "\n Enter the translation factors (x,y): \t\t";
                                cin>> x >> y;
                                original ();
                                triangle (x1+x, y1+y, x2+x, y2+y, x3+x, y3+y);
                                closegraph ();
                                break;
                        case 2:
                                cout<< "\n Enter the angle of rotation : \t\t";
                                cin>> t;
                                cout<< "\n Enter the reference point of rotation (rx,ry) : \t";
                                cin>> x >> y;
                                original ();
                                rotate (t, x, y);
                                closegraph ();
                                break;
                        case 3:
```

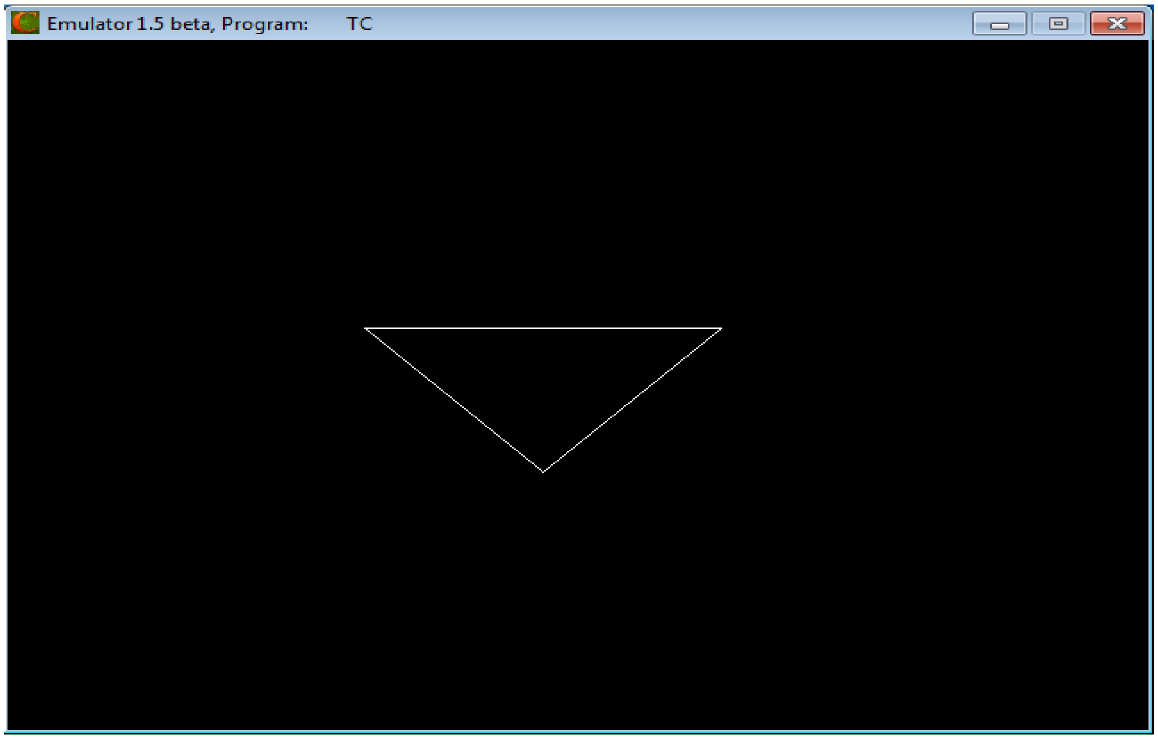
```
                        cout<< "\n Enter the scaling factors (x,y): \t\t";
                        cin>> x >> y;
                        original ();
                        triangle (x1*x, y1*y, x2*x, y2*y, x3*x, y3*y);
                        closegraph ();
                        break;
                case 4:
                        original ();
                        rotate (180, x2, y2);
                        closegraph ();
                        break;
                case 5:
                        cout<< "\n Enter the Shear Value : \t\t";
                        cin>> x;
                        original ();
                        triangle (x1, y1, x2+x, y2, x3, y3);
                        closegraph ();
                        break;
                case 6:
                        initgraph (&gd, &gm, "C:\\TC\\BGI");
                        rectangle (w1, w2, w3, w4);
                        outtextxy (300, 10, "Window Port");
                        triangle (x1, y1, x2, y2, x3, y3);
                        x = (v3 - v1) / (w3 - w1);
                        y = (v4 - v2) / (w4 - w2);
                        vx1 = v1 + floor (((x1 - w1) * x) + 0.5);
                        vy1 = v2 + floor (((y1 - w2) * y) + 0.5);
                        vx2 = v1 + floor (((x2 - w1) * x) + 0.5);
                        vy2 = v2 + floor (((y2 - w2) * y) + 0.5);
                        vx3 = v1 + floor (((x3 - w1) * x) + 0.5);
                        vy3 = v2 + floor (((y3 - w2) * y) + 0.5);
                        rectangle (v1, v2, v3, v4);
                        outtextxy (450, 85, "View Port");
                        triangle (vx1, vy1, vx2, vy2, vx3, vy3);
                        closegraph ();
                }
        } while (ch<= 6);
        getch ();
}

void original ( )
{
        initgraph (&gd, &gm, "C:\\TC\\BGI");
        triangle (x1, y1, x2, y2, x3, y3);
}
```

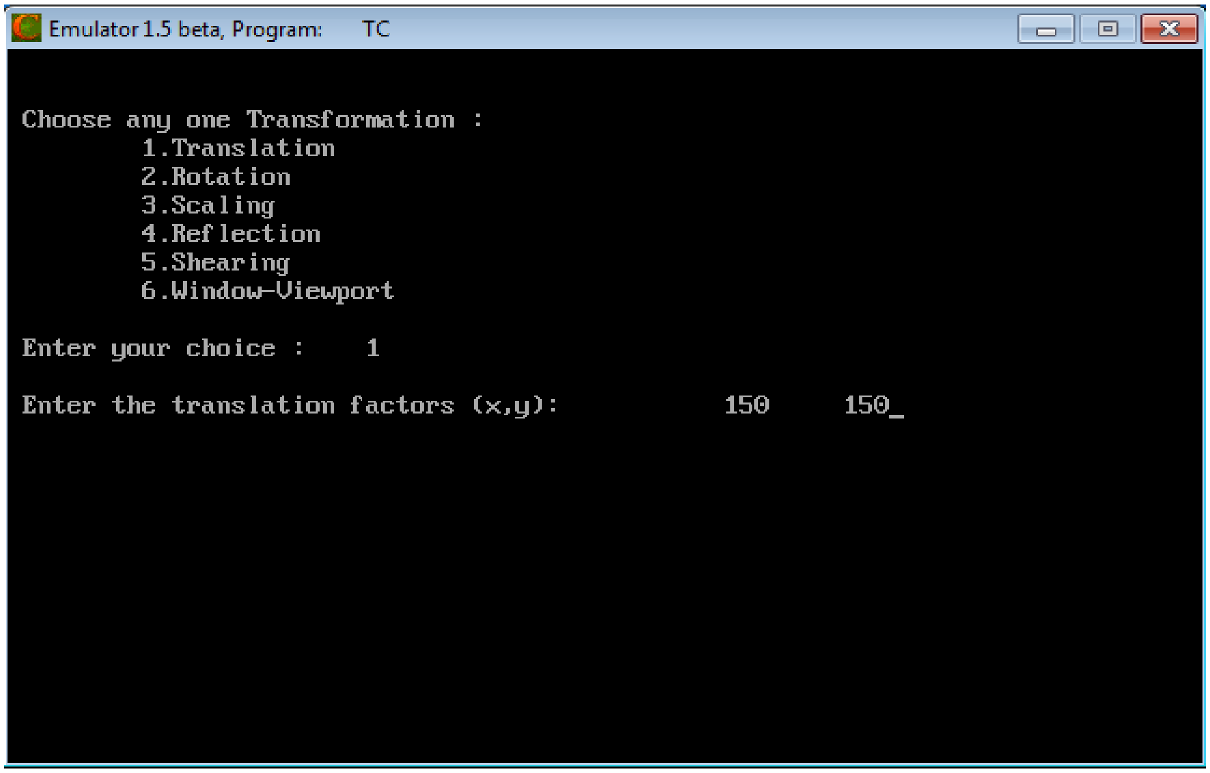
```
void triangle (float x1, float y1, float x2, float y2, float x3, float y3)
{
        line (x1, y1, x2, y2);
        line (x2, y2, x3, y3);
        line (x3, y3, x1, y1);
        getch ();
}

void rotate (float t, float rx, float ry)
{
        t = t * (3.14 / 180);
        rx1 = rx + (x1 - rx) * cos (t) - (y1 - ry) * sin (t);
        ry1 = ry + (x1 - rx) * sin (t) + (y1 - ry) * cos (t);
        rx2 = rx + (x2 - rx) * cos (t) - (y2 - ry) * sin (t);
        ry2 = ry + (x2 - rx) * sin (t) + (y2 - ry) * cos (t);
        rx3 = rx + (x3 - rx) * cos (t) - (y3 - ry) * sin (t);
        ry3 = ry + (x3 - rx) * sin (t) + (y3 - ry) * cos (t);
        triangle (rx1, ry1, rx2, ry2, rx3, ry3);
}
```

OUTPUT:

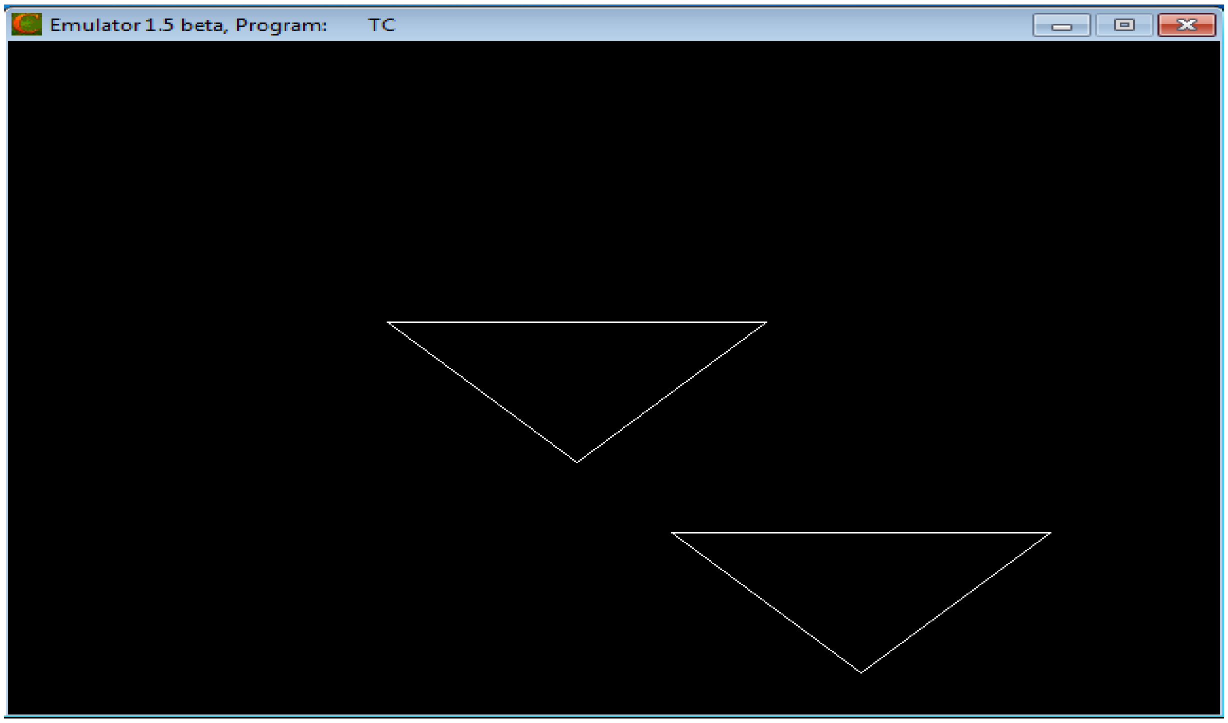


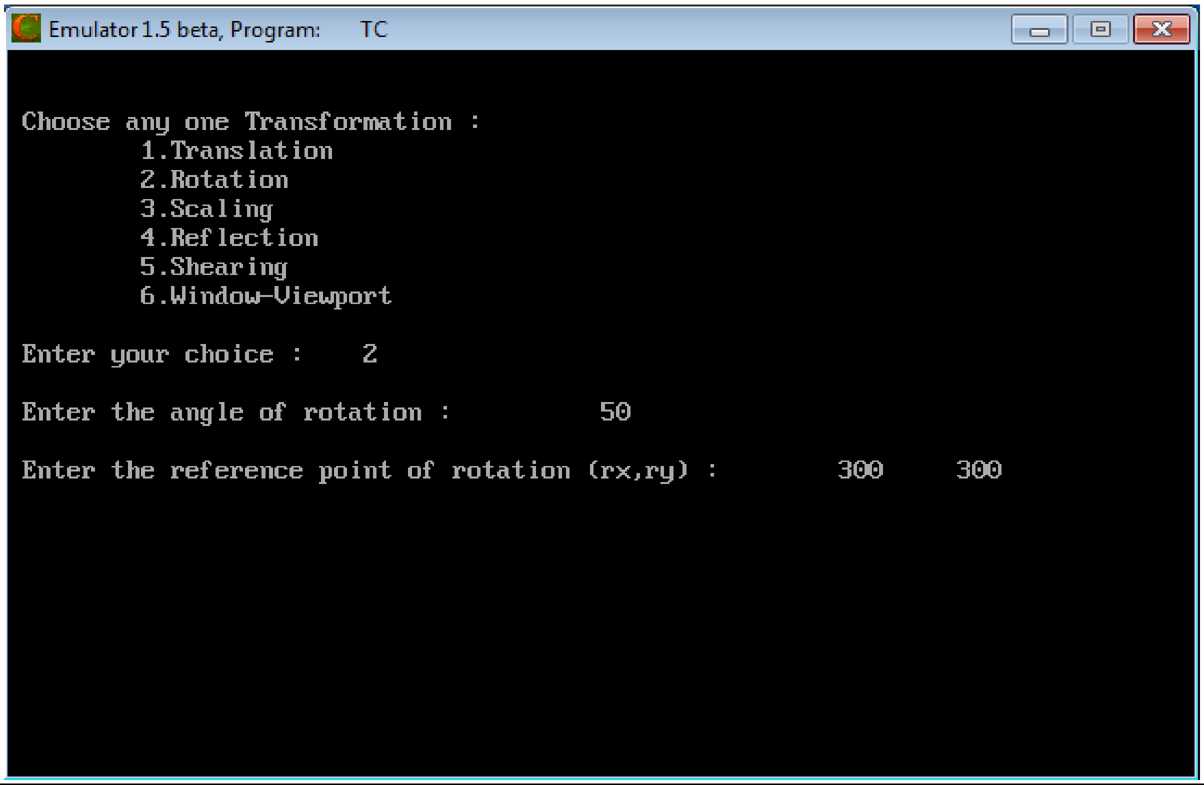
```
                ***** 2D Geometric Transformations *****

 Enter the coordinates of triangle (x1, y1, x2, y2, x3, y3):
200     200     300     300     400     200
```



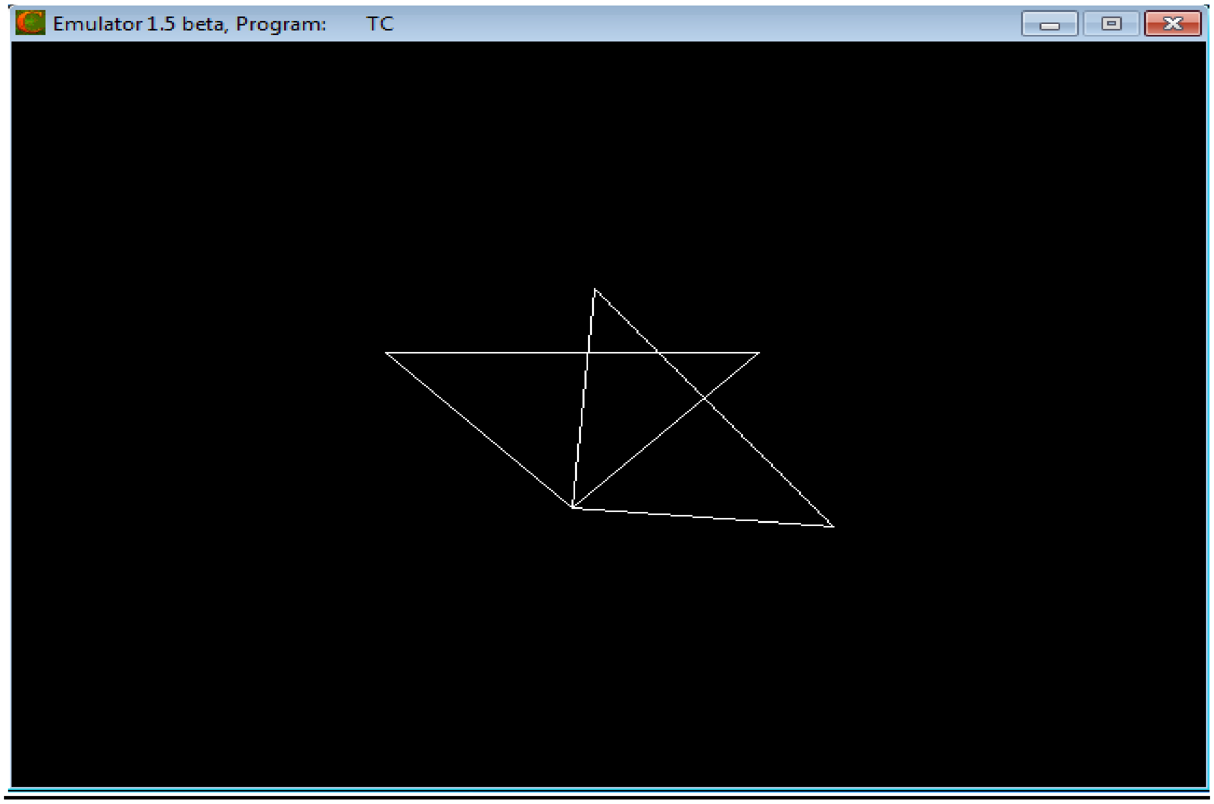


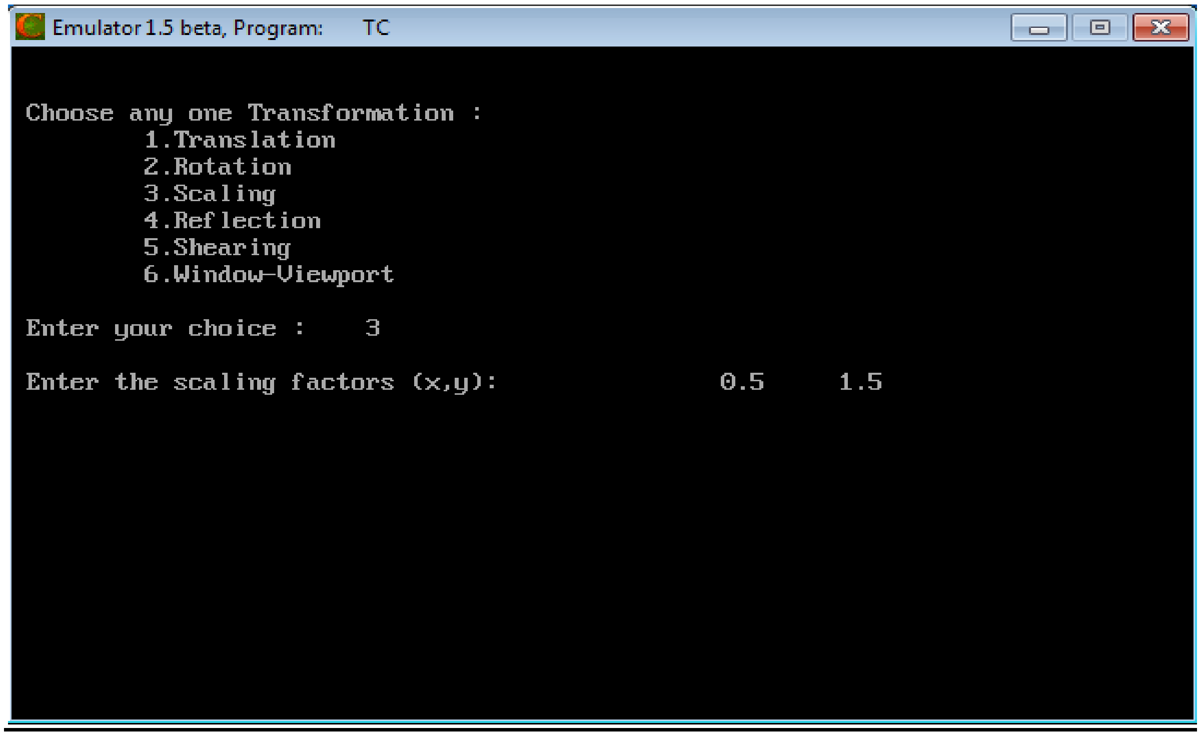
```
Choose any one Transformation :
        1.Translation
        2.Rotation
        3.Scaling
        4.Reflection
        5.Shearing
        6.Window-Viewport

Enter your choice :    1

Enter the translation factors (x,y):           150     150_
```



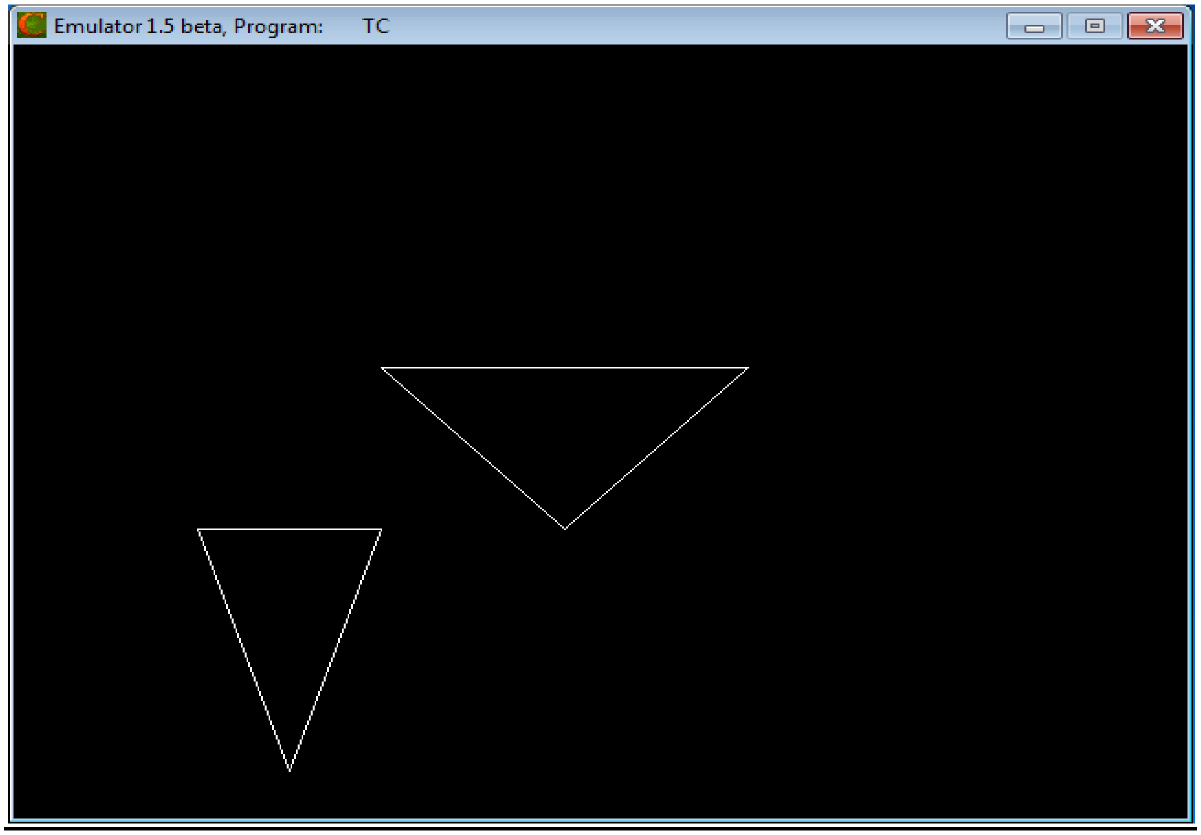


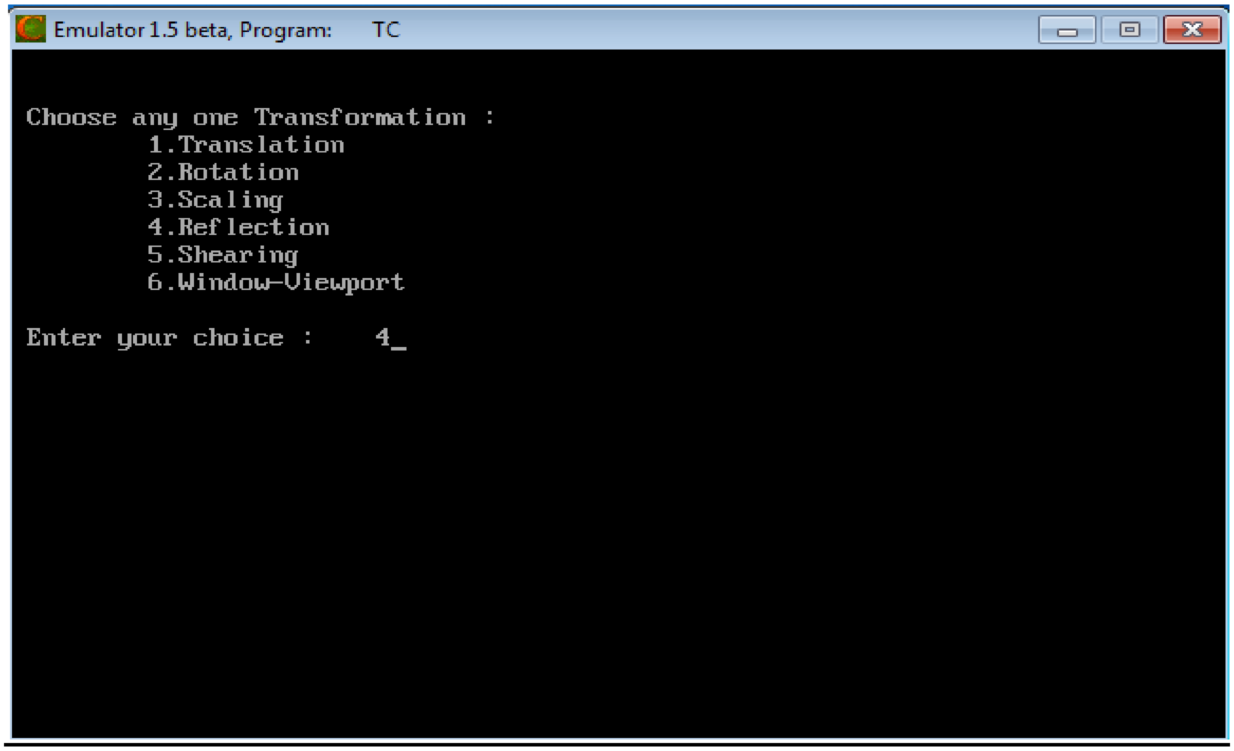
```
Choose any one Transformation :
        1.Translation
        2.Rotation
        3.Scaling
        4.Reflection
        5.Shearing
        6.Window-Viewport

Enter your choice :    2

Enter the angle of rotation :           50

Enter the reference point of rotation (rx,ry) :        300     300
```



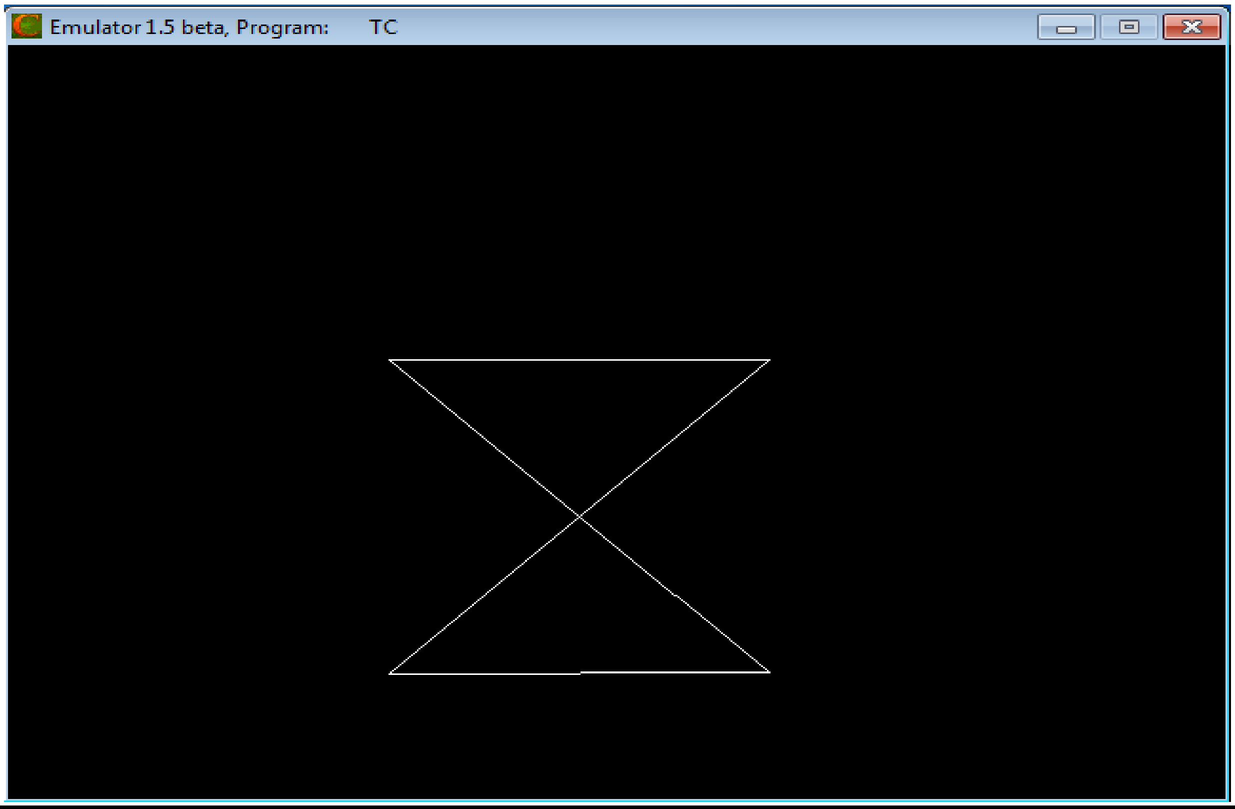


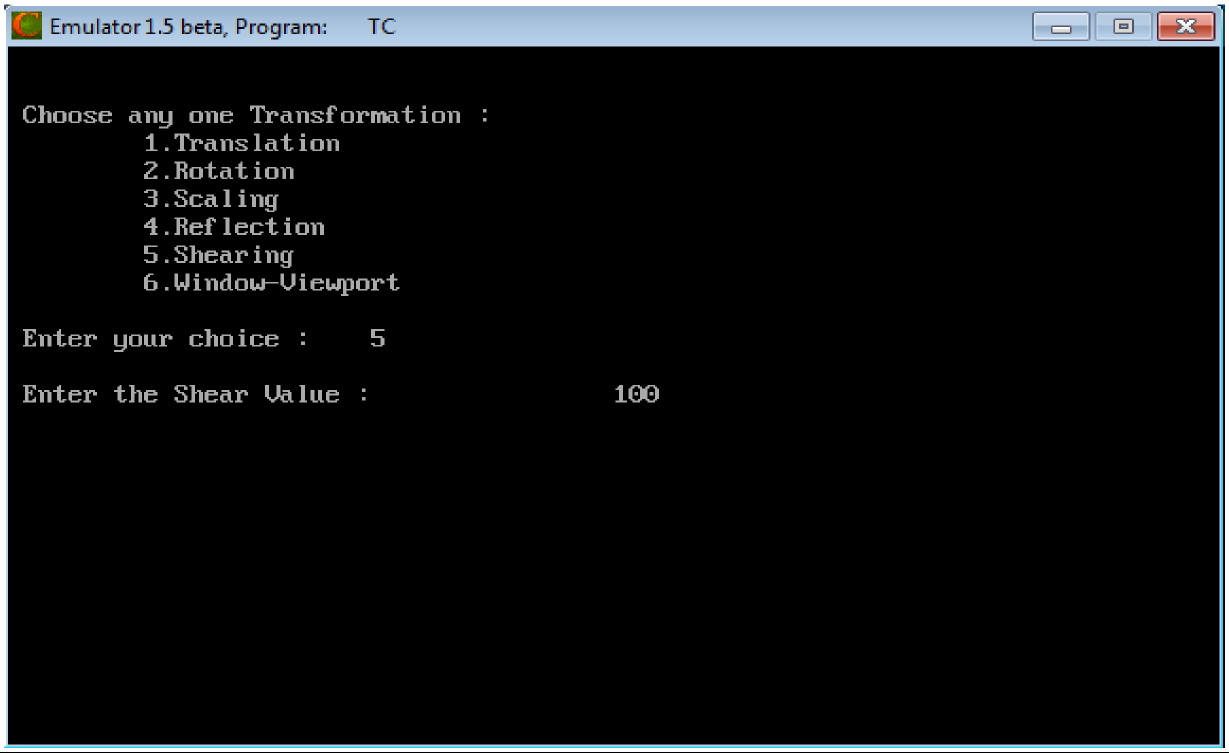
```
Choose any one Transformation :
        1.Translation
        2.Rotation
        3.Scaling
        4.Reflection
        5.Shearing
        6.Window-Viewport

Enter your choice :    3

Enter the scaling factors (x,y):              0.5     1.5
```



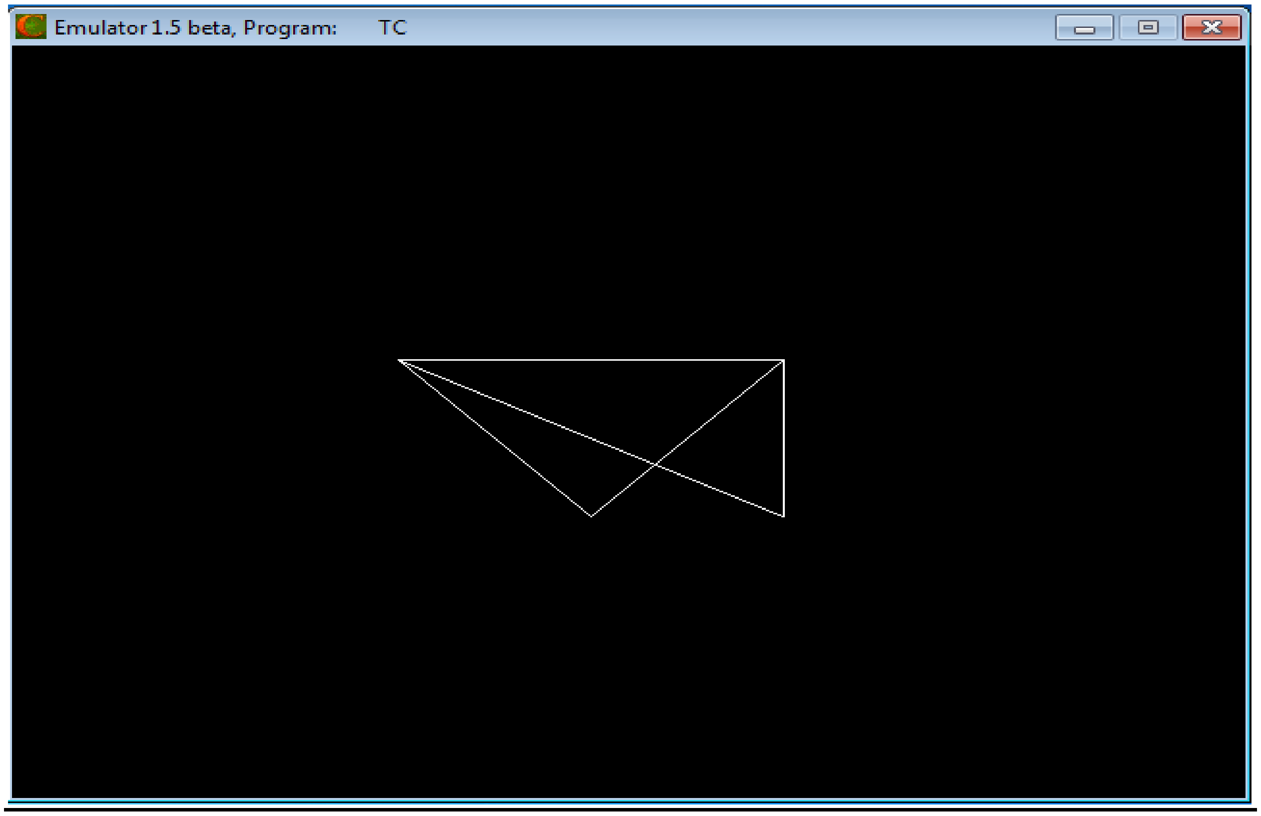


```
Choose any one Transformation :
        1.Translation
        2.Rotation
        3.Scaling
        4.Reflection
        5.Shearing
        6.Window-Viewport

Enter your choice :    4_
```

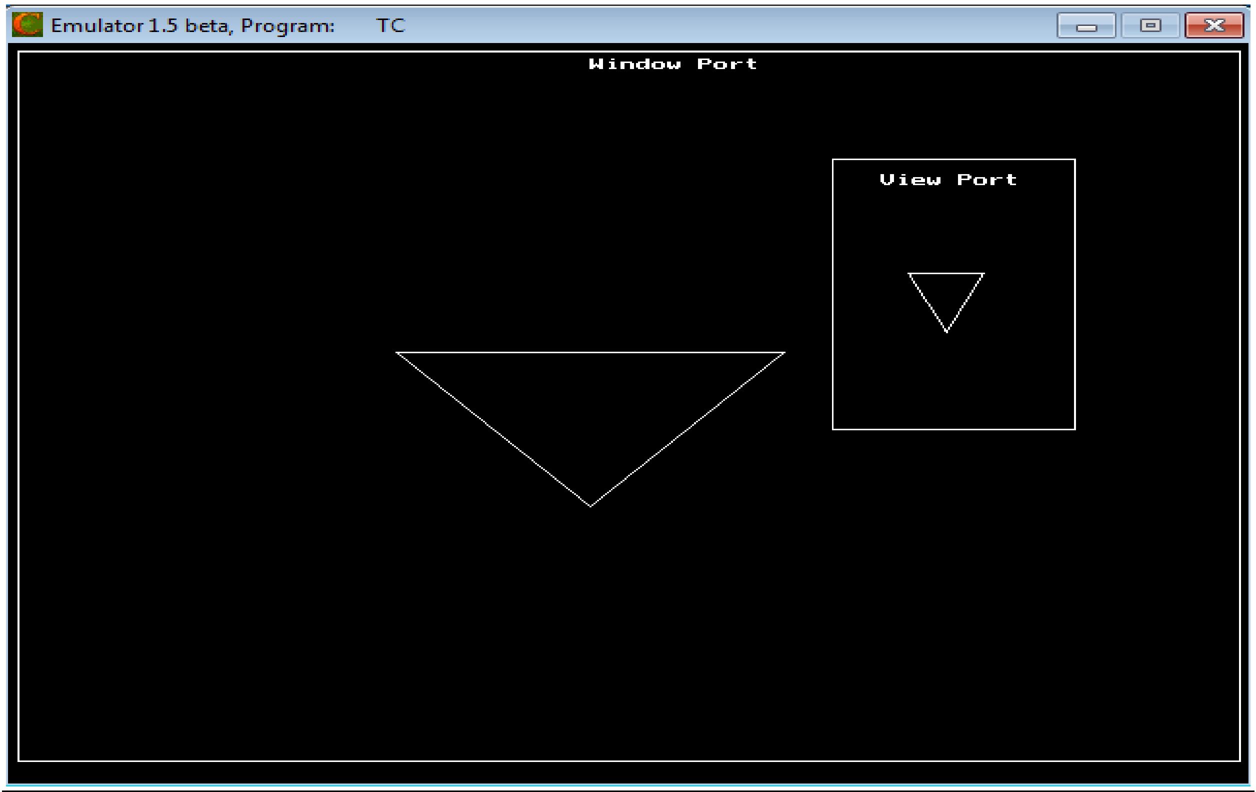




```
Choose any one Transformation :
        1.Translation
        2.Rotation
        3.Scaling
        4.Reflection
        5.Shearing
        6.Window-Viewport

Enter your choice :    5

Enter the Shear Value :                100
```



(If we choose 6 then we get)



(If we choose the number is in greater than 6 then we exit)

RESULT:

Thus the implementation of the 2D Geometric Transformations (translation, rotation, scaling, reflection, shearing, and window-view port) has been implemented and the output was verified.

VIVA QUESTIONS

1. **What is Transformation?**
   Transformation is the process of introducing changes in the shape size and orientation of the object using scaling rotation reflection shearing & translation etc.

2. **What is translation?**
   Translation is the process of changing the position of an object in a straight-line path from one coordinate location to another. Every point (x, y) in the object must undergo a displacement to (x´,y´). the transformation is:
   x´ = x + tx
   y´ = y+ty

3. **What is rotation?**
   A 2-D rotation is done by repositioning the coordinates along a circular path, in X = rcos (q + f) and Y = r sin (q + f).

4. **What is scaling?**
   The scaling transformations changes the shape of an object and can be carried out by multiplying each vertex (x,y) by scaling factor Sx,Sy where Sx is the scaling factor of x and Sy is the scaling factor of y.

5. **What is shearing?**
   The shearing transformation actually slants the object along the X direction or the Y direction as required.ie; this transformation slants the shape of an object along a required plane.

6. **What is reflection?**
   The reflection is actually the transformation that produces a mirror image of an object. For this use some angles and lines of reflection.

7. **Distinguish between window port & view port?**
   A portion of a picture that is to be displayed by a window is known as window port. The display area of the part selected or the form in which the selected part is viewed is known as view port.

8. **What is the need of homogeneous coordinates?**
   To perform more than one transformation at a time, use homogeneous coordinates or matrixes. They reduce unwanted calculations intermediate steps saves time and memory and produce a sequence of transformations

9. **What is fixed point scaling?**
   The location of a scaled object can be controlled by a position called the fixed point that is to remain unchanged after the scaling transformation.

10. **Define viewing transformation.**
    The mapping of a part of world coordinate scene to device coordinates are called viewing transformation. Two dimensional viewing transformations is simply referred to as window to viewport transformation or the windowing transformation.

APPLICATIONS

- Simulators (flight, driving)
- Mechanical CAD (Computer Aided Design)
- Architectural visualization
- Virtual reality
- Advertising

EX.NO: 3 IMPLEMENTATION OF COMPOSITE 2D TRANSFORMATIONS

DATE:

AIM:

To implement the following Composite 2D Transformations in the following order:

a. Shearing b. Rotation c. Translation d. Reflection
e. Scaling f. Window - View Port Transformation

ALGORITHM:

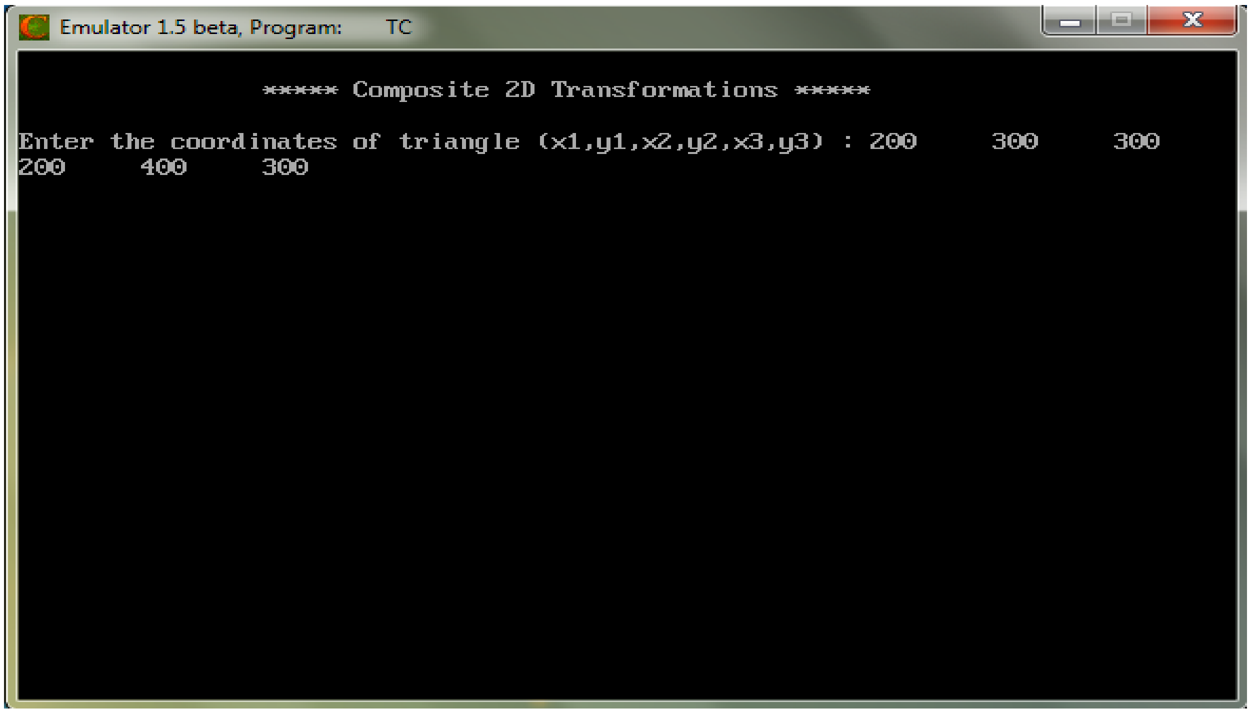
```
1. Get the coordinates of triangle (x1, y1, x2, y2, x3, y3).
2. Draw the original triangle.
3. Get the shear value and draw the sheared triangle in the following coordinates (x1, y1, x2+x,
   y2, x3, y3).
4. Get the rotation angle (t) and reference point of the rotation (rx, ry)
a) Change the t value to t = t * (3.14 / 180) and calculate the rotated coordinates by the following
   formulae
        rx1 = rx + (x1 - rx) * cos (t) - (y1 - ry) * sin (t);
        ry1 = ry + (x1 - rx) * sin (t) + (y1 - ry) * cos (t);
b) Similarly calculate the coordinates rx2, ry2, rx3, ry3
c) Draw the rotated triangle in the following coordinates (rx1, ry1, rx2, ry2, rx3, ry3).
5. Get the translation factors(x, y) and draw the translated triangle in the following coordinates
   (x1+x, y1+y, x2+x, y2+y, x3+x, y3+y).
6. Rotate the triangle in 1800 at (x2, y2) and draw the rotated triangle (which is reflected
   triangle).
7. Get the scaling factors(x, y) and draw the scaled triangle in the following coordinates (x1*x,
   y1*y, x2*x, y2*y, x3*x, y3*y).
8. (i) Draw the rectangle in window port coordinates (w1, w2, w3, w4) and draw the original
      triangle.
   (ii) Calculate the x, y and view port coordinates by following formulae:
        x = (v3 - v1) / (w3 - w1);
        y = (v4 - v2) / (w4 - w2);
        vx1 = v1 + floor (((x1 - w1) * x) + 0.5);
        vy1 = v2 + floor (((y1 - w2) * y) + 0.5);
   (iii) Similarly calculate the coordinates vx2, vy2, vx3, vy3.
   (iv) Draw the rectangle in view port coordinates (v1, v2, v3, v4) and draw the triangle in
        View port by the following coordinates (vx1, vy1, vx2, vy2, vx3, vy3).
```

PROGRAM:

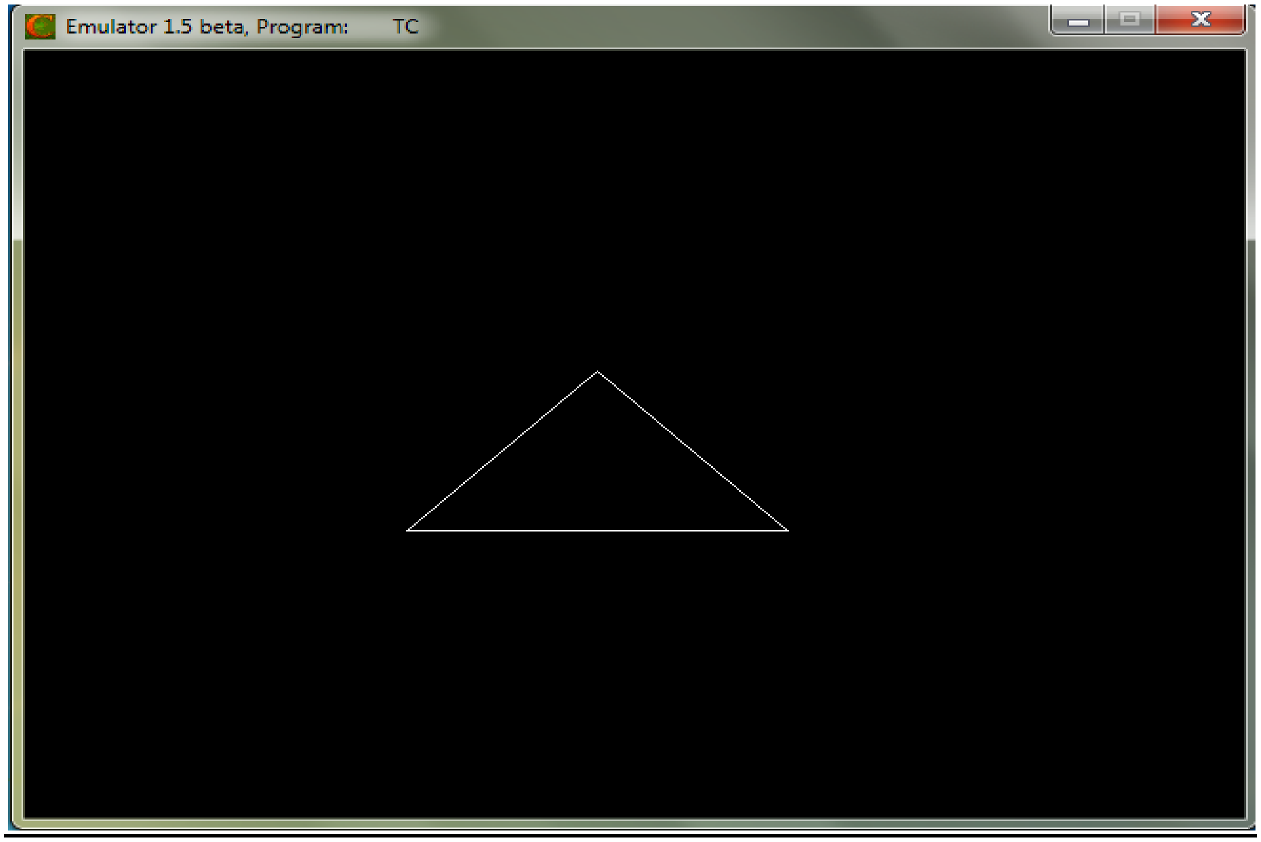
```
#include<graphics.h>
#include<iostream.h>
#include<conio.h>
```

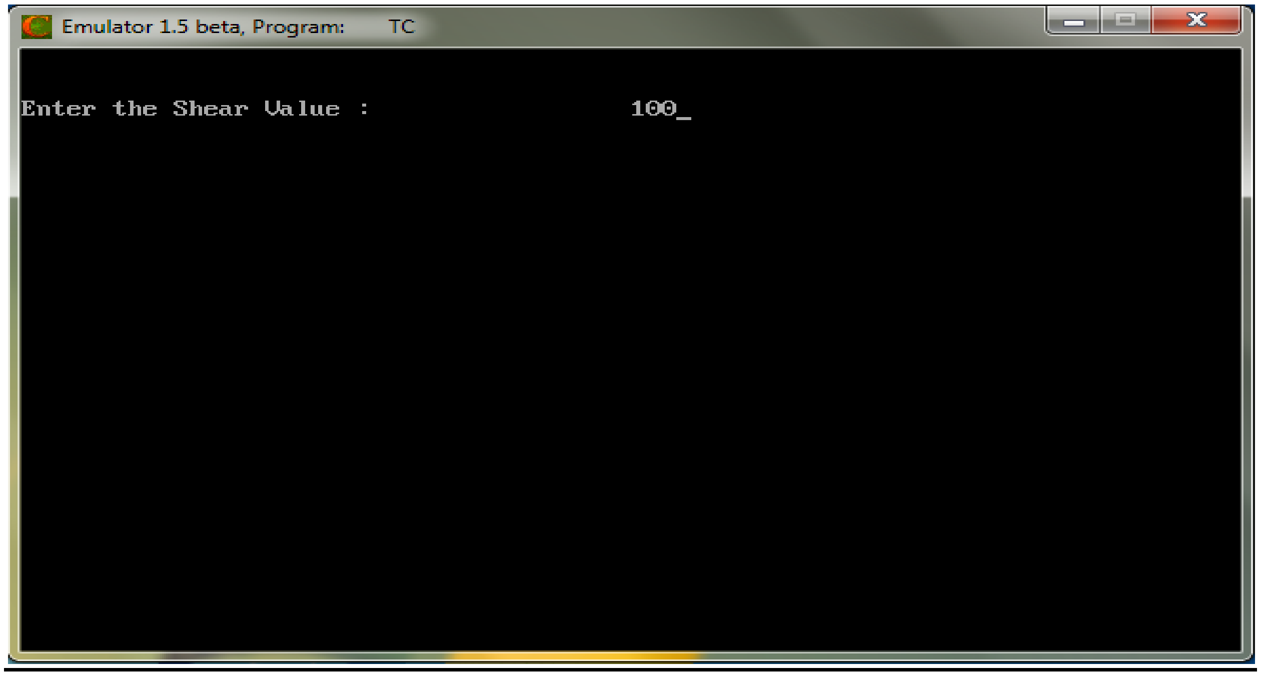
```
#include<math.h>
float x1,y1,x2,y2,x3,y3,x,y,rx1,ry1,rx2,ry2,rx3,ry3,rx,ry,t,w1=5,w2=5,w3=635,w4=465,v1=425,
v2=75,v3=550,v4=250;
int gd,gm,i;
void triangle(float,float,float,float,float,float);
void rotate(float,float,float);
void main()
{
        clrscr();
        cout<<"\n\t\t***** Composite 2D Transformations *****";
        cout<<"\n\nEnter the coordinates of triangle (x1,y1,x2,y2,x3,y3) : ";
        cin>>x1>>y1>>x2>>y2>>x3>>y3;
        initgraph(&gd,&gm,"C:\\TC\\BGI");
        triangle(x1,y1,x2,y2,x3,y3);
        closegraph();
        cout<<"\n\nEnter the Shear Value : \t\t";
        cin>>x;
        initgraph(&gd,&gm,"C:\\TC\\BGI");
        triangle(x1,y1,x2,y2,x3,y3);
        x2=x2+x;
        triangle(x1,y1,x2,y2,x3,y3);
        closegraph();
        cout<<"\n\nEnter the angle of rotation : \t\t";
        cin>>t;
        cout<<"\nEnter the reference point of rotation (rx,ry) : \t";
        cin>>rx>>ry;
        initgraph(&gd,&gm,"C:\\TC\\BGI");
        triangle(x1,y1,x2,y2,x3,y3);
        rotate(t,rx,ry);
        closegraph();
        cout<<"\n\nEnter the translation factors (x,y): \t\t";
        cin>>x>>y;
        initgraph(&gd,&gm,"C:\\TC\\BGI");
        triangle(x1,y1,x2,y2,x3,y3);
        x1=x1+x;
        y1=y1+y;
        x2=x2+x;
        y2=y2+y;
        x3=x3+x;
        y3=y3+y;
        triangle(x1,y1,x2,y2,x3,y3);
        closegraph();

        initgraph(&gd,&gm,"C:\\TC\\BGI");
        triangle(x1,y1,x2,y2,x3,y3);
        rotate(180,x2,y2);
        closegraph();
        cout<<"\n\nEnter the scaling factors (x,y): \t\t";
        cin>>x>>y;
        initgraph(&gd,&gm,"C:\\TC\\BGI");
        triangle(x1,y1,x2,y2,x3,y3);
        x1=x1*x;
        y1=y1*y;
        x2=x2*x;
        y2=y2*y;
        x3=x3*x;
        y3=y3*y;
        triangle(x1,y1,x2,y2,x3,y3);
        closegraph();
        initgraph(&gd,&gm,"C:\\TC\\BGI");
        rectangle(w1,w2,w3,w4);
        outtextxy(300,10,"Window Port");
        triangle(x1,y1,x2,y2,x3,y3);
        x=(v3-v1)/(w3-w1);
        y=(v4-v2)/(w4-w2);
        x1=v1+floor(((x1-w1)*x)+0.5);
        y1=v2+floor(((y1-w2)*y)+0.5);
        x2=v1+floor(((x2-w1)*x)+0.5);
        y2=v2+floor(((y2-w2)*y)+0.5);
        x3=v1+floor(((x3-w1)*x)+0.5);
        y3=v2+floor(((y3-w2)*y)+0.5);
        rectangle(v1,v2,v3,v4);
        outtextxy(450,85,"View Port");
        triangle(x1,y1,x2,y2,x3,y3);
}
void triangle(float x1,float y1,float x2,float y2,float x3,float y3)
{
        line(x1,y1,x2,y2);
        line(x2,y2,x3,y3);
        line(x3,y3,x1,y1);
        getch();
}

void rotate(float t,float rx,float ry)
{
        t=t*(3.14/180);
        rx1=rx+(x1-rx)*cos(t)-(y1-ry)*sin(t);
        ry1=ry+(x1-rx)*sin(t)+(y1-ry)*cos(t);
        rx2=rx+(x2-rx)*cos(t)-(y2-ry)*sin(t);
        ry2=ry+(x2-rx)*sin(t)+(y2-ry)*cos(t);
        rx3=rx+(x3-rx)*cos(t)-(y3-ry)*sin(t);
        ry3=ry+(x3-rx)*sin(t)+(y3-ry)*cos(t);
        x1=rx1;
        y1=ry1;
        x2=rx2;
        y2=ry2;
        x3=rx3;
        y3=ry3;
        triangle(x1,y1,x2,y2,x3,y3);
}
```

OUTPUT:

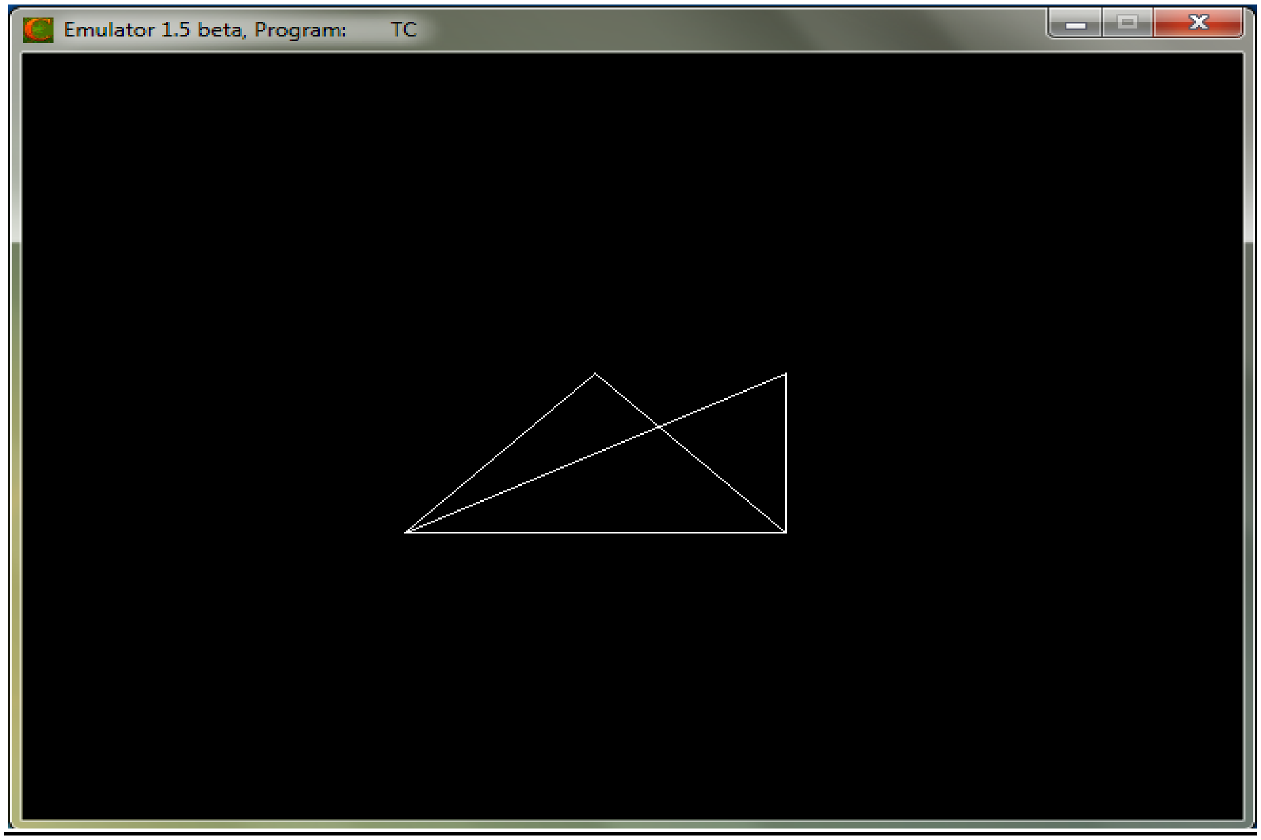


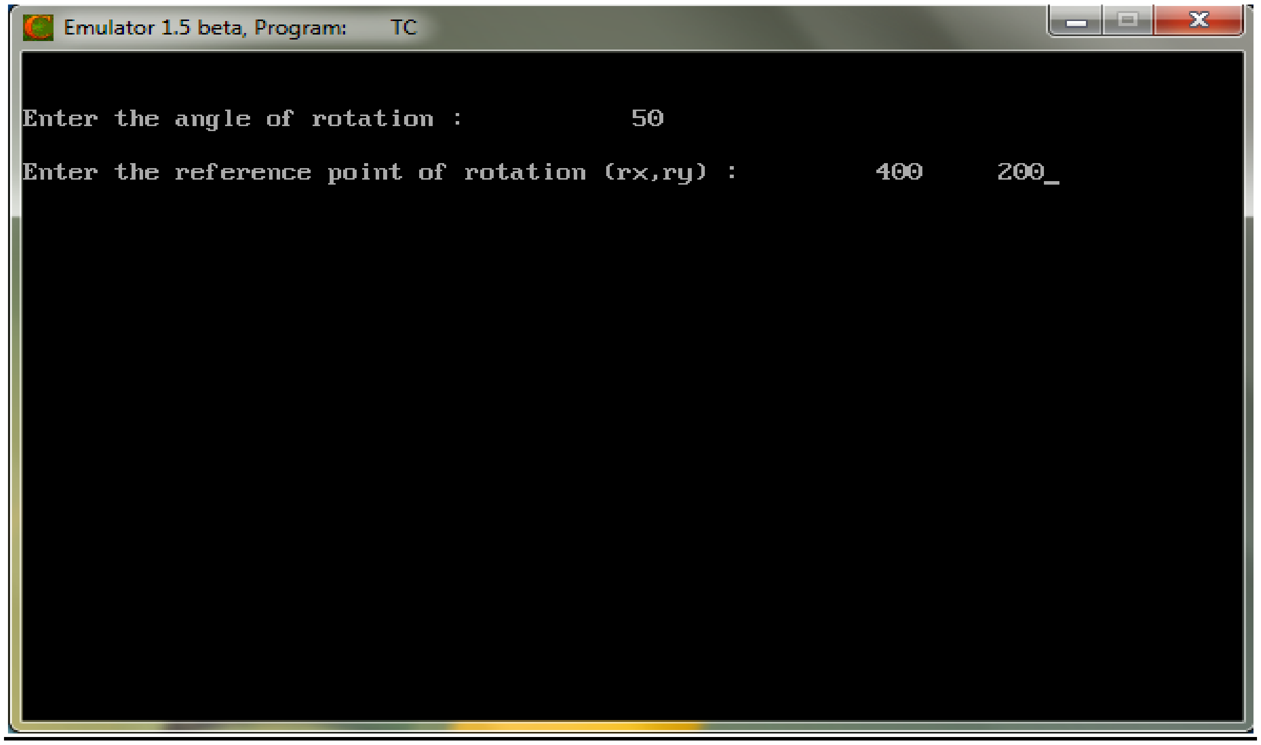
```
                ***** Composite 2D Transformations *****

Enter the coordinates of triangle (x1,y1,x2,y2,x3,y3) : 200     300     300
200     400     300
```



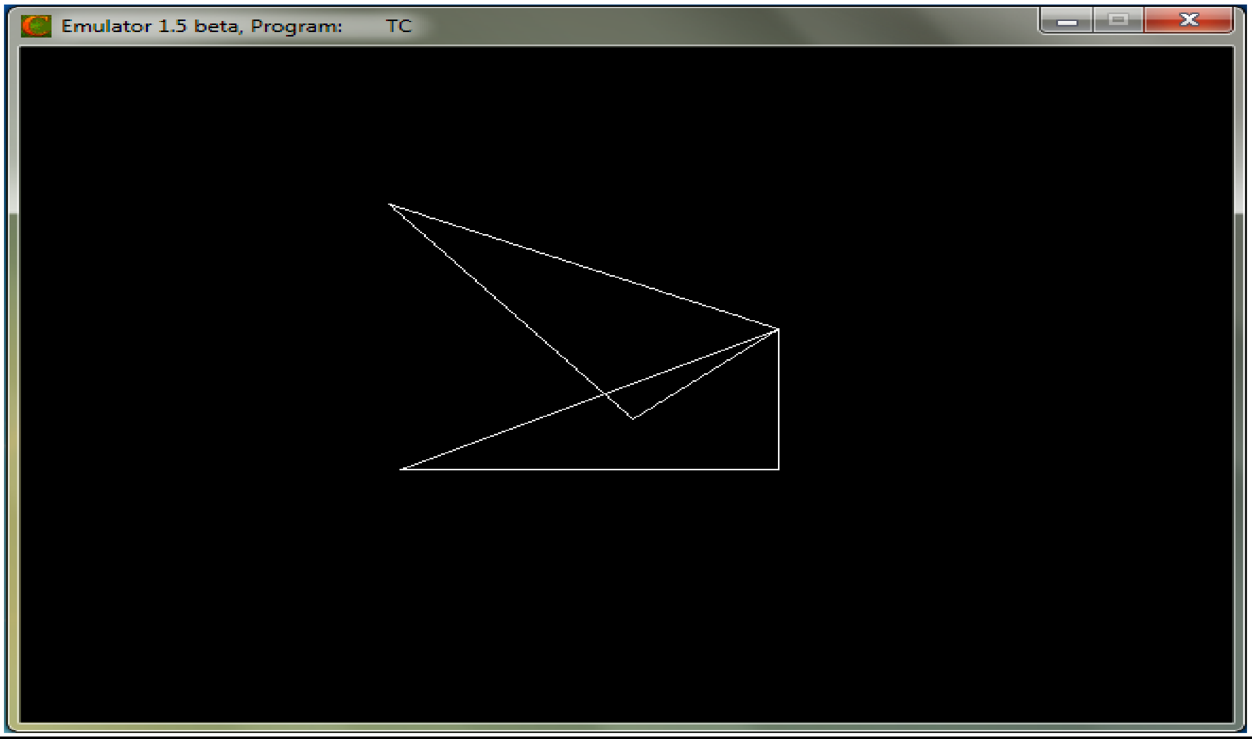


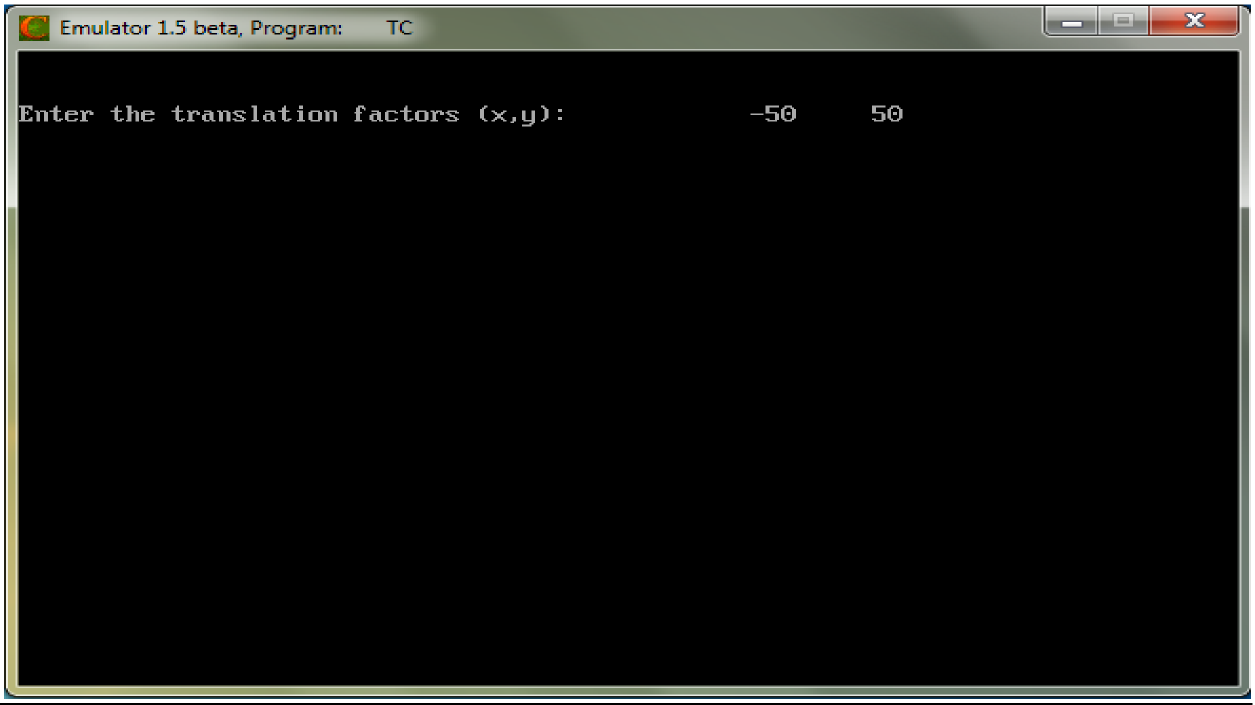
```
Enter the Shear Value :                    100_
```



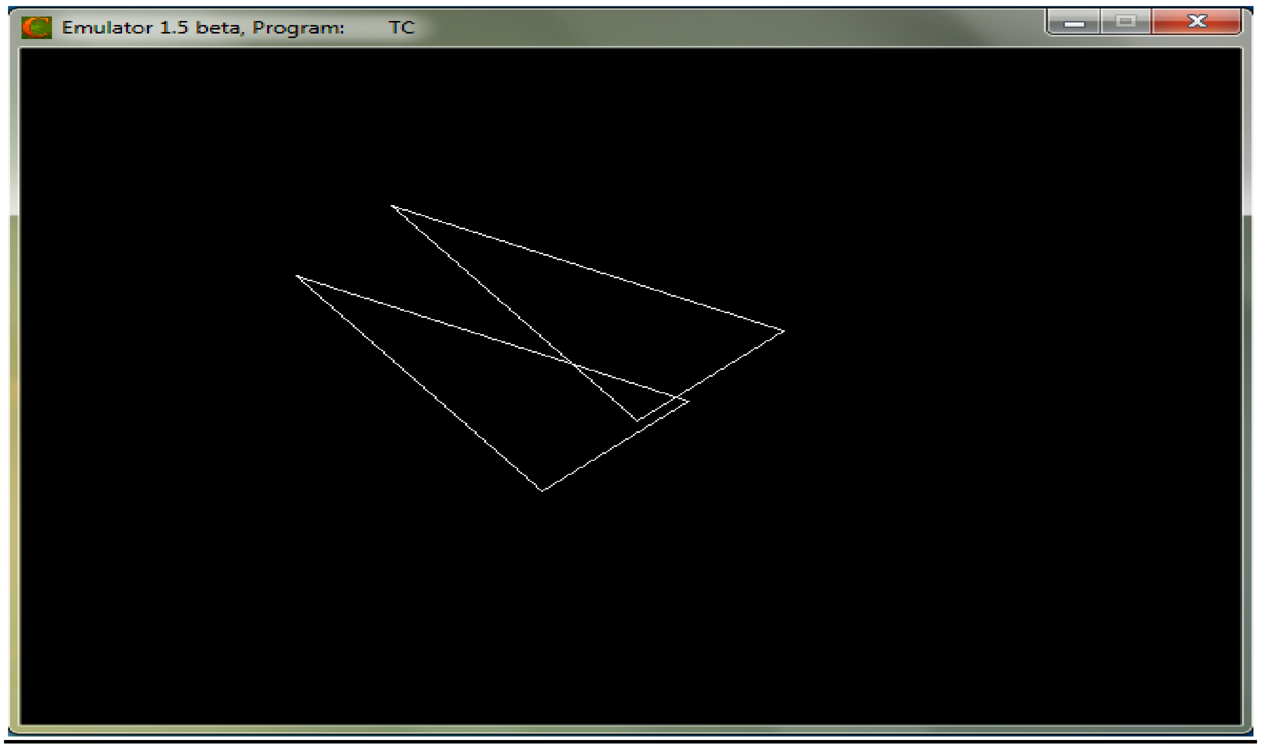


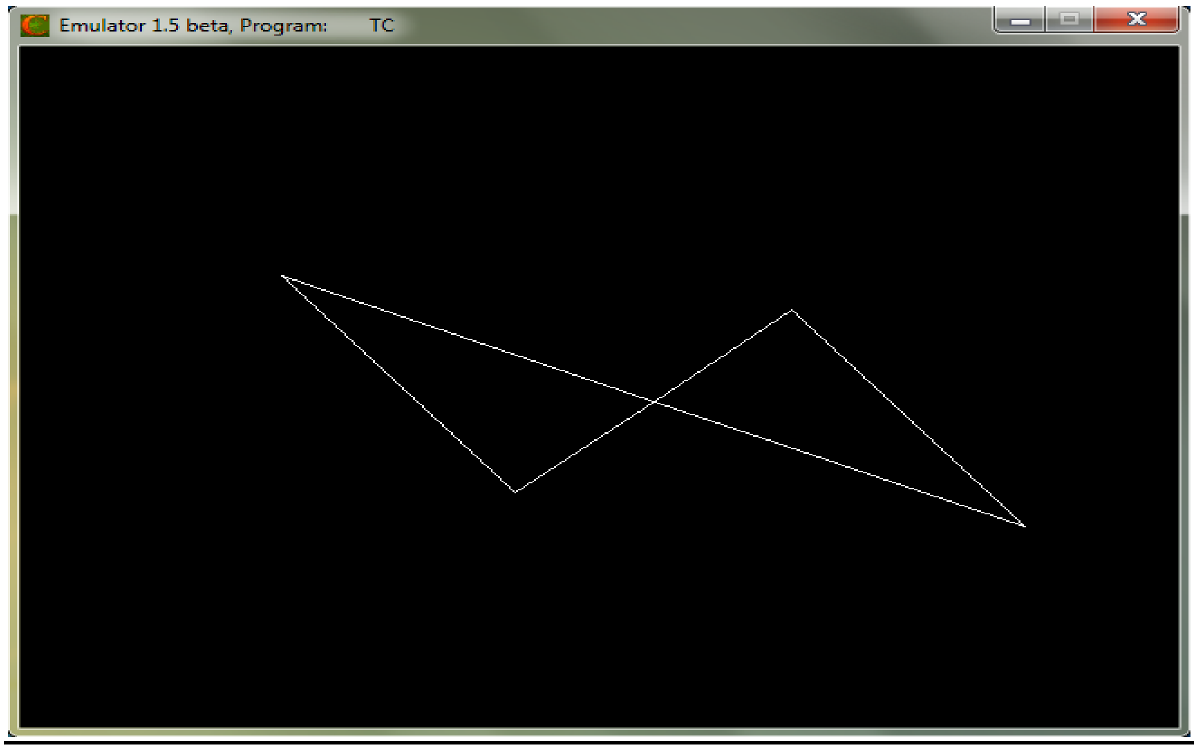
```
Enter the angle of rotation :           50

Enter the reference point of rotation (rx,ry) :         400     200_
```

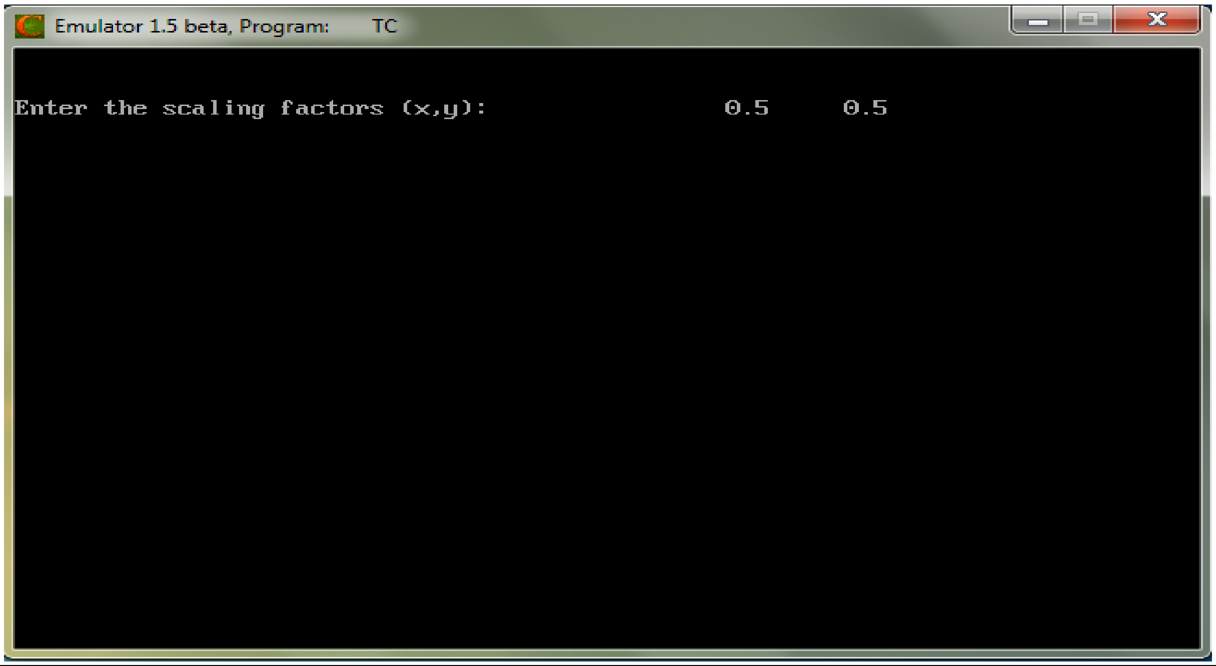




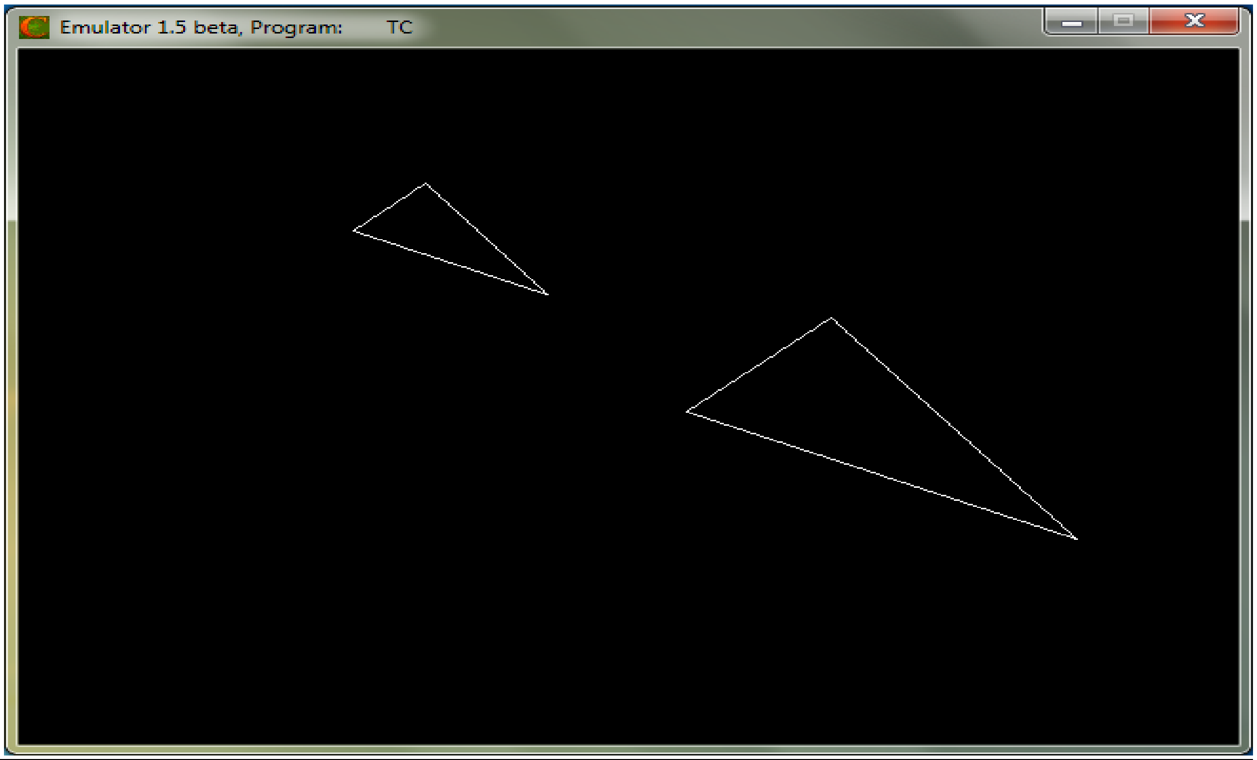
```
Enter the translation factors (x,y):            -50     50
```

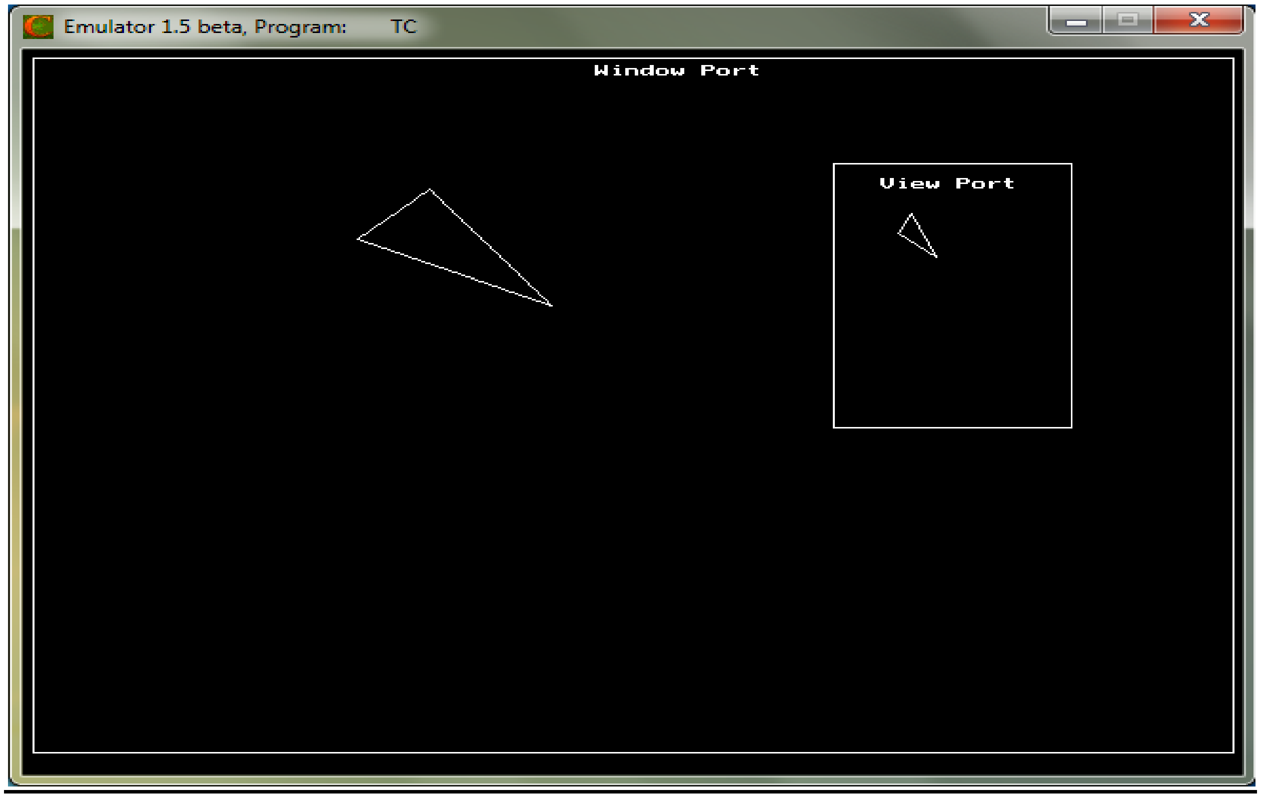

Emulator 1.5 beta, Program: TC


Emulator 1.5 beta, Program: TC



```
Enter the scaling factors (x,y):              0.5     0.5
```





RESULT:

Thus the program to implementation of composite 2D transformations has been implemented and the output was verified.

VIVA QUESTIONS

1. **What is a composite transformation?**
   Composite transformation can be achieved by concatenation of transformation matrices to obtain a combined transformation matrix. **[T][X] = [X] [T1] [T2] [T3] [T4] …. [Tn]**

2. **What is rotation about an arbitrary point?**
   To rotate an object about an arbitrary point (X_p, Y_p), we have to carry out three steps − Translate point (X_p, Y_p) to the origin.Rotate it about the origin.Finally, translate the center of rotation back where it belonged.

3. **What are the basic transformations?**
   - Translation
   - Shearing
   - Rotation
   - Translation
   - Reflection
   - Scaling
   - Window - View Port Transformation

4. **What is Transformation?**
   Transformation is the process of introducing changes in the shape size and orientation of the object using scaling rotation reflection shearing & translation etc.

5. **What is translation?**
   Translation is the process of changing the position of an object in a straight-line path from one coordinate location to another. Every point (x, y) in the object must undergo a displacement to (x´,y´). the transformation is:
   x´ = x + tx
   y´ = y+ty

6. **What is rotation?**
   A 2-D rotation is done by repositioning the coordinates along a circular path, in X = rcos (q + f) and Y = r sin (q + f).

7. **What is scaling?**
   The scaling transformations changes the shape of an object and can be carried out by multiplying each vertex (x,y) by scaling factor Sx,Sy where Sx is the scaling factor of x and Sy is the scaling factor of y.

8. **What is shearing?**
   The shearing transformation actually slants the object along the X direction or the Y direction as required.ie; this transformation slants the shape of an object along a required plane.

9. **Define Affine transformation.**
   A coordinate transformation of the form X= axxx +axyy+bx, y ‟ayxx+ayyy+by is called a two-dimensional affine transformation. Each of the transformed

coordinates x „and y „is a linear function of the original coordinates x and y, and parameters aij and bk are constants determined by the transformation type.

10. **What is the need of homogeneous coordinates?**
To perform more than one transformation at a time, use homogeneous coordinates or matrixes. They reduce unwanted calculations intermediate steps saves time and memory and produce a sequence of transformations

APPLICATIONS

- Video Games
- Simulators (flight, driving)
- Virtual reality
- Web Design
- Advertising

EX.NO: 4 LINE CLIPPING

DATE:

AIM:

To write a program to implement the line clipping.

ALGORITHM:

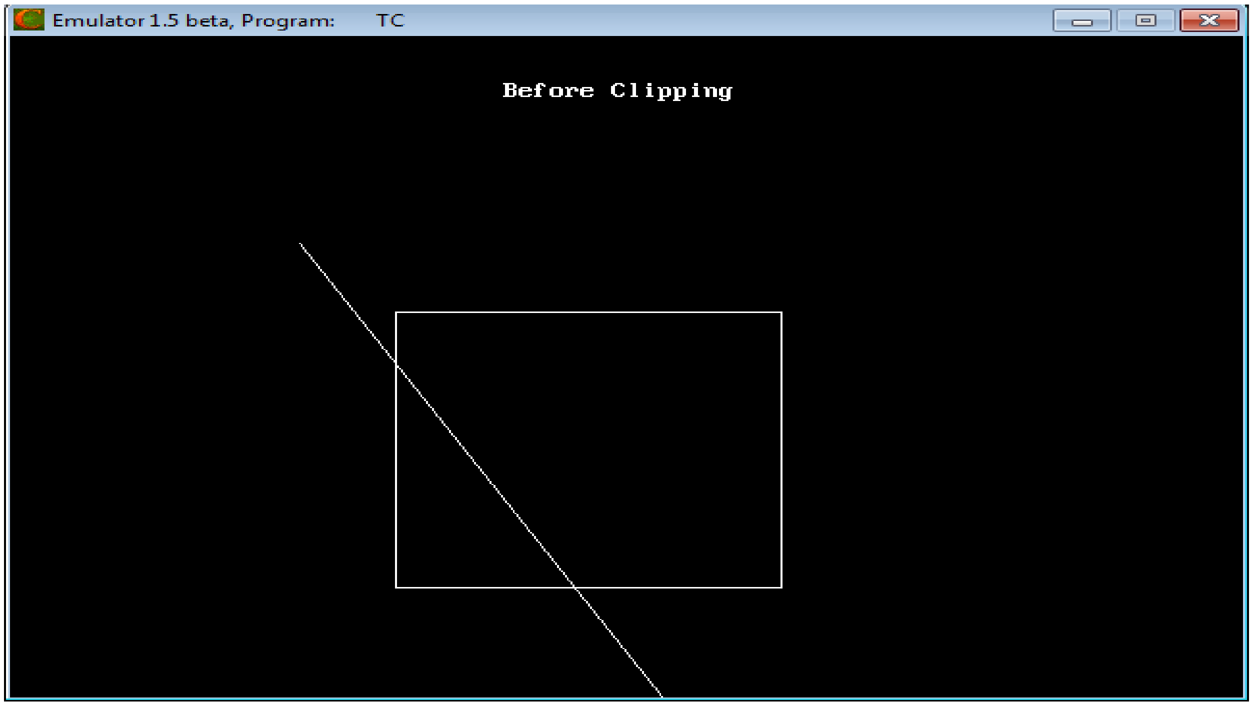
1. Get the clip window coordinates.
2. Get the line end points.
3. Draw the window and the line.
4. Remove the line points which are plotted in outside the window.
5. Draw the window with clipped line.

PROGRAM:

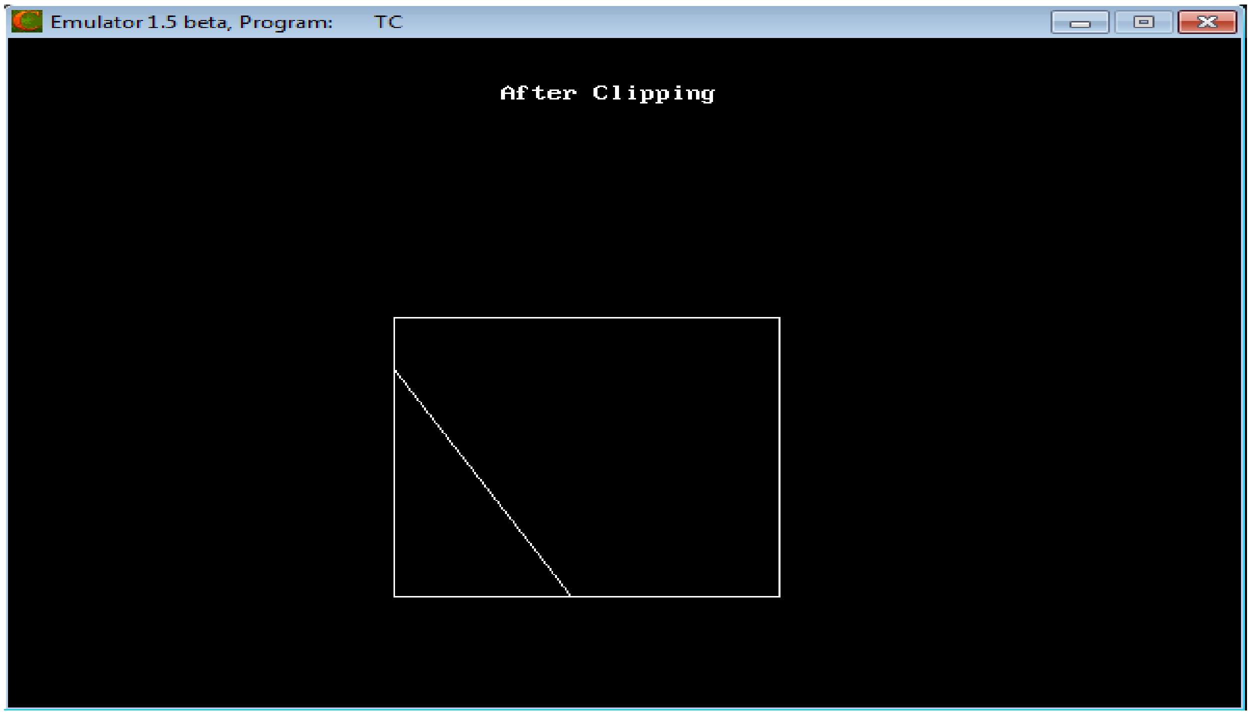
```
#include<graphics.h>
#include<iostream.h>
#include<conio.h>
float x1,y1,x2,y2,wx1,wy1,wx2,wy2,m;
int gd,gm;
void main()
{
        clrscr();
        cout<<"\nEnter the clip window coordinates : ";
        cin>>wx1>>wy1>>wx2>>wy2;
        cout<<"\nEnter the line end points : ";
        cin>>x1>>y1>>x2>>y2;
        m=(y2-y1)/(x2-x1);
        initgraph(&gd,&gm,"C:\\TC\\BGI");
        cout<<"\n\n\t\t\t\tBefore Clipping";
        rectangle(wx1,wy1,wx2,wy2);
        line(x1,y1,x2,y2);
        getch();
        closegraph();
        if(x1<wx1)
        {
                y1=y1+(wx1-x1)*m;
                x1=wx1;
        }
        if(x2>wx2)
        {
                y2=y2-(x2-wx2)*m;
                x2=wx2;
        }
        if(y2>wy2)
        {
                x2=x2-((y2-wy2)/m);
                y2=wy2;
        }
        if(y1<wy1)
        {
                x1=x1+(wy1-y1)/m;
                y1=wy1;
        }
        if(x2<wx1)
        {
                y2=y2+(wx1-x2)*m;
                x2=wx1;
        }
        if(x1>wx2)
        {
                y1=y1-(x1-wx2)*m;
                x1=wx2;
        }
        if(y2<wy1)
        {
                x2=x2+((wy1-y2)/m);
                y2=wy1;
        }
        if(y1>wy2)
        {
                x1=x1-((y1-wy2)/m);
                y1=wy2;
        }
        initgraph(&gd,&gm,"C:\\TC\\BGI");
        cout<<"\n\n\t\t\t\tAfter Clipping";
        rectangle(wx1,wy1,wx2,wy2);
        line(x1,y1,x2,y2);
        getch();
}
```

OUTPUT:

```
Enter the clip window coordinates: 200   200   400   400
Enter the line end points: 150   150   350   500
```



Before Clipping



After Clipping

RESULT:

Thus the program to implementation of the line clipping has been implemented and the output was verified.

VIVA QUESTIONS

1. **Define clipping**
   Clipping is the method of cutting a graphics display to neatly fit a predefined graphics region or the view port.
2. **What are the types of clipping?**
   - Point clipping
   - Line clipping
   - Area clipping
   - Curve clipping
   - Text clipping
3. **Define Line clipping?**
   The concept of line clipping is same as point clipping. In line clipping, we will cut the portion of line which is outside of window and keep only the portion that is inside the window.
4. **What is the use of clipping?**
   Clipping in computer graphics is to remove objects, lines or line segments that are outside the viewing volume.
5. **List the different types of text clipping methods available?**
   - All-or-none string clipping - if all of the string is inside a clip window, keep it otherwise discards.
   - All-or-none character clipping – discard only those characters that are not completely inside the window. Any character that either overlaps or is outside a window boundary is clipped.
   - Individual characters – if an individual character overlaps a clip window boundary, clip off the parts of the character that are outside the window.
6. **How will you clip a point?**
   Assuming that the clip window is a rectangle in standard position, we save a point P=(x,y) for display if the following inequalities are satisfied:
   $xwmin \le x \le xwmax$ $ywmin \le y \le ywmax$
7. **Define Exterior clipping**
   This is just opposite to clipping. This removes the lines coming inside the windows and displays the remaining. Covering is mainly used to make labels on the complex pictures.
8. **What is the need of homogeneous coordinates?**
   To perform more than one transformation at a time, use homogeneous coordinates or matrixes. They reduce unwanted calculations intermediate steps saves time and memory and produce a sequence of transformations.
9. **Define text clipping.**
   Various techniques are used to provide text clipping in a computer graphics. It depends on the methods used to generate characters and the requirements of a particular application.

10. **Define point clipping**

Point clipping tells us whether the given point (X, Y) is within the given window or not; and decides whether we will use the minimum and maximum coordinates of the window.

APPLICATIONS

- Drawing and Painting operations: It allows part of a picture to be selected for copying, moving, erasing or duplicating depending on the application.
- Computer Aided Design (CAD)
- Picture enhancements
- Graphic design

EX.NO: 5 IMPLEMENTATION OF 3D TRANSFORMATIONS

DATE:

AIM:

To implement the following 3D Transformations:

a. Translation b. Rotation c. Scaling

ALGORITHM:

1. Assign the value of edge [20] [3] as a coordinates of cube.
2. Print the menu for choosing 3D Transformation.
3. If user choose translation then get the translation factor (tx, ty, tz) and draw the translated cube.
4. If user choose rotation then print the menu for choosing rotation axis and get the rotation angle.
   a) If user choose rotation about X axis then rotate the cube along X axis and draw the rotated cube.
   b) If user choose rotation about Y axis then rotate the cube along Y axis and draw the rotated cube.
   c) If user choose rotation about Z axis then rotate the cube along Z axis and draw the rotated cube.
5. If user choose scaling then get the scaling factor (sx, sy, sz) and draw the scaled cube.

PROGRAM:

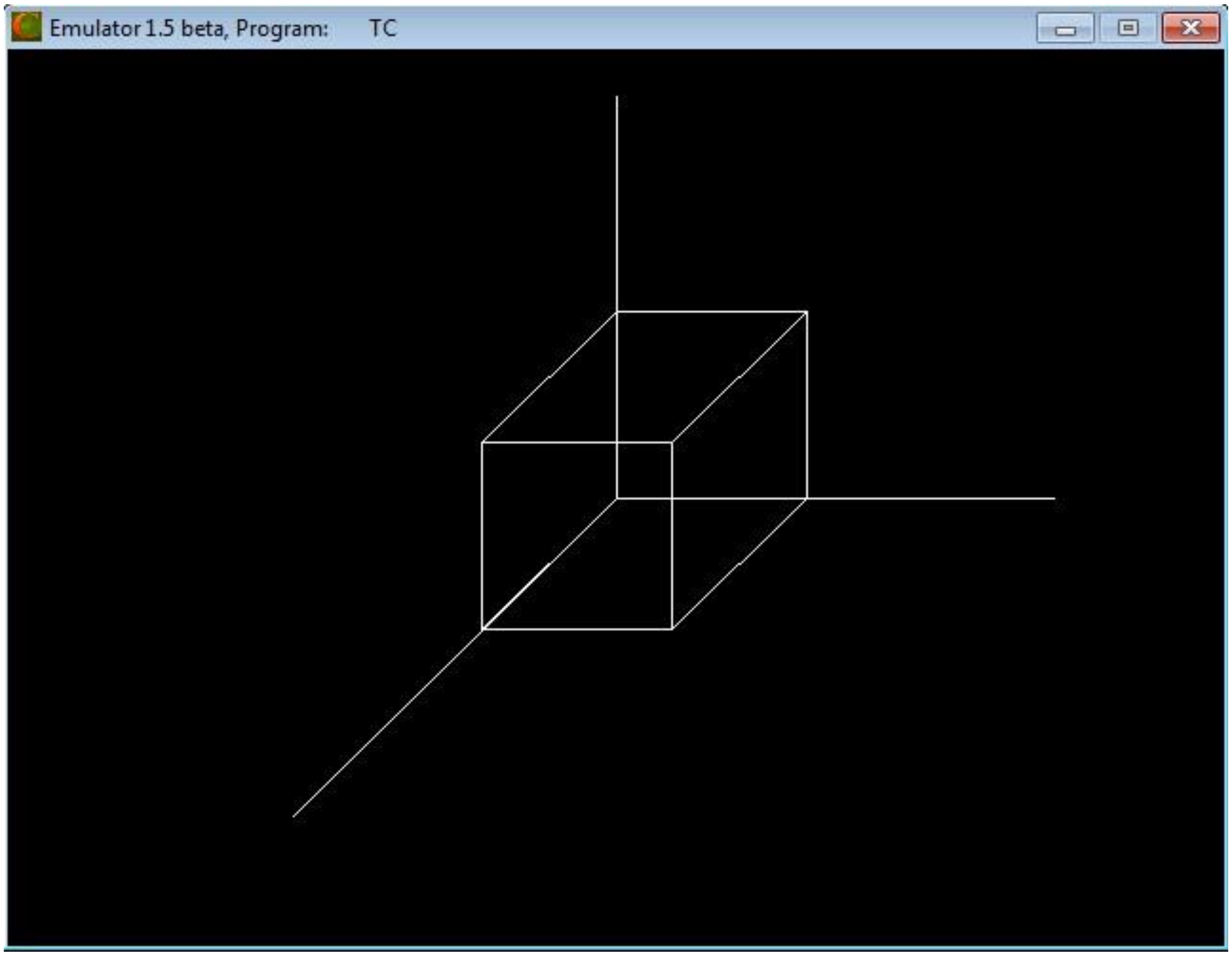
```
#include<graphics.h>
#include<iostream.h>
#include<conio.h>
#include<math.h>
int gd,gm,i,ch1,ch2;
double x1,x2,y1,y2,x,y,z,theta,temp,temp1;
void draw(double[20][3]);
void main()
{
        do
        {
                double edge[20][3]={ 100,0,0,
                100,100,0,
                0,100,0,
                0,100,100,
                0,0,100,
                0,0,0,
                100,0,0,
                100,0,100,
                100,100,100,
                100,100,100,
                100,100,100,
                100,100,0,
		100,100,100,
		100,100,100,
		100,100,100,
		0,100,100,
		0,100,0,
		0,0,0,
		0,0,100,
		100,0,100
	};
	clrscr();
	cout<<"Choose any one 3D Transformation : ";
	cout<<"\n\t1. Translation ";
	cout<<"\n\t2. Rotation ";
	cout<<"\n\t3. Scaling ";
	cout<<"\nEnter your choice : \t";
	cin>>ch1;
	switch(ch1)
	{
		case 1:
			cout<<"\nEnter the translation factors(tx,ty,tz) : \t";
			cin>>x>>y>>z;
			draw(edge);
			for(i=0;i<20;i++)
			{
				edge[i][0]+=x;
				edge[i][1]+=y;
				edge[i][2]+=z;
			}
			draw(edge);
			break;
		case 2:
			cout<<"\n\n\t1.Rotation about X Axis ";
			cout<<"\n\t2. Rotation about Y Axis ";
			cout<<"\n\t3. Rotation about Z Axis ";
			cout<<"\nEnter your choice : \t";
			cin>>ch2;
			cout<<"\n\nEnter the angle of rotation : \t";
			cin>>theta;
			theta=(theta*3.14)/180;
			switch(ch2)
			{
				case 1:
					draw(edge);
					for(i=0;i<20;i++)
					{
						temp=edge[i][1];
						temp1=edge[i][2];
						edge[i][1]=temp*cos(theta)-temp1*sin(theta);
```

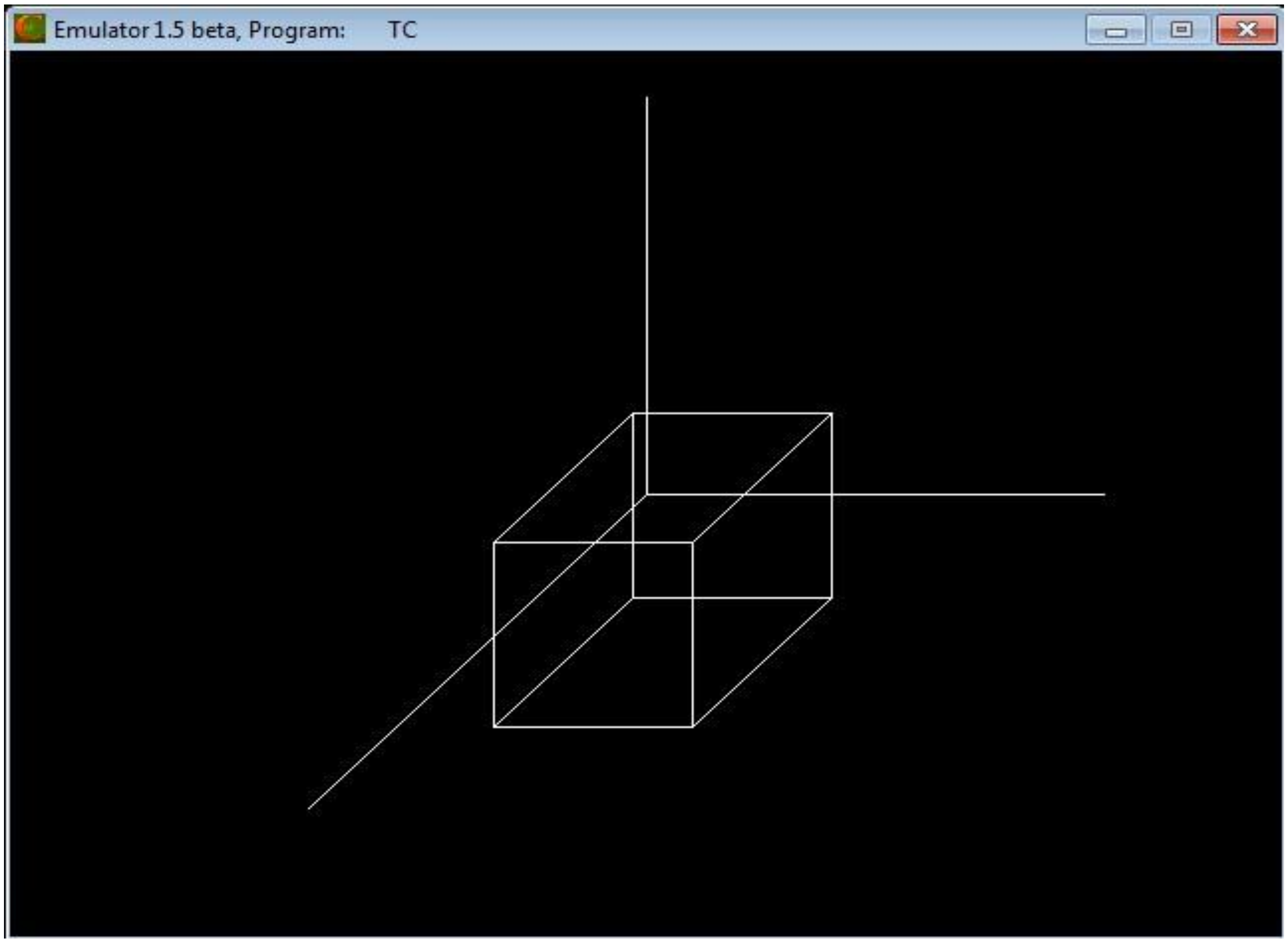
```
                    edge[i][2]=temp*sin(theta)+temp1*cos(theta);
                }
                draw(edge);
                break;
            case 2:
                draw(edge);
                for(i=0;i<20;i++)
                {
                    temp=edge[i][0];
                    temp1=edge[i][2];
                    edge[i][0]=temp*cos(theta)+temp1*sin(theta);
                    edge[i][2]=-temp*sin(theta)+temp1*cos(theta);
                }
                draw(edge);
                break;
            case 3:
                draw(edge);
                for(i=0;i<20;i++)
                {
                    temp=edge[i][0];
                    temp1=edge[i][1];
                    edge[i][0]=temp*cos(theta)-temp1*sin(theta);
                    edge[i][1]=temp*sin(theta)+temp1*cos(theta);
                }
                draw(edge);
                break;
                }
                break;
            case 3:
                cout<<"\n\nEnter the scaling factors (sx,sy,sz) : \t";
                cin>>x>>y>>z;
                draw(edge);
                for(i=0;i<20;i++)
                {
                    edge[i][0]*=x;
                    edge[i][1]*=y;
                    edge[i][2]*=z;
                }
                draw(edge);
                break;
            }
        }while(ch1<4);
}

void draw(double edge[20][3])
{
        initgraph(&gd,&gm,"C:\\TC\\BGI");
        line(320,240,320,25);
```

```
		line(320,240,550,240);
		line(320,240,150,410);
		for(int i=0;i<19;i++)
		{
			x1=edge[i][0]+edge[i][2]*(cos(2.3562));
			y1=edge[i][1]-edge[i][2]*(sin(2.3562));
			x2=edge[i+1][0]+edge[i+1][2]*(cos(2.3562));
			y2=edge[i+1][1]-edge[i+1][2]*(sin(2.3562));
			line(x1+320,240-y1,x2+320,240-y2);
		}
		getch();
		closegraph();
	}
```

OUTPUT:
Case 1:

```
Choose any one 3D Transformation:
1. Translation
2. Rotation
3. Scaling
Enter your choice: 1
Enter the translation factors (tx,ty,tz) : 100 50 150
```


Emulator 1.5 beta, Program: TC



Case 2 (a):

Choose any one 3D Transformation:

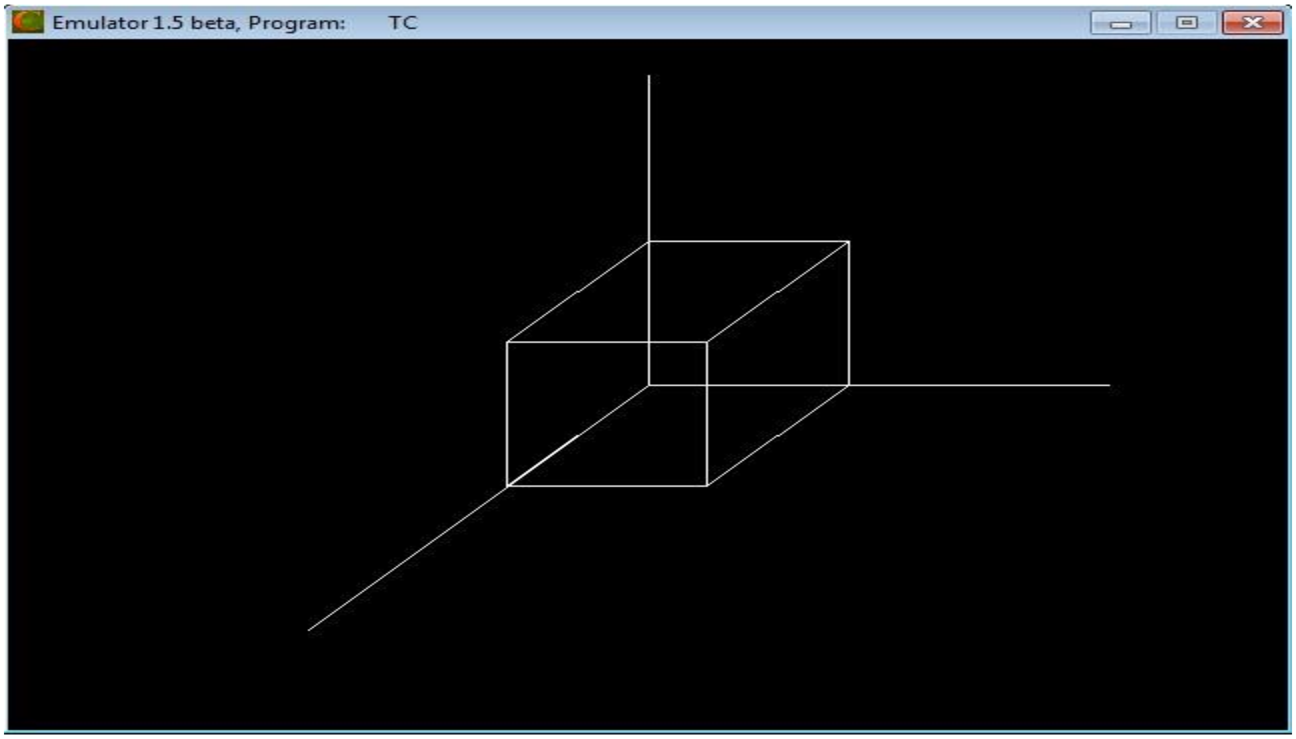
1. Translation
2. Rotation
3. Scaling

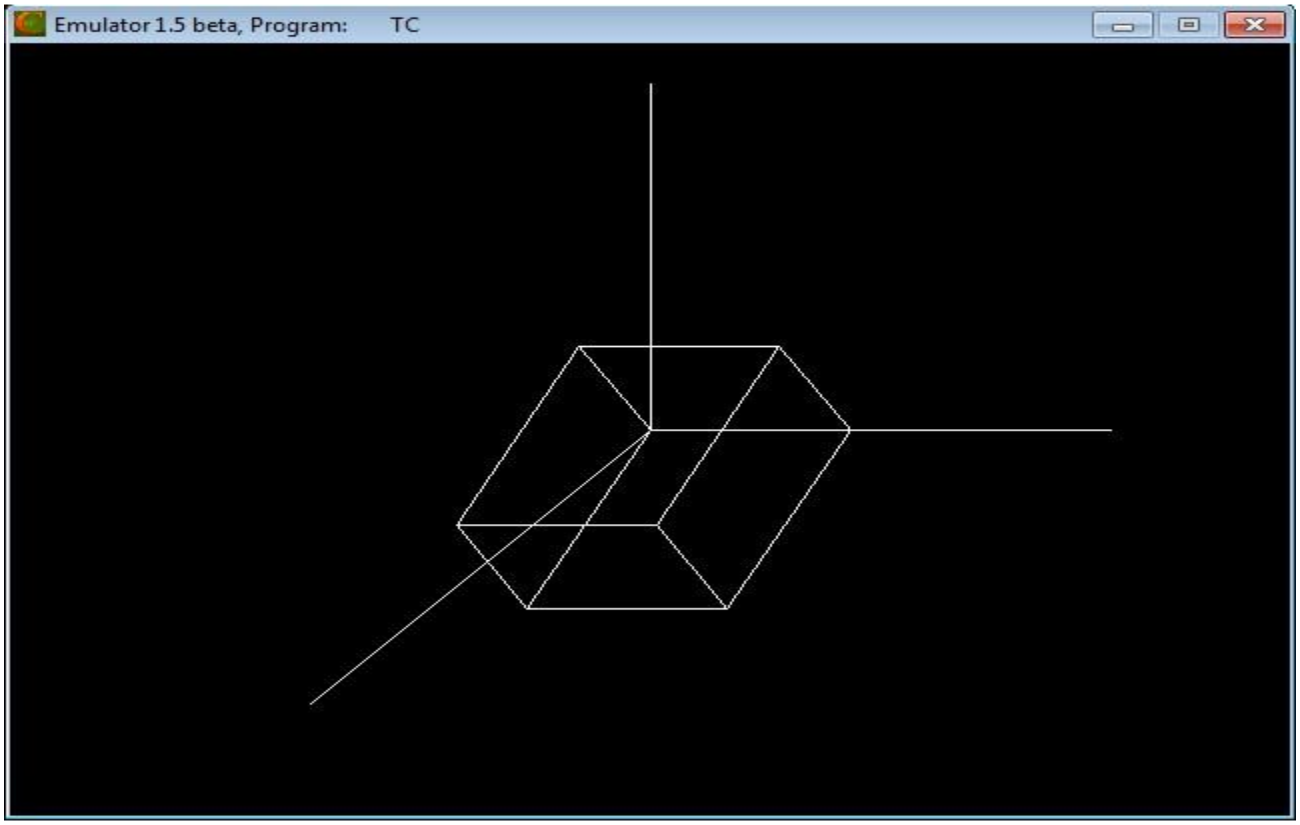
Enter your choice: 2

1. Rotation about X Axis
2. Rotation about Y Axis
3. Rotation about Z Axis

Enter your choice: 1

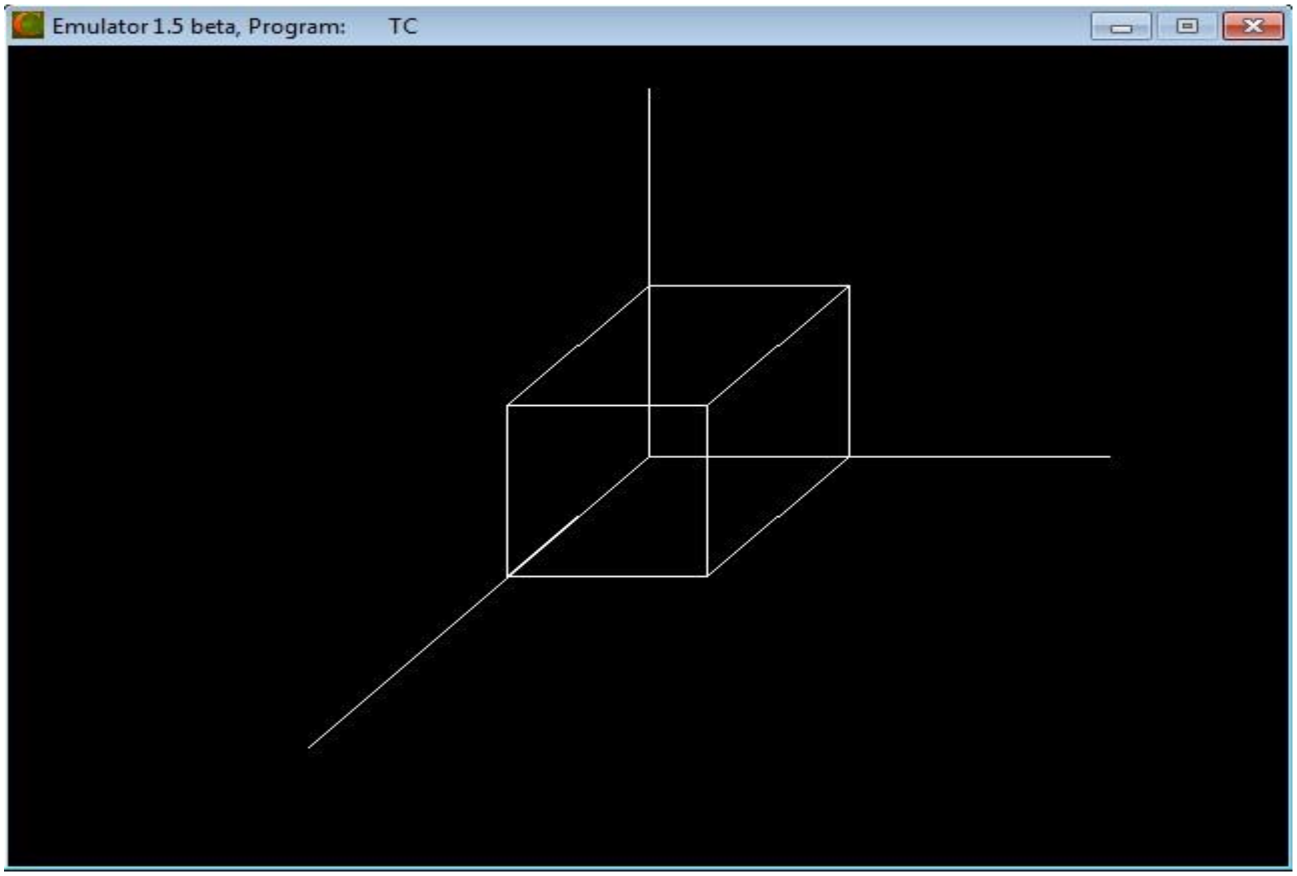
Enter the angle of rotation: 30

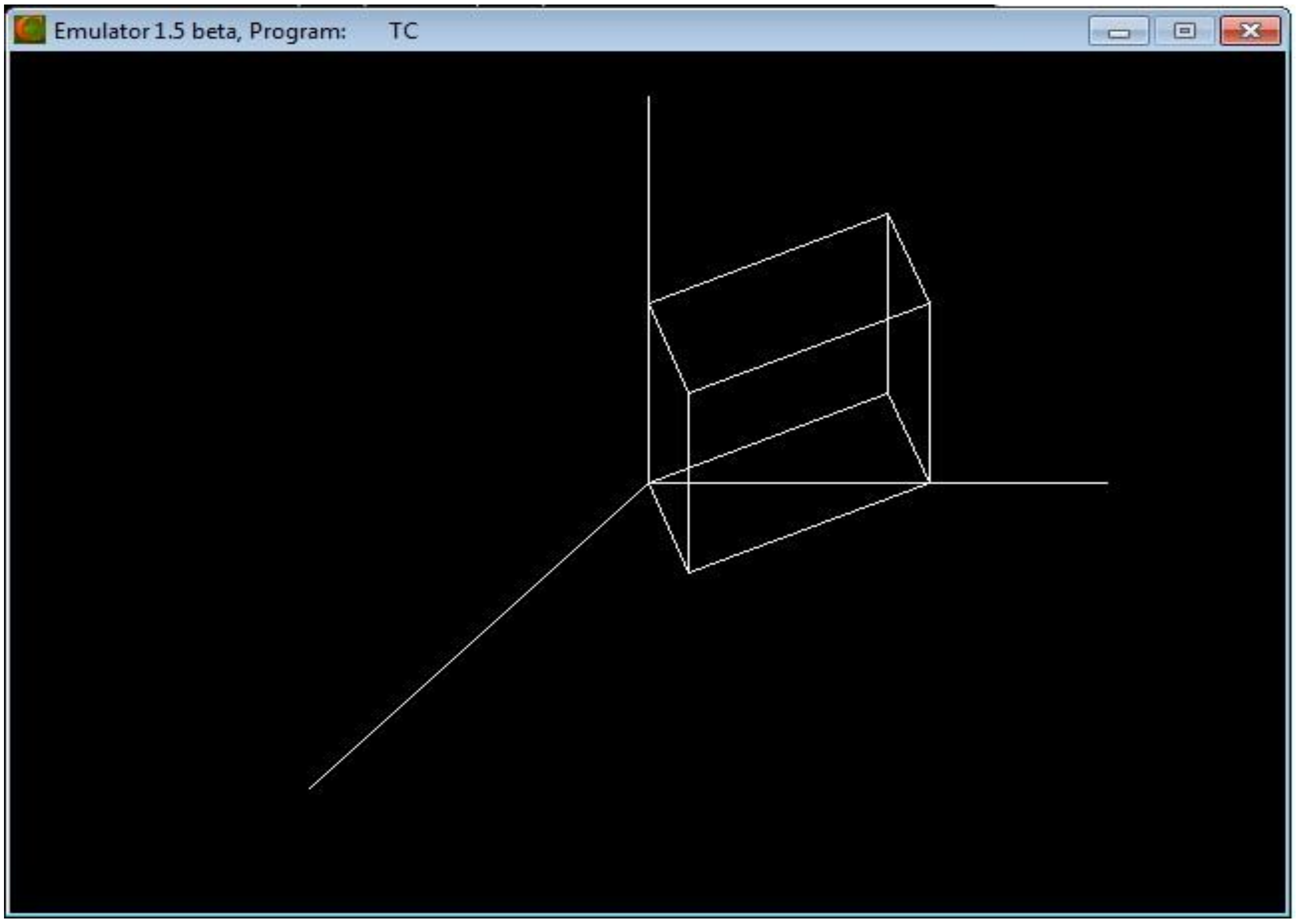

Emulator 1.5 beta, Program: TC


Emulator 1.5 beta, Program: TC

Case 2 (b):

```
Choose any one 3D Transformation:

1. Translation
2. Rotation
3. Scaling

Enter your choice: 2

1. Rotation about X Axis
2. Rotation about Y Axis
3. Rotation about Z Axis
Enter your choice: 2

Enter the angle of rotation: 45
```





Case 2 (c):

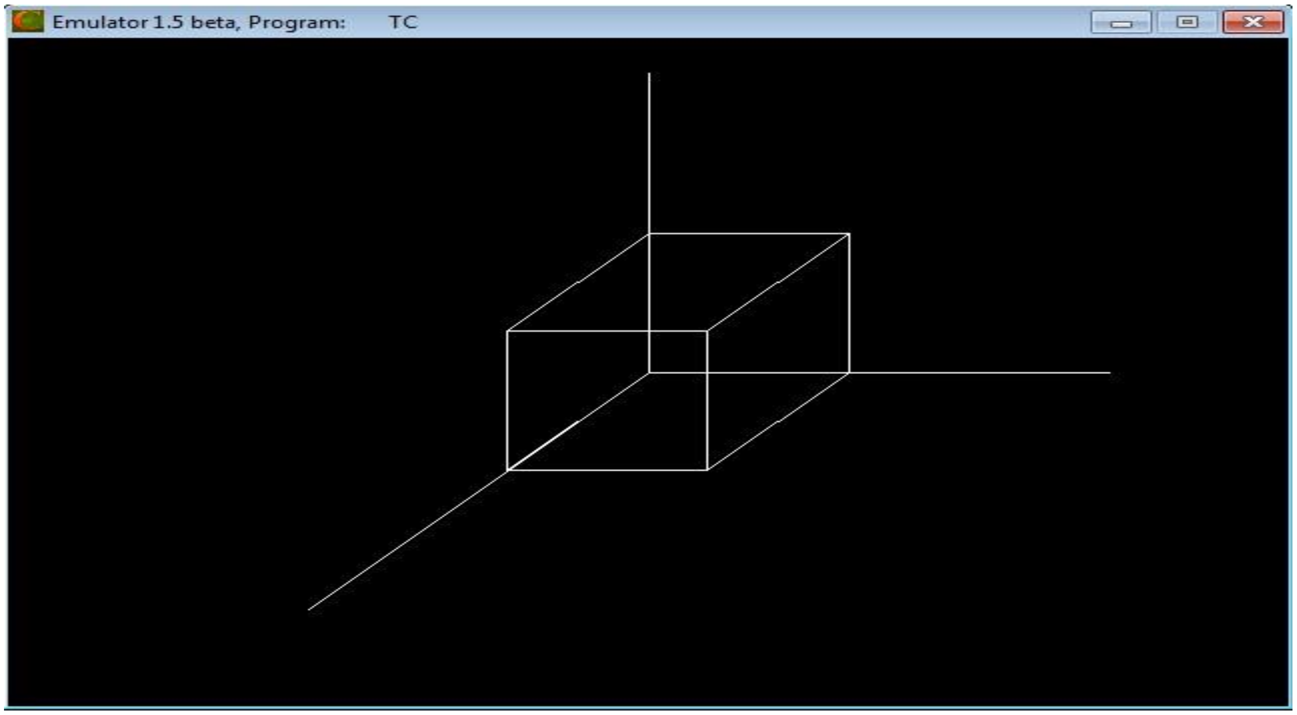
Choose any one 3D Transformation:

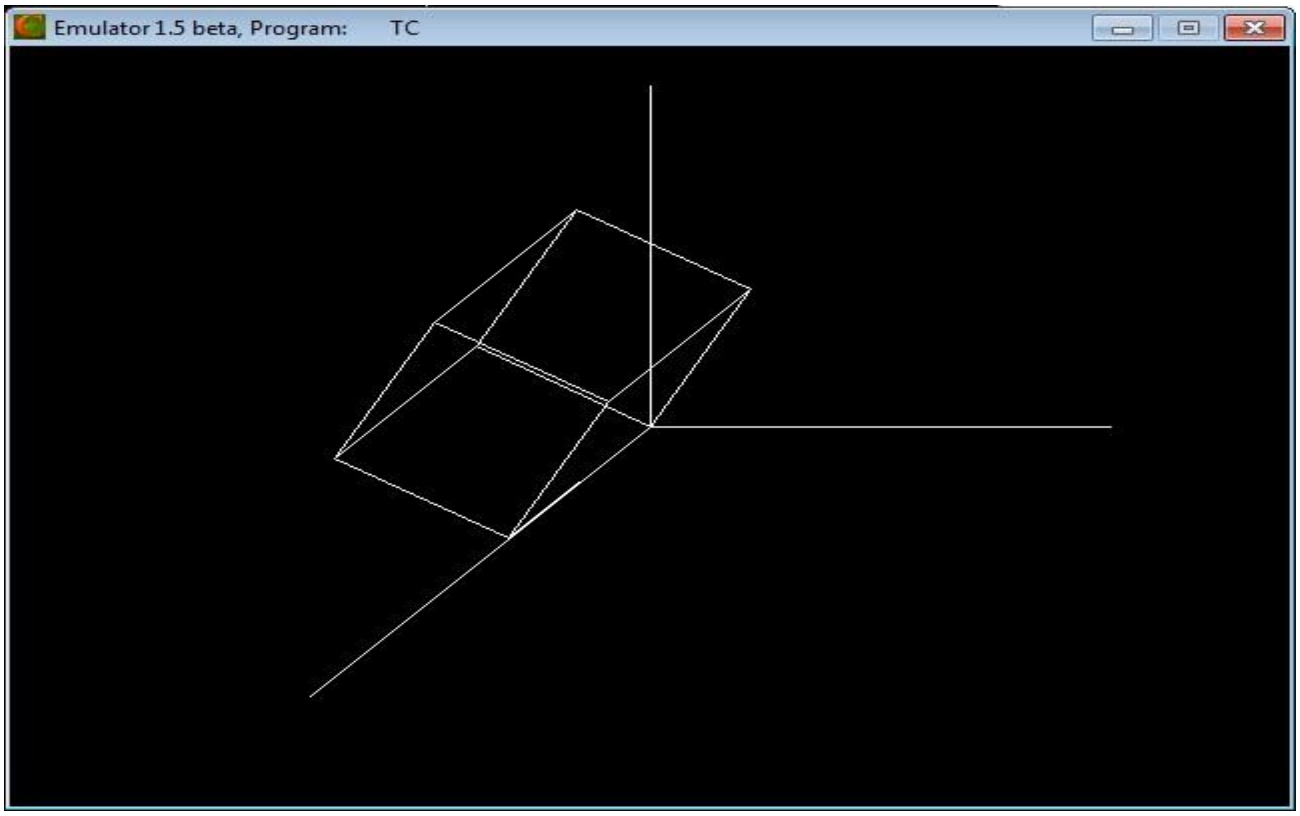
1. Translation
2. Rotation
3. Scaling

Enter your choice: 2

1. Rotation about X Axis
2. Rotation about Y Axis
3. Rotation about Z Axis
Enter your choice: 3

Enter the angle of rotation: 60


Emulator 1.5 beta, Program: TC


Emulator 1.5 beta, Program: TC

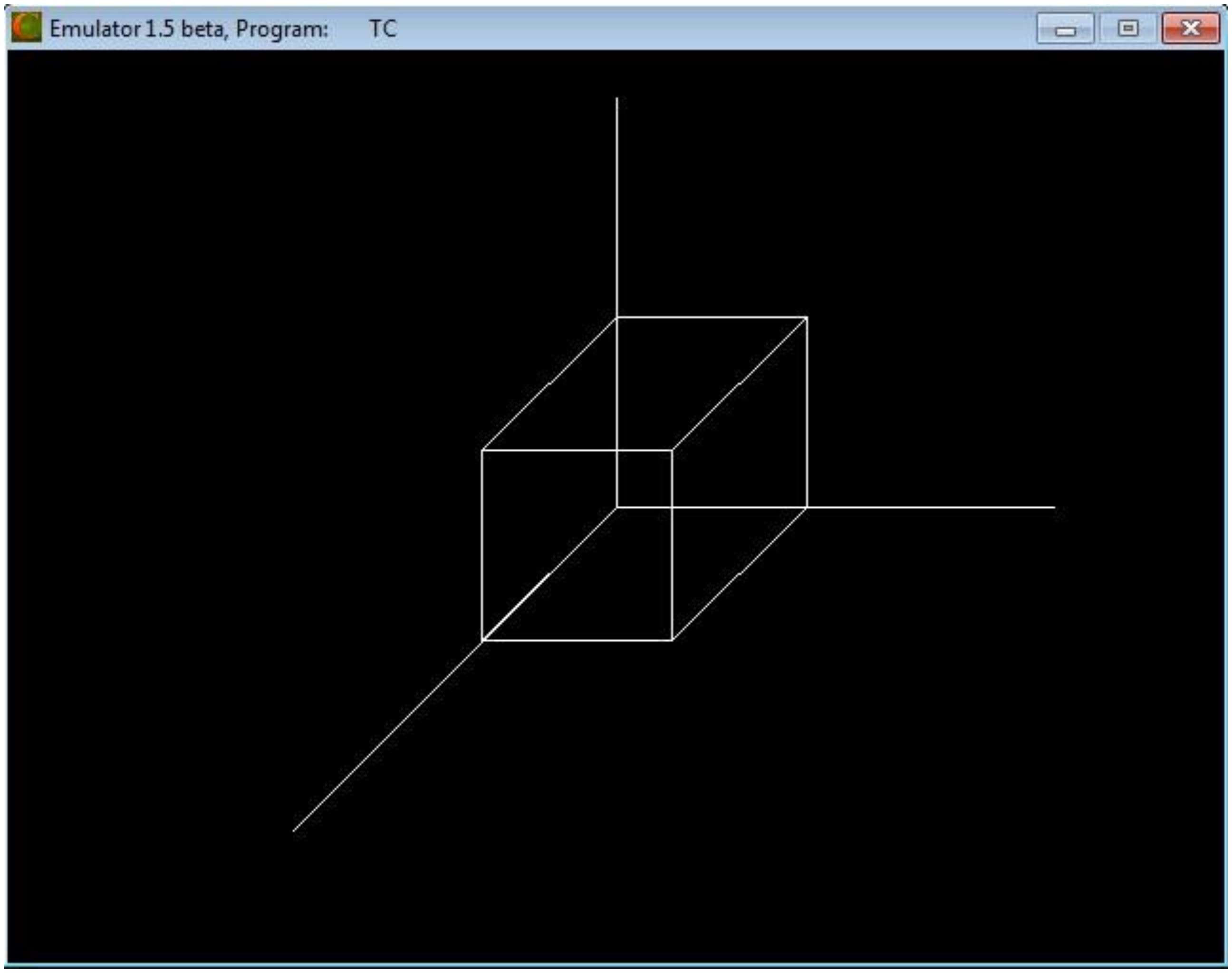
Case 3:

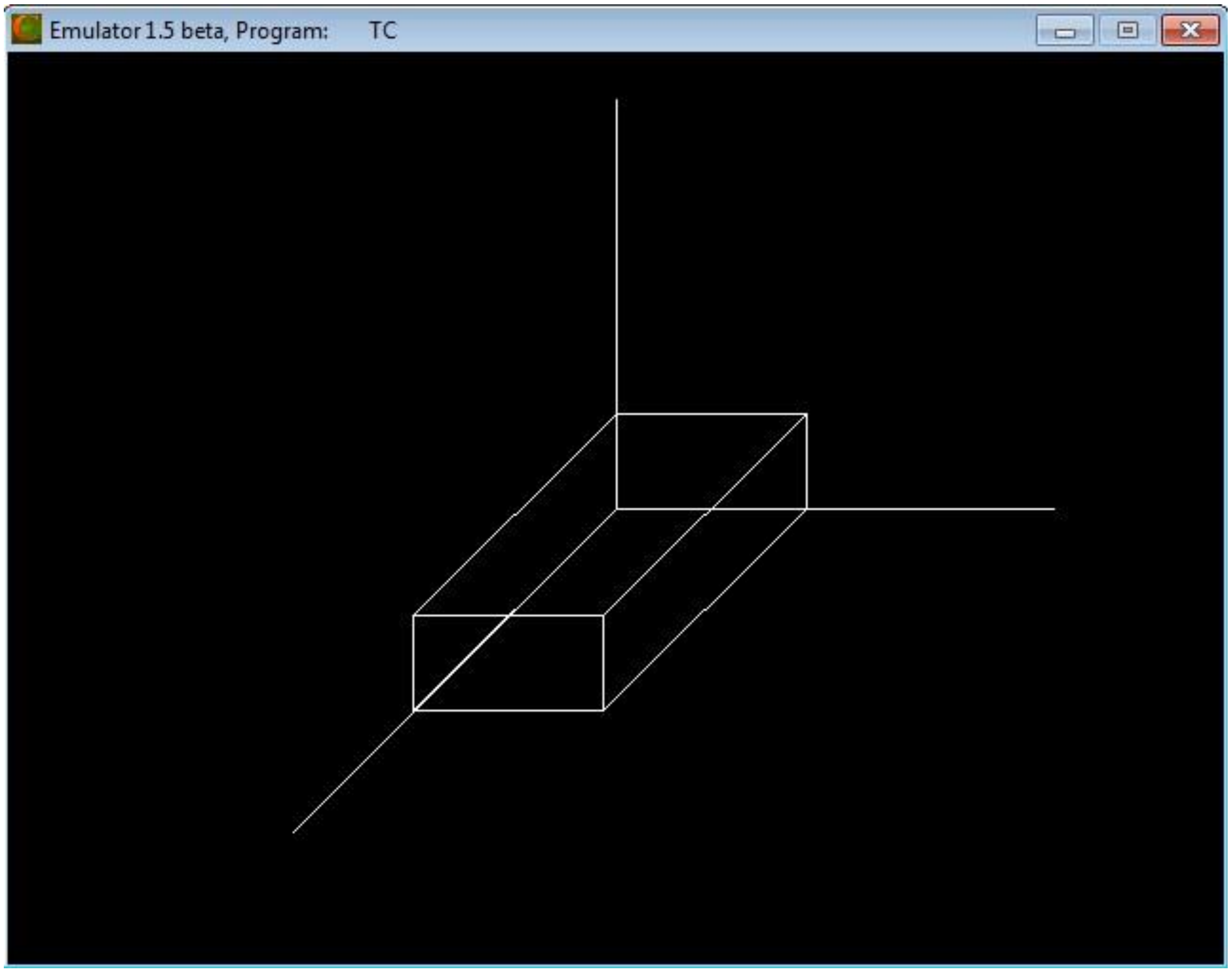
Choose any one 3D Transformation:

1. Translation
2. Rotation
3. Scaling

Enter your choice: 3

Enter the scaling factors (sx,sy,sz) : 1 0.5 1.5





RESULT:

Thus the implementation of the 3D Transformations (translation, rotation and scaling) has been implemented and the output was verified.

VIVA QUESTIONS

1. **What are the various representation schemes used in three dimensional objects?**
   Boundary representation (B-res) – describe the 3 dimensional object as a set of surfaces that separate the object interior from the environment.
   Space-portioning representation – describe interior properties, by partitioning the spatial region Containing an object into a set of small, no overlapping, contiguous solids.

2. **What is Polygon mesh?**
   Polygon mesh is a method to represent the polygon, when the object surfaces are tiled, it is more convenient to specify the surface facets with a mesh function. The various meshes are
   Triangle strip – (n-2) connected triangles
   Quadrilateral mesh – generates (n-1)(m-1) Quadrilateral

3. **What is Bezier Basis Function?**
   Bezier Basis functions are a set of polynomials, which can be used instead of the primitive polynomial basis, and have some useful properties for interactive curve design.

4. **What is the use of control points?**
   Spline curve can be specified by giving a set of coordinate positions called control points, which indicates the general shape of the curve, can specify spline curve.

5. **What is Transformation?**
   Transformation is the process of introducing changes in the shape size and orientation of the object using scaling rotation reflection shearing & translation etc.

6. **What is translation?**
   Translation is the process of changing the position of an object in a straight-line path from one coordinate location to another. Every point (x, y) in the object must undergo a displacement to (x´,y´). the transformation is:
   x´ = x + tx
   y´ = y+ty

7. **What is rotation?**
   A 2-D rotation is done by repositioning the coordinates along a circular path, in X = rcos (q + f) and Y = r sin (q + f).

8. **What is scaling?**
   The scaling transformations changes the shape of an object and can be carried out by multiplying each vertex (x,y) by scaling factor Sx,Sy where Sx is the scaling factor of x and Sy is the scaling factor of y.

9. **What is shearing?**
   The shearing transformation actually slants the object along the X direction or the Y direction as required.ie; this transformation slants the shape of an object along a required plane.

10. **What is a Blobby object?**
    Some objects do not maintain a fixed shape, but change their surface characteristics in certain motions or when in proximity to other objects. That is known as blobby objects. Example – molecular structures, water droplets.

APPLICATIONS

- Video game
- Image Processing*:* Medical field
- Entertainment: It mostly used for making videos, motion pictures, cartoon animation films.
- Computer Aided Design (CAD)

EX.NO: 6 3D PROJECTIONS - PARALLEL AND PERSPECTIVE

DATE:

AIM:

To write a program to implement the following 3D projections:

a. Parallel Projection b. Perspective Projection

ALGORITHM:

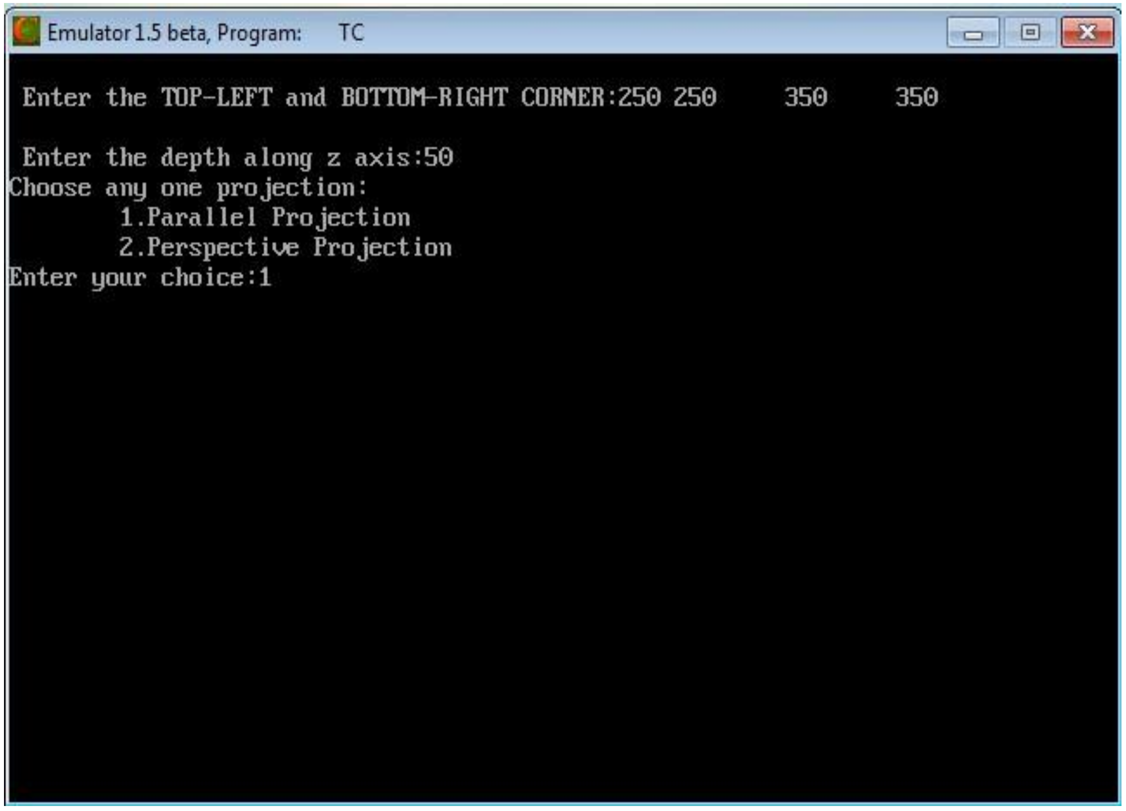
1. Get the coordinate to draw the cube.
2. Print the menu : 1. Parallel Projection 2. Perspective Projection
3. If user choose Parallel Projection then find the coordinates of following views and draw them:
   a) Side View
   b) Front View
   c) Top View

4. If user choose Perspective Projection then simply draw the cube using bar3d( ) function.

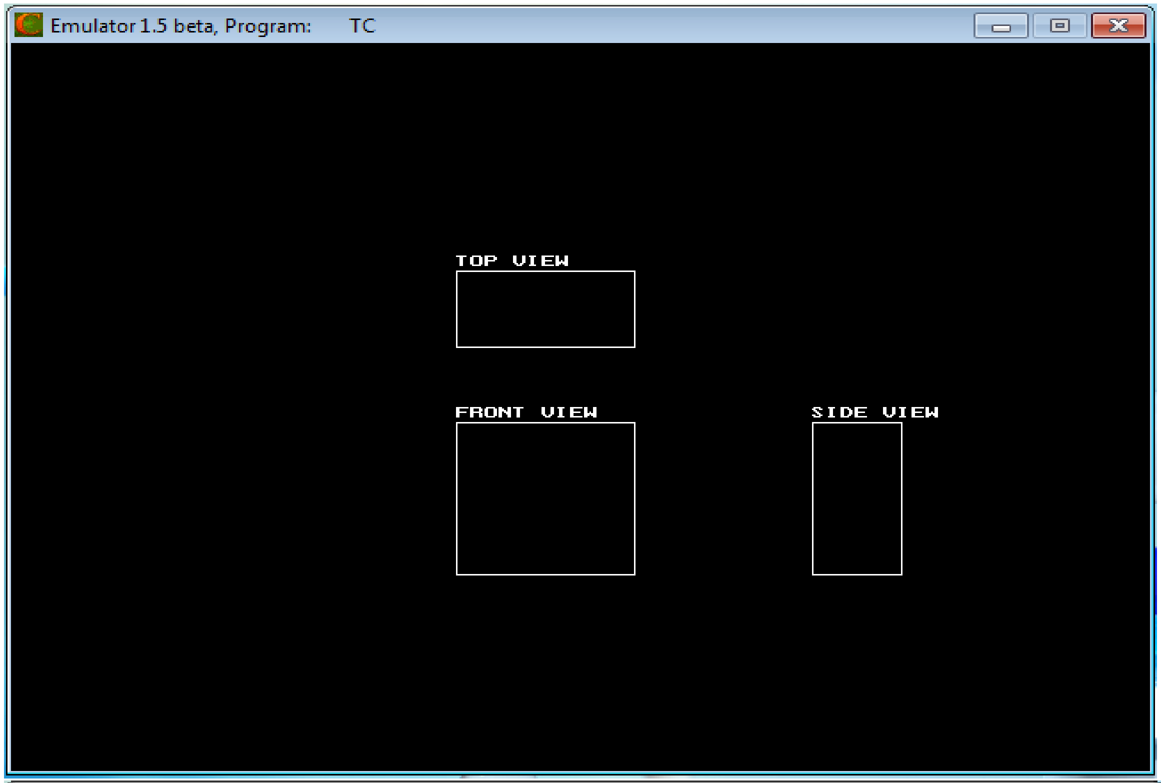
PROGRAM:

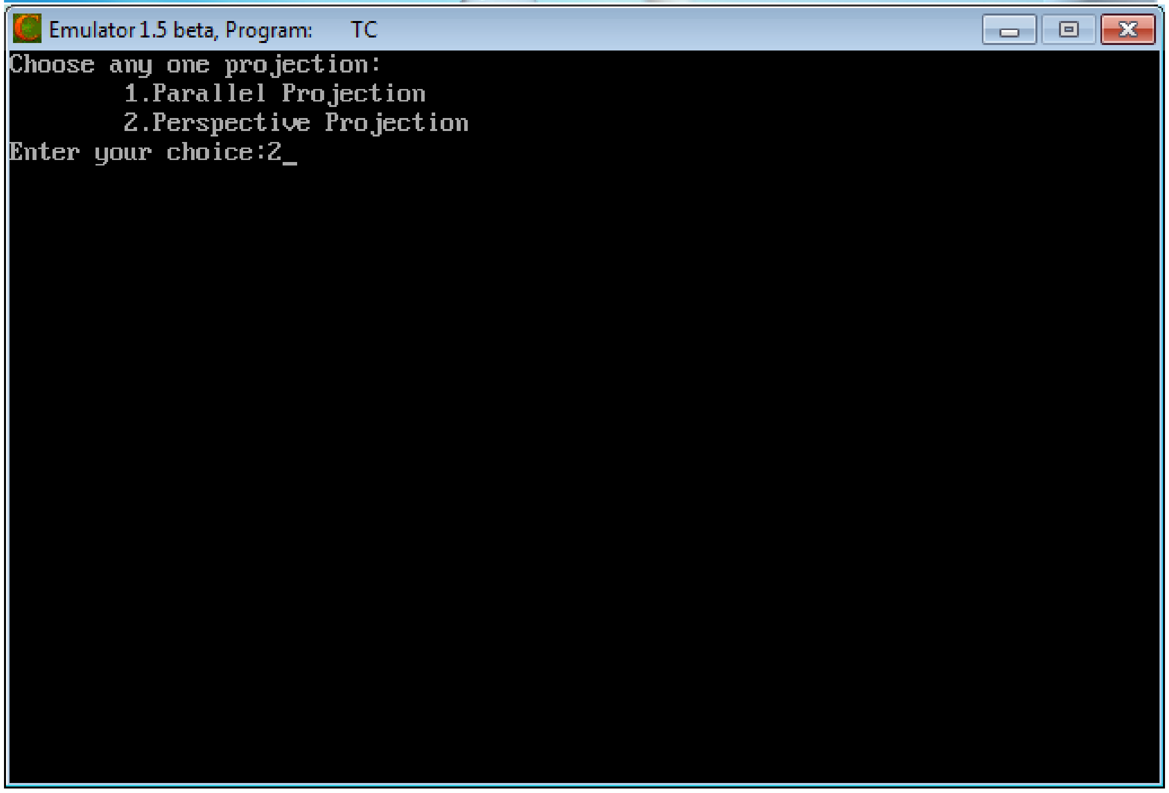
```
#include<graphics.h>
#include<iostream.h>
#include<conio.h>
int gd,gm,x1,y1,x2,y2,dep,ch;
void main()
{
        cout<<"\n Enter the TOP-LEFT and BOTTOM-RIGHT CORNER:";
        cin>>x1>>y1>>x2>>y2;
        cout<<"\n Enter the depth along z axis:";
        cin>>dep;
        do
        {
                cout<<"Choose any one projection:\n\t1.Parallel Projection\n\t2.Perspective
                Projection\nEnter your choice:";
                cin>>ch;
                initgraph(&gd,&gm,"C:\\TC\\BGI");
                switch(ch)
                {
                        case 1:
                                rectangle(x2+100,y1,x2+100+dep,y2);
                                outtextxy(x2+100,y1-10,"SIDE VIEW");
                                rectangle(x1,y1,x2,y2);
                                outtextxy(x1,y1-10,"FRONT VIEW");
                                rectangle(x1,y1-(y2-y1),x2,x1+dep-(y2-y1));
                        outtextxy(x1,y1-(y2-y1)-10,"TOP VIEW");
                        getch();
                        closegraph();
                        break;
                case 2:
                        bar3d(x1,y1,x2,y2,dep,1);
                        getch();
                        closegraph();
                        break;
        }
    }while(ch<3);
}
```

OUTPUT:

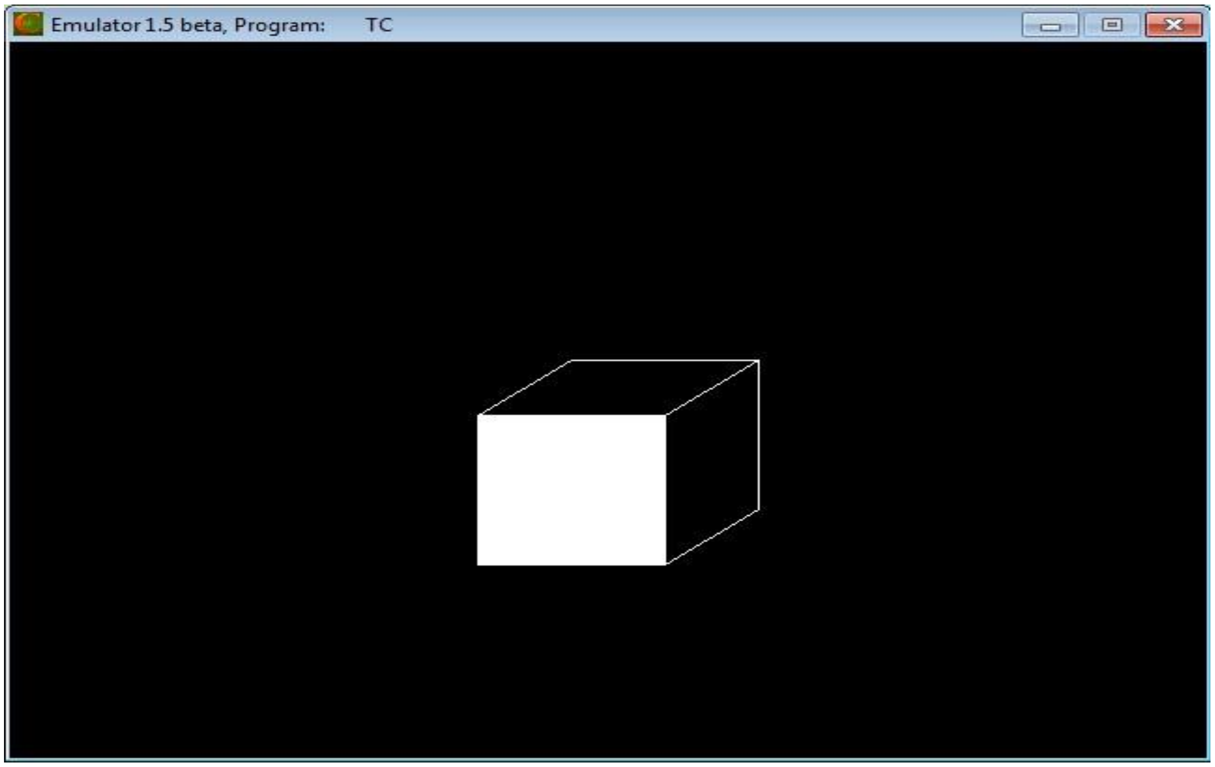


```
 Enter the TOP-LEFT and BOTTOM-RIGHT CORNER:250 250     350     350

 Enter the depth along z axis:50
Choose any one projection:
        1.Parallel Projection
        2.Perspective Projection
Enter your choice:1
```





```
Choose any one projection:
        1.Parallel Projection
        2.Perspective Projection
Enter your choice:2_
```



RESULT:

Thus the implementations of 3D projections (Parallel and Perspective) has been created and the output was verified.

VIVA QUESTIONS

1. **What are the two types of projections?**
   - Parallel projection
   - Perspective projection

2. **Define projections?**
   The process of displaying 3D into a 2D display unit is known as projection. The projection transforms 3D objects into a 2D projection plane. The process of converting the description of objects from world coordinates to viewing coordinates is known as projection

3. **What are the two types of parallel projection?**
   The parallel projections are basically categorized into two types, depending on the relation between the direction of projection and the normal to the view plane. They are orthographic parallel projection and oblique projection

4. **What is Projection reference point?**
   In Perspective projection, the lines of projection are not parallel. Instead, they all converge at a single point called Projection reference point.

5. **What is orthographic parallel projection?**
   When the direction of the projection is normal (perpendicular) to the view plane then the projection is known as orthographic parallel projection

6. **What is orthographic oblique projection?**
   When the direction of the projection is not normal (not perpendicular) to the view plane then the projection is known as oblique projection.

7. **What is an axonometric orthographic projection?**
   The orthographic projection can display more than one face of an object. Such an orthographic projection is called axonometric orthographic projection.

8. **What is cavalier projection?**
   The cavalier projection is one type of oblique projection, in which the direction of projection makes a 45-degree angle with the view plane.

9. **What is cabinet projection?**
   The cabinet projection is one type of oblique projection, in which the direction of projection makes a n angle of arctan (2)=63.4- with the view plane.

10. **What is vanishing point?**
    The perspective projections of any set of parallel lines that are not parallel to the projection plane converge to appoint known as vanishing point.

APPLICATIONS

- Flight simulator program.
- Special Effects for cinema.
- Ultrasonic medical scanners
- Graphic Presentation

EX.NO: 7 **CREATING 3D SCENES**

DATE:

AIM:

To create the 3D Scene in C++.

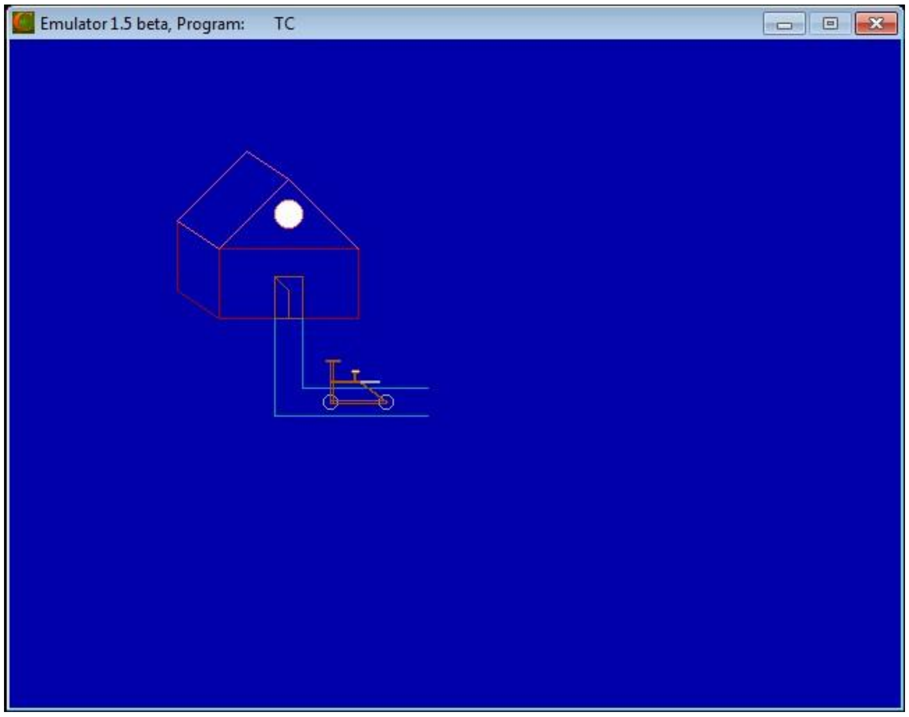
ALGORITHM:

1. Set the background color.
2. Find the coordinates to draw a 3D scene.
3. Plot that coordinates using pre-defined functions like line( ), rectangle( ), fillellipse( ).
4. Set the color of the objects by setcolor( ).

PROGRAM:

```
#include<graphics.h>
#include<iostream.h>
#include<conio.h>
intgd,gm;
void main()
{
        initgraph(&gd,&gm,"C:\\TC\\BGI");
        setbkcolor(BLUE);
        setcolor(RED);
        rectangle(150,150,250,200);
        line(150,200,120,180);
        line(120,180,120,130);
        setcolor(LIGHTRED);
        line(120,130,150,150);
        line(150,150,200,100);
        line(200,100,250,150);
        fillellipse(200,125,10,10);
        line(120,130,170,80);
        line(170,80,200,100);
        setcolor(BROWN);
        rectangle(190,170,210,200);
        line(190,170,200,180);
        line(200,180,200,200);
        setcolor(CYAN);
        line(190,200,190,270);
        line(210,200,210,250);
        line(190,270,300,270);
        line(210,250,300,250);
        setcolor(LIGHTGRAY);
        circle(230,260,5);
        circle(270,260,5);
        setcolor(BROWN);
        line(230,259,270,259);
        line(230,261,270,261);
        line(230,230,230,261);
        line(232,230,232,260);
        line(227,230,237,230);
        line(227,231,237,231);
        line(270,260,250,245);
        line(271,260,251,245);
        line(250,245,230,245);
        line(250,246,230,246);
        line(248,245,248,240);
        line(247,245,247,240);
        fillellipse(248,238,3,1);
        setcolor(LIGHTGRAY);
        line(252,245,265,245);
        line(252,246,265,246);
        getch();
}
```

OUTPUT:



RESULT:

Thus the 3D scene has been created and the output was verified.

VIVA QUESTIONS

1. **What are subtractive colors?**
   RGB model is an additive system, the Cyan-Magenta-Yellow (CMY) model is a subtractive color model. In a subtractive model, the more that an element is added, the more that it subtracts from white. So, if none of these are present the result is white, and when all are fully present the result is black.

2. **Define primary colors.**
   The two or three colors used to produce other colors in a color model are referred to as primary colors.

3. **Define YIQ color model**
   In the YIQ color model, luminance (brightness) information in contained in the Y parameter, chromaticity information (hue and purity) is contained into the I and Q parameters.

4. **What is a color model?**
   A color model is a method for explaining the properties or behavior of color within some particular context.
   Example: XYZ model, RGB model.

5. **What is texture?**
   The realism of an image is greatly enhanced by adding surface texture to various faces of a mesh object. The basic technique begins with some texture function, texture(s,t) in texture space , which has two parameters s and t.

6. **Define rendering.**
   Rendering is the process of generating an image from a model (or models in what collectively could be called a *scene*file), by means of computer programs. Also, the results of such a model can be called a rendering.

7. **Differentiate flat and smooth shading.**
   The main distinction is between a shading method that accentuates the individual polygons (flat shading) and a method that blends the faces to de-emphasize the edges between them (smooth shading).

8. **Define shading**
   Shading is a process used in drawing for depicting levels of darkness on paper by applying media more densely or with a darker shade for darker areas, and less densely or with a lighter shade for lighter areas.

9. **What are two methods for computing shadows?**
   - Shadows as Texture.
   - Creating shadows with the use of a shadow buffer.

10. **What are the two common sources of textures?**

- Bitmap Textures.
- Procedural Textures.

APPLICATIONS

- Making 3D movies.
- Special Effects for cinema.
- Computer Arts
- Scientific and Business Visualization
- Video Game.

EX.NO: 8 **IMAGE EDITING AND MANIPULATION**

DATE:

AIM:

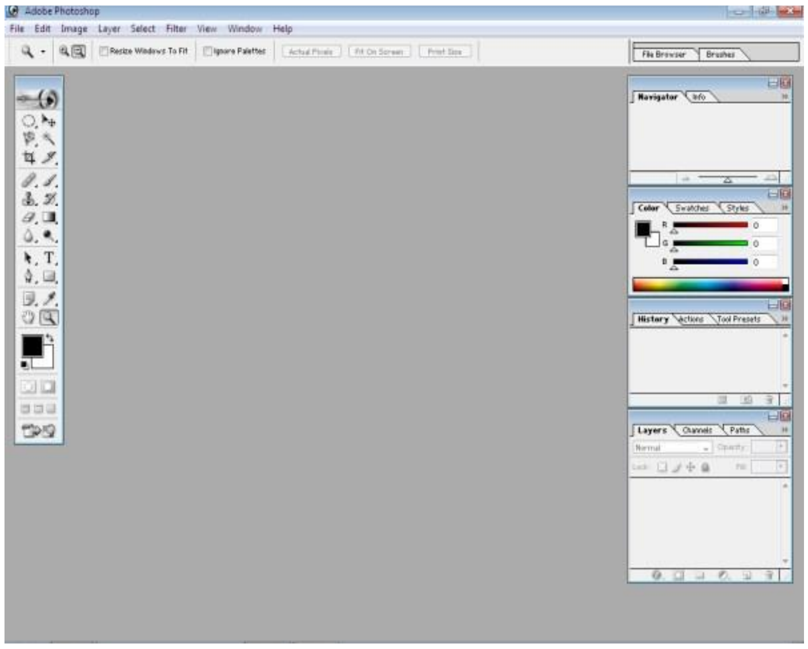
To perform the following operations,

(i) Basic Operations on image using Adobe Photoshop 7.0
(ii) Creating gif animated images in PhotoScape
(iii) To optimize the image in Adobe Photoshop 7.0

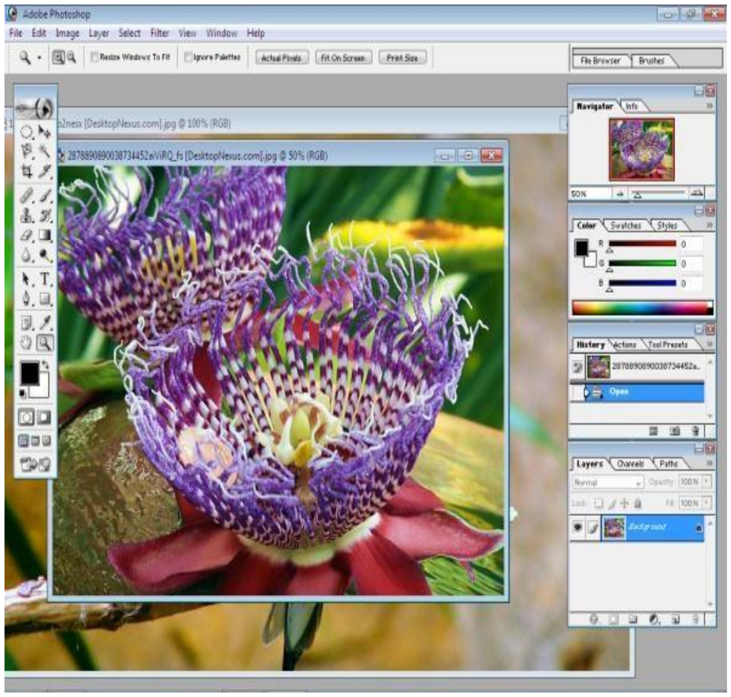
PROCEDURE:

1. **To perform the basic operations on image using Adobe Photoshop 7.0**

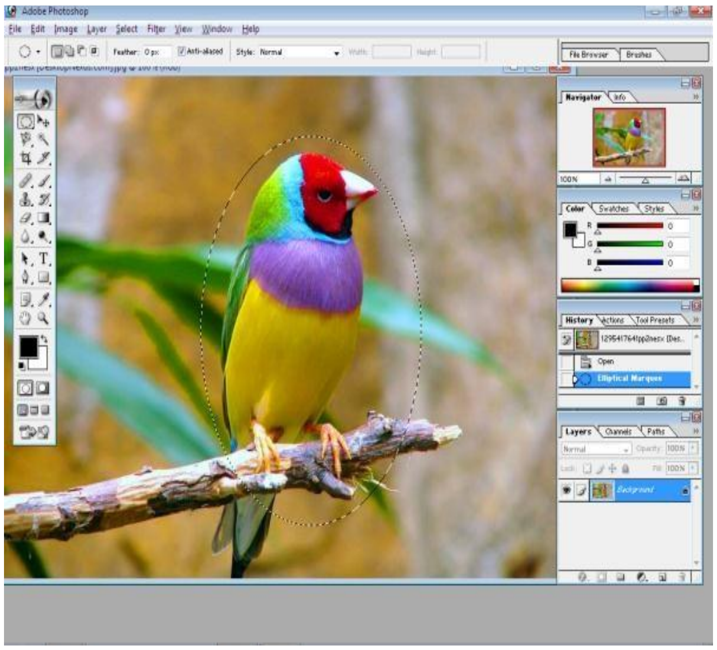
   i. Open the Adobe Photoshop 7.0.



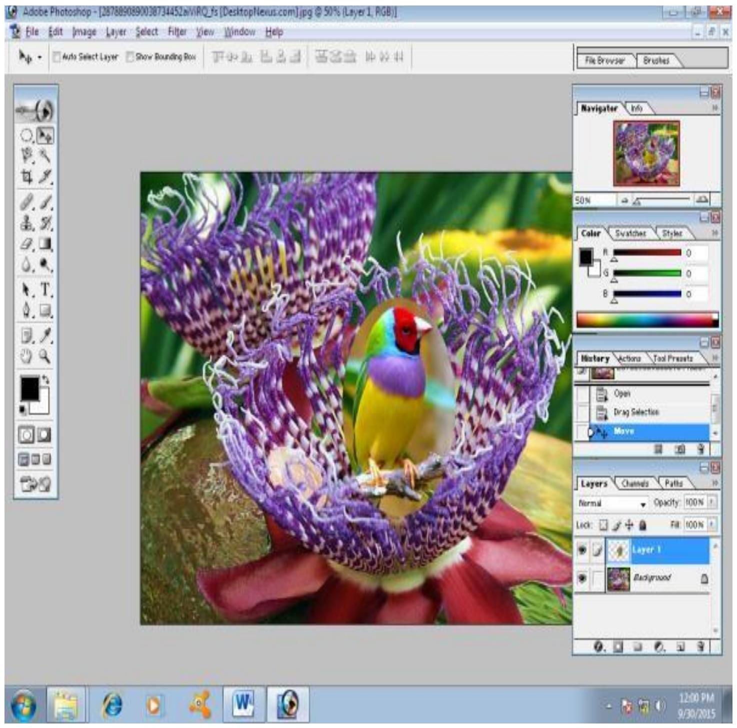
   ii. Open the images.



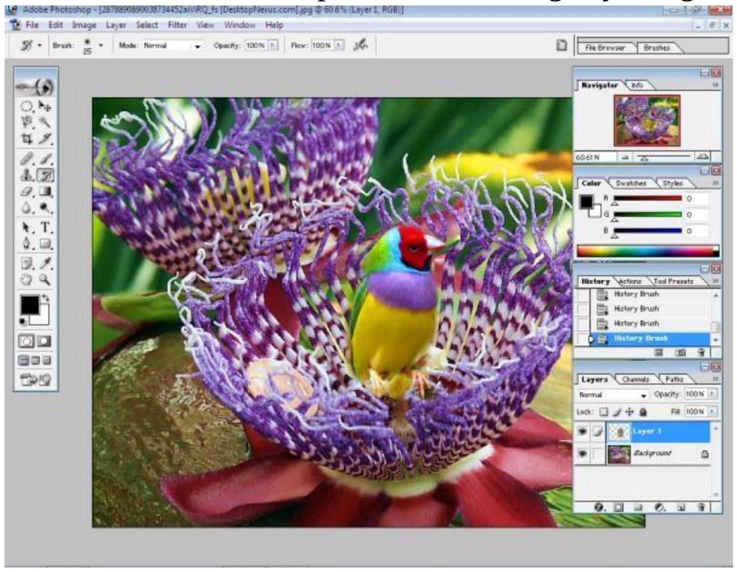
iii. Select the particular portion of the image by Elliptical Marquee Tool.



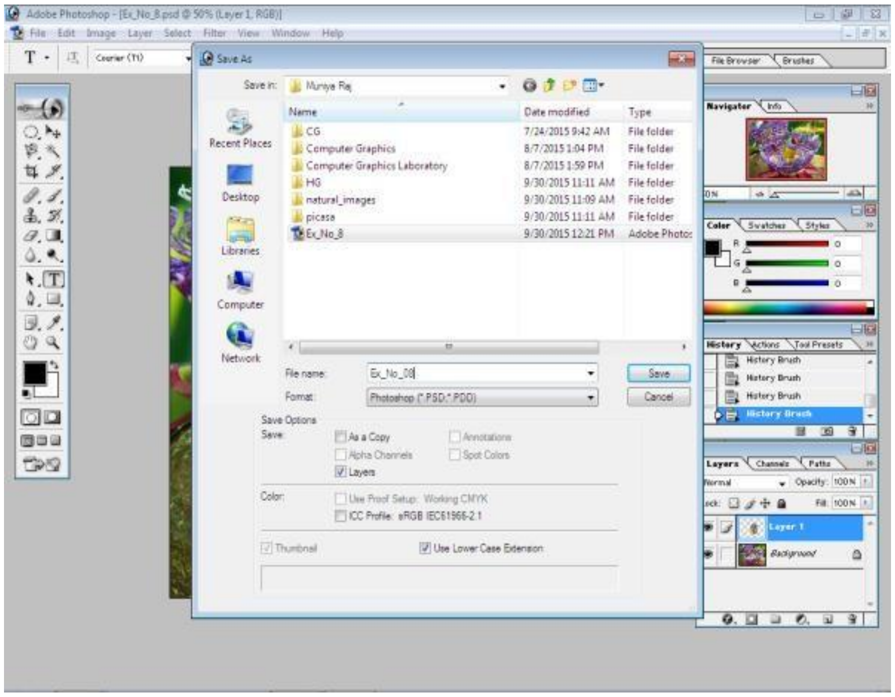
iv. Move that selected portion to another image by using Move Tool.



v. Eliminate the unwanted portion of the image by using History Brush Tool.



vi. Save the modified image in jpg format


Adobe Photoshop - [Ex_No_8.psd @ 50% (Layer 1, RGB)]
File Edit Image Layer Select Filter View Window Help
Save As
Recent Places
Desktop
Libraries
Computer
Network
Save
Cancel

Save in: Muniya Raj

Name	Date modified	Type
CG	7/24/2015 9:42 AM	File folder
Computer Graphics	8/7/2015 1:04 PM	File folder
Computer Graphics Laboratory	8/7/2015 1:59 PM	File folder
HG	9/30/2015 11:11 AM	File folder
natural_images	9/30/2015 11:09 AM	File folder
picasa	9/30/2015 11:11 AM	File folder
Ex_No_8	9/30/2015 12:21 PM	Adobe Photos

File name: Ex_No_08

Format: Photoshop (*.PSD;*.PDD)

Save Options

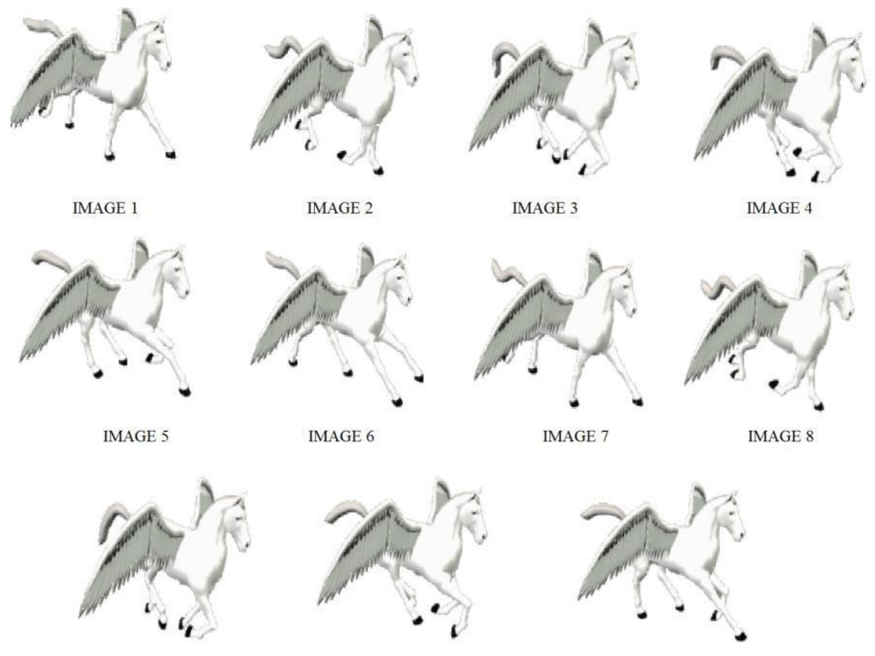
Save: ☐ As a Copy ☐ Annotations ☐ Alpha Channels ☐ Spot Colors ☑ Layers

Color: ☐ Use Proof Setup: Working CMYK ☐ ICC Profile: sRGB IEC61966-2.1

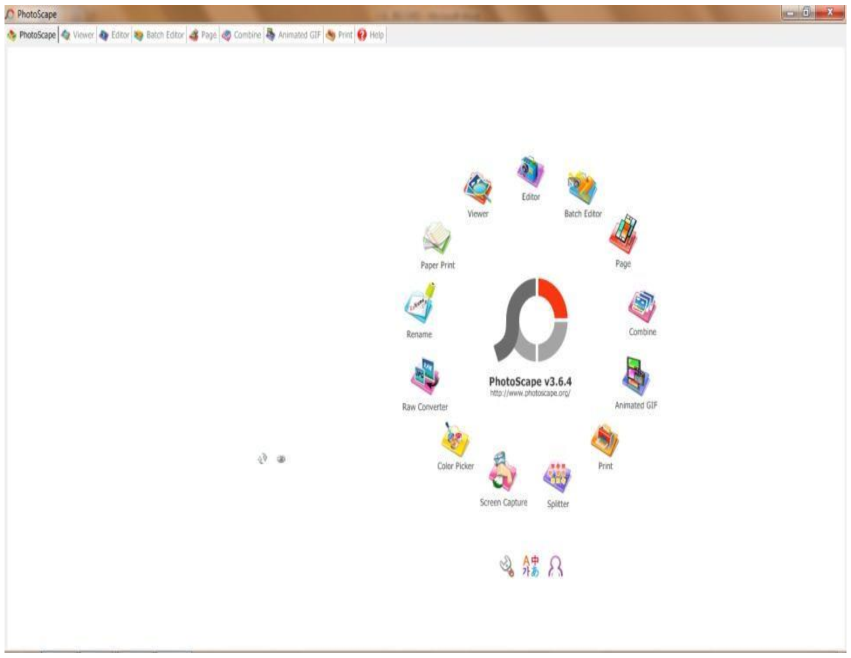
☑ Thumbnail ☑ Use Lower Case Extension

2. **To Create the gif animated images in PhotoScape**

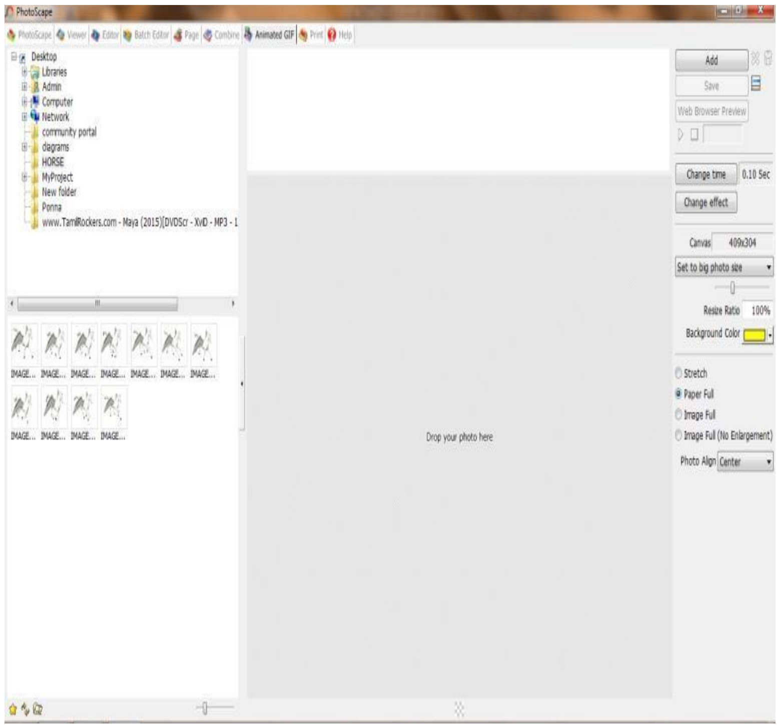
    i. Draw the images in different scenes to animate. Save all the images in jpg format.



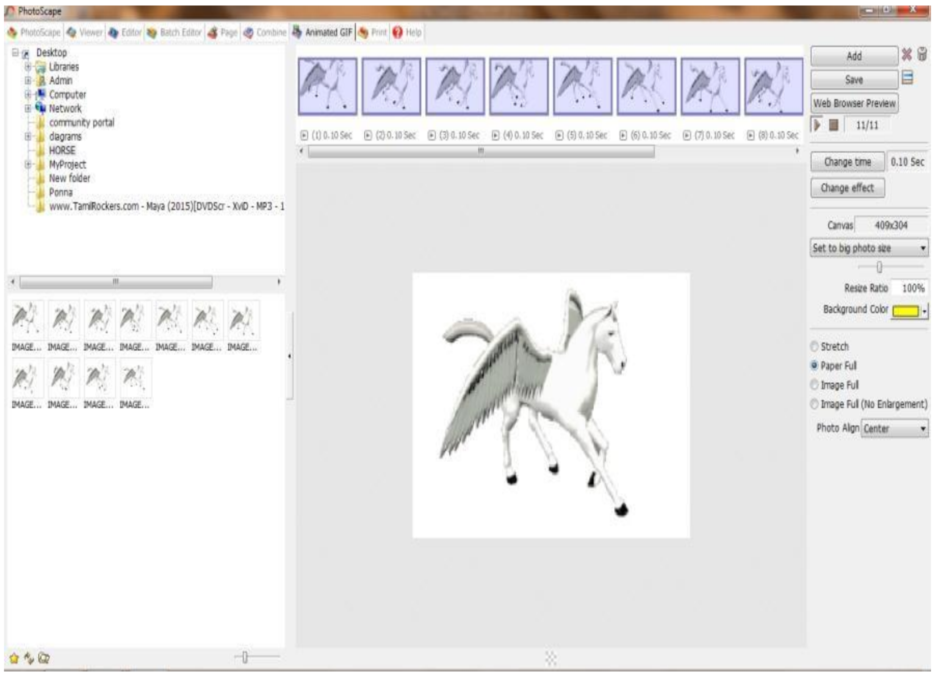
    ii. Open PhotoScape.



iii. Choose Animated GIF from the menu.



iv. Drop all the images in center bottom portion of the window.


PhotoScape
PhotoScape Viewer Editor Batch Editor Page Combine Animated GIF Print Help
Desktop
Libraries
Admin
Computer
Network
community portal
diagrams
HORSE
MyProject
New folder
Ponna
www.TamilRockers.com - Maya (2015)[DVDScr - XviD - MP3 - 1
Add
Save
Web Browser Preview
Change effect
Stretch
Image Full
Image Full (No Enlargement)

(1) 0.10 Sec (2) 0.10 Sec (3) 0.10 Sec (4) 0.10 Sec (5) 0.10 Sec (6) 0.10 Sec (7) 0.10 Sec (8) 0.10 Sec

11/11

Change time 0.10 Sec

Canvas 409x304

Set to big photo size

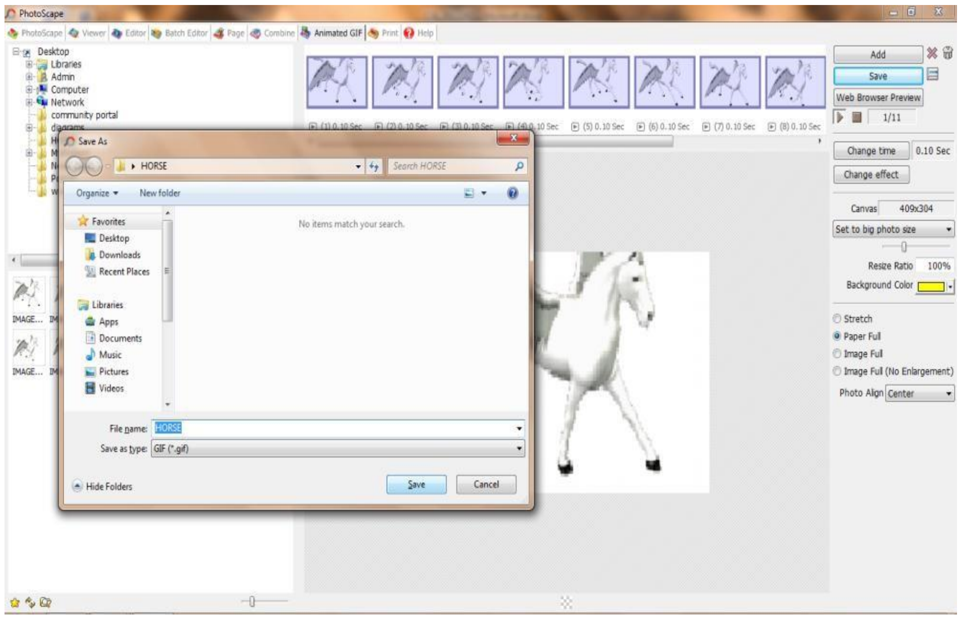
Resize Ratio 100%

Background Color 🟨

Paper Full

Photo Align Center

v. Save the animated file in gif format.



HORSE

File name: HORSE

Save as type: GIF (*.gif)

1/11

Change time 0.10 Sec

Canvas 409x304

Set to big photo size

Resize Ratio 100%

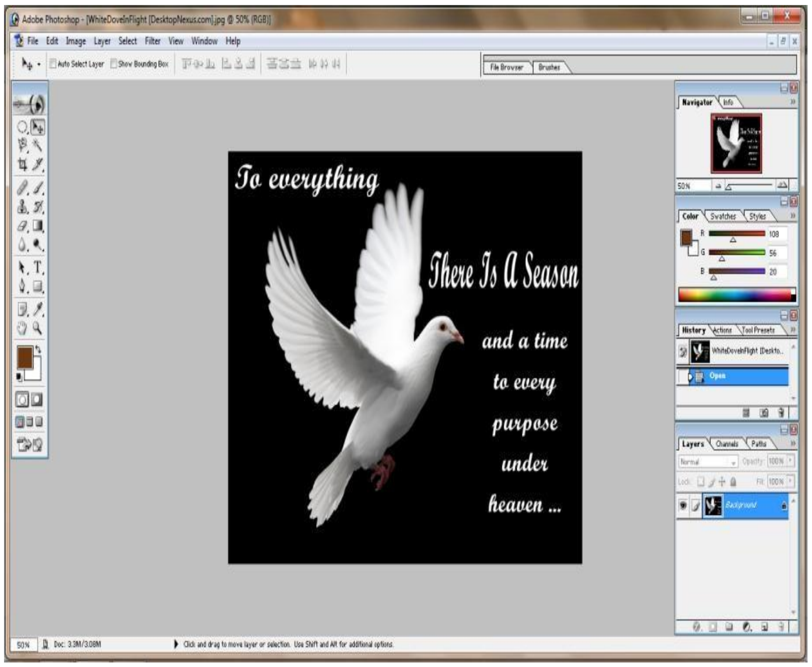
Background Color 🟨

Paper Full

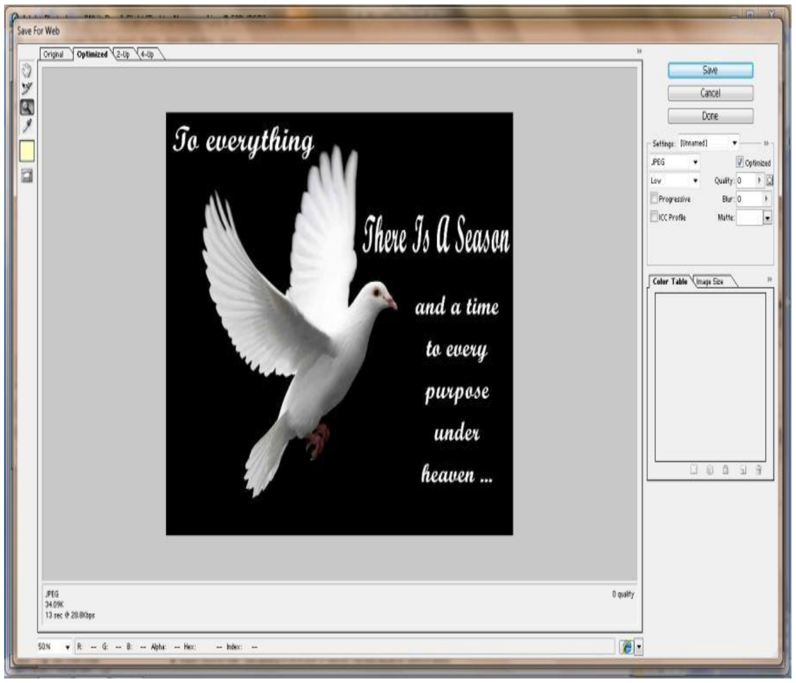
Photo Align Center

3. **To optimize the image in Adobe Photoshop 7.0**

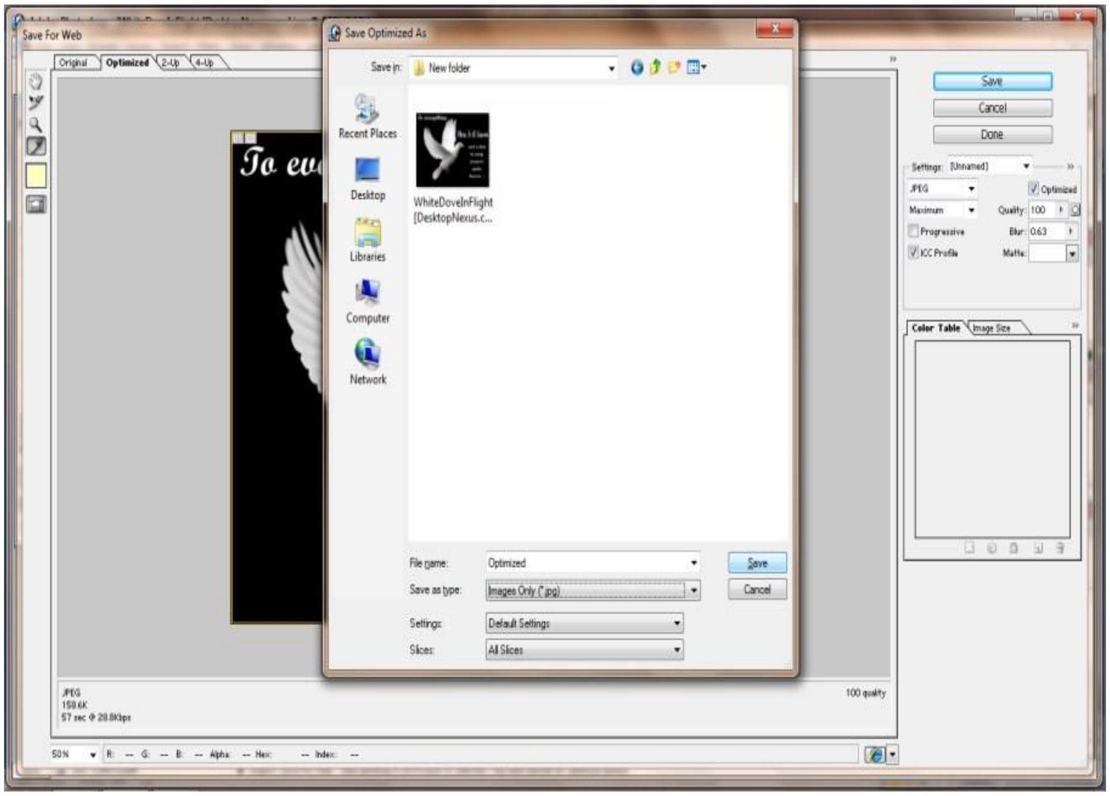
i.Open the image in Adobe Photoshop 7.0 which size is 85.2 KB


Adobe Photoshop - [WhiteDoveInFlight [DesktopNexus.com].jpg @ 50% (RGB)]
File Edit Image Layer Select Filter View Window Help
To everything
There Is A Season
and a time
to every
purpose
under
heaven ...

ii. File , Save for Web (or press Alt + Shift + Ctrl + S)



iii.Adjust the Quality and Blur of the image. Finally the optimized image in jpg format which size is 40.3 KB.



File name: Optimized
Save as type: Images Only (*.jpg)
Settings: Default Settings
Slices: All Slices

Settings: [Unnamed]
JPEG
[✓] Optimized
Maximum
Quality: 100
[ ] Progressive
Blur: 0.63
[✓] ICC Profile
Matte:

JPEG
158.6K
57 sec @ 28.8Kbps

50%

RESULT:

Thus the operations (Basic Operations on image, creating gif animated image, optimize the image) has been performed and the output was verified.

VIVA QUESTIONS

1. **Define Image editing?**
   Image editing encompasses the processes of altering images, whether they are digital photographs, traditional photochemical photographs, or illustrations.

2. **Define frame.**
   One of the shape photographs that a film or video is made of is known as frame.

3. **Define Raster images.**
   Raster images are stored in a computer in the form of a grid of picture elements, or pixels. These pixels contain the image's color and brightness information.

4. **How to remove the unwanted elements in image?**
   Most image editors can be used to remove unwanted branches, etc., using a "clone" tool. Removing these distracting elements draws focus to the subject, improving overall composition.

5. **What is Fractals?**
   A Fractal is an object whose shape is irregular at all scales.

6. **What is graftals?**
   Graftals are applicable to represent realistic rendering plants and trees. A tree is represented by a String of symbols 0, 1.

7. **Give some examples for computer graphics standards**.
   - CORE – The Core graphics standard
   - GKS -- The Graphics Kernel system
   - PHIGS – The Programmers Hierarchical Interactive Graphics System.
   - GSX – The Graphics system extension
   - NAPLPS – The North American presentation level protocol syntax.

8. **What is hue?**
   The perceived light has a dominant frequency (or dominant wavelength). The dominant frequency is also called as hue or simply as color.

APPLICATIONS

- Making 3D movies.
- Making animated movies and images.
- Special Effects for cinema: To improve the picture quality
- Education field.

EX.NO: 9 **2D ANIMATION**

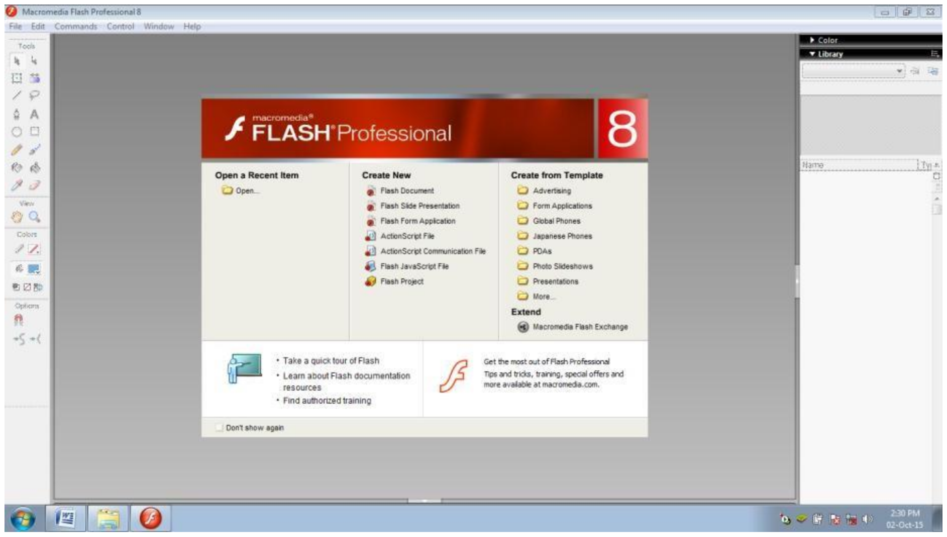
DATE:

AIM:

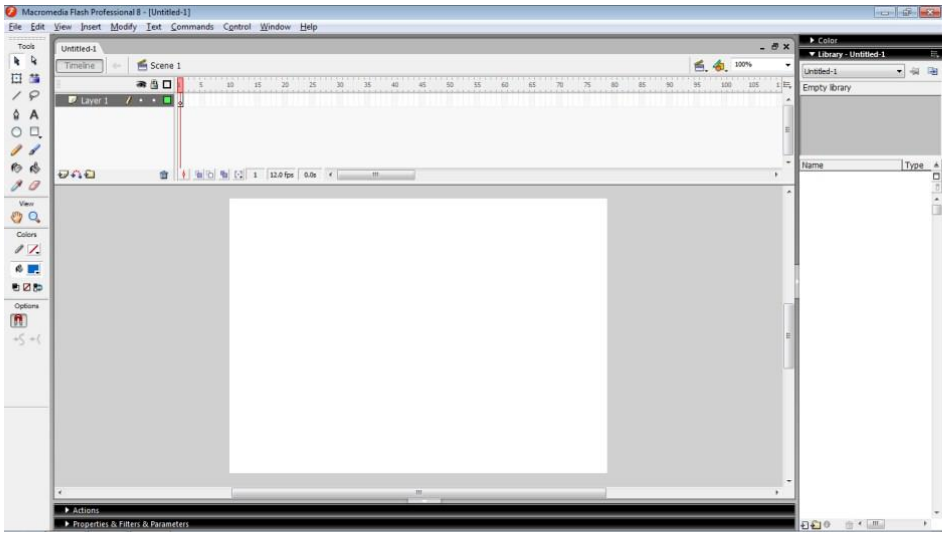
To create interactive animation using Macromedia Flash 8.

PROCEDURE:

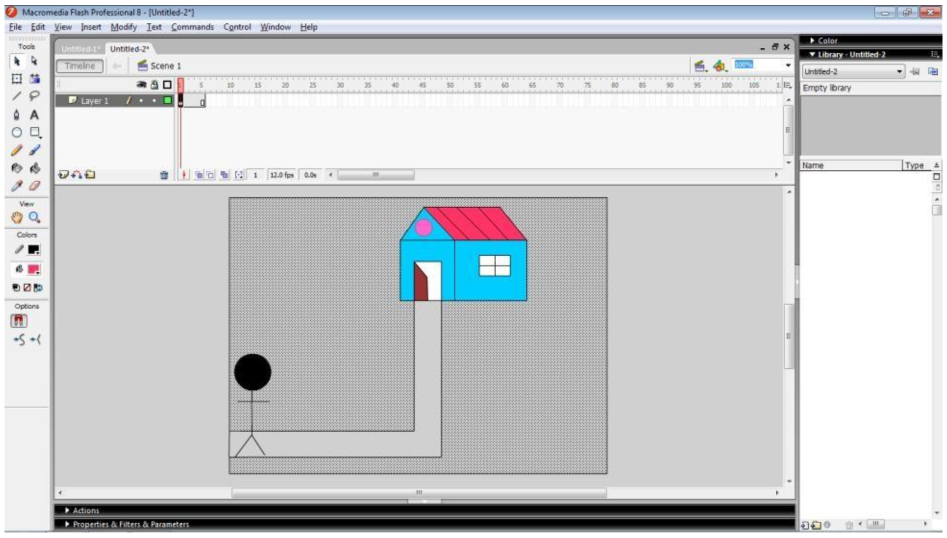
1. Open Macromedia Flash 8.


Macromedia Flash Professional 8
File Edit Commands Control Window Help
macromedia FLASH Professional 8
Open a Recent Item
Open...
Create New
Flash Document
Flash Slide Presentation
Flash Form Application
ActionScript File
ActionScript Communication File
Flash JavaScript File
Flash Project
Create from Template
Advertising
Form Applications
Global Phones
Japanese Phones
PDAs
Photo Slideshows
Presentations
More...
Extend
Macromedia Flash Exchange
Take a quick tour of Flash
Learn about Flash documentation resources
Find authorized training
Get the most out of Flash Professional
Tips and tricks, training, special offers and more available at macromedia.com.
Don't show again
Color
Library
2:30 PM
02-Oct-15

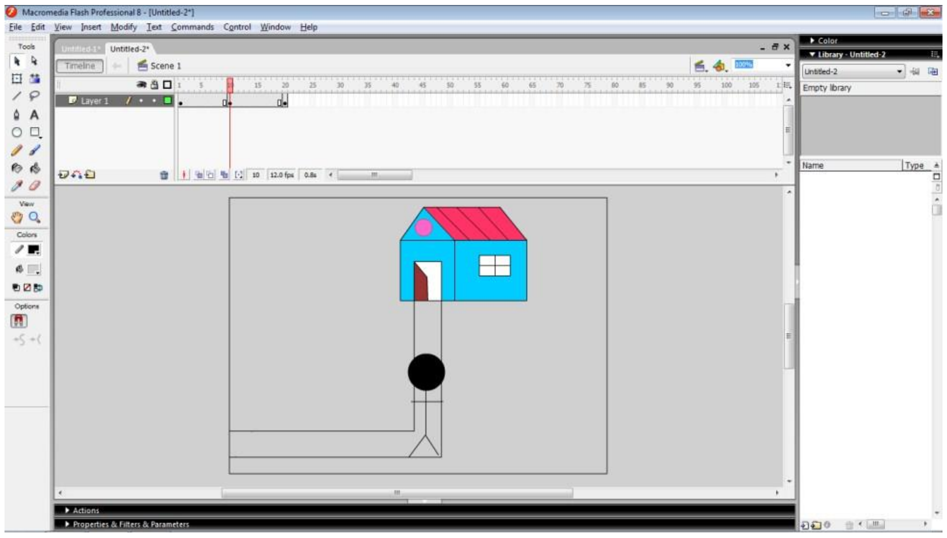
2. Create New • Flash Document.


Macromedia Flash Professional 8 - [Untitled-1]
File Edit View Insert Modify Text Commands Control Window Help
Untitled-1
Timeline
Scene 1
Layer 1
12.0 fps
0.0s
100%
Color
Library - Untitled-1
Empty library
Name
Type
Actions
Properties & Filters & Parameters

3. Draw the 2D image (scene) to animate.


Macromedia Flash Professional 8 - [Untitled-2*]
File Edit View Insert Modify Text Commands Control Window Help
Timeline
Scene 1
Layer 1
Library - Untitled-2
Empty library
Actions
Properties & Filters & Parameters

4. Insert a new frame and change the scene for new frame.


Macromedia Flash Professional 8 - [Untitled-2*]
File Edit View Insert Modify Text Commands Control Window Help
Timeline
Scene 1
Layer 1
Library - Untitled-2
Empty library
Actions
Properties & Filters & Parameters

5. Similarly create many frames.

6. Control ・ Play (or press Enter) to run the Animation.

RESULT:

Thus the interactive 2D Animation has been created and the output was verified.

VIVA QUESTIONS

1. **Define frame.**
   One of the shape photographs that a film or video is made of is known as frame.

2. **What is tweening?**
   It is the process, which is applicable to animation objects defined by a sequence of points, and that change shape from frame to frame.

3. **Define computer graphics animation.**
   Computer graphics animation is the use of computer graphics equipment where the graphics output presentation dynamically changes in real time. This is often also called real time animation. Mention the steps in animation sequence.

4. **What is the normal speed of a visual animation?**
   Visual animation requires a playback of at least 25 frames per second.

5. **What is computer graphics realism?**
   The creation of realistic picture in computer graphics is known as realism. It is important in fields such as simulation, design, entertainments, advertising, research, education, command, and control.

6. **What is graftals?**
   Graftals are applicable to represent realistic rendering plants and trees. A tree is represented by a String of symbols 0, 1.

7. **How realistic pictures are created in computer graphics?**
   To create a realistic picture, it must be process the scene or picture through viewing-coordinate transformations and projection that transform three-dimensional viewing coordinates onto two-dimensional device coordinates.

8. **What is geometric fractal?**
   A geometric fractal is a fractal that repeats self-similar patterns over all scales.

APPLICATIONS

- Making 2D animated movies and images.
- Web designing.
- Special Effects for cinema.
- Motion pictures, Cartoon animation films
- Education field.