**Program: 2D Transformation on polygon**

#include<stdio.h>

#include<math.h>

#include<graphics.h>

#include<dos.h>

void multiply(float B[3][3],float C[3][3],float D[3][3]);

void draw(float D[3][3]);

void print(float A[3][3]);

void main()

{

float x1,y1,x2,y2,x3,y3,obj[3][3],T[3][3],S[3][3],R[3][3],Sh[3][3],r[3][3];

int i,j,x,y,gd=DETECT,gm,ch;

initgraph(&gd,&gm,"C:\\TURBOC3\\BGI\\");

/\*printf("\n 1.Translation ");

printf("\n 2.Rotation wrt origin ");

printf("\n 3.Rotation wrt fix point ");

printf("\n 4.scaling wrt origin ");

printf("\n 5.scaling wrt fix point ");

printf("\n 6.reflection wrt x axis ");

printf("\n 7.Reflection wrt y axis ");

printf("\n 8.reflection wrt origin ");

printf("\n 9.shearing about x axis ");

printf("\n 10.shearing about y axis "); \*/

line(0,240,640,240);

line(320,0,320,480);

printf("\nEnter the coordinates of Polygon:\n");

printf("\nEnter first coordinates x1 and y1: \t");

scanf("%f%f", &x1, &y1);

printf("Enter second coordinates x2 and y2:\t");

scanf("%f%f", &x2, &y2);

printf("Enter third coordinates x3 and y3: \t");

scanf("%f%f", &x3, &y3);

line(320+x1,240-y1,320+x2,240-y2);

line(320+x2,240-y2,320+x3,240-y3);

line(320+x3,240-y3,320+x1,240-y1);

obj[0][0]=x1;

obj[0][1]=y1;

obj[1][0]=x2;

obj[1][1]=y2;

obj[2][0]=x3;

obj[2][1]=y3;

obj[0][2]=1;

obj[1][2]=1;

obj[2][2]=1;

printf("\n Object matrix is: \n");

print(obj);

printf("\nEnter your choice [1 - 10]: ");

scanf("%d",&ch);

switch(ch)

{

case 1:

{

float tx,ty,trans[3][3];

print(obj);

printf("\nEnter values for tx and ty: ");

scanf("%f%f",&tx,&ty);

for(i=0;i<3;i++){

for(j=0;j<3;j++){

if(i==j)

trans[i][j]=1;

else

trans[i][j]=0;

}

}

trans[2][0]=tx;

trans[2][1]=ty;

printf("\nTranslation matrix: \n");

print(trans);

multiply(obj,trans,T);

printf("\nOutput matrix: \n");

print(T);

draw(T);

}

break;

case 2:

{

float rotate[3][3],theta;

print(obj);

printf("\nEnter the angle in degree: ");

scanf("%f",&theta);

theta=theta\*(3.14/180);

rotate[0][0]=cos(theta);

rotate[1][1]=cos(theta);

rotate[0][2]=0;

rotate[1][2]=0;

rotate[2][0]=0;

rotate[2][1]=0;

rotate[2][2]=1;

rotate[0][1]=sin(theta);

rotate[1][0]=-sin(theta);

printf("\nRotation matrix: \n");

print(rotate);

multiply(obj,rotate,r);

printf("\nOuput matrix: \n");

print(r);

draw(r);

}

break;

case 3:

{

float rotate[3][3],theta;

print(obj);

printf("\nEnter the angle in degree: ");

scanf("%f",&theta);

printf("enter fixed point\n");

scanf("%d%d",&x,&y);

theta=theta\*(3.14/180);

rotate[0][0]=cos(theta);

rotate[1][1]=cos(theta);

rotate[0][2]=0;

rotate[1][2]=0;

rotate[2][0]=x\*(1-cos(theta))+y\*(sin(theta));

rotate[2][1]=y\*(1-cos(theta))-x\*(sin(theta));

rotate[2][2]=1;

rotate[0][1]=sin(theta);

rotate[1][0]=-sin(theta);

printf("\nRotation matrix: \n");

print(rotate);

multiply(obj,rotate,r);

printf("\nOuput matrix: \n");

print(r);

draw(r);

}

break;

case 4:

{

float scale[3][3],sx,sy;

print(obj);

printf("\nEnter values for sx and sy: ");

scanf("%f%f",&sx,&sy);

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

scale[i][j]=0;

}

}

scale[0][0]=sx;

scale[1][1]=sy;

scale[2][2]=1;

printf("\nScaling matrix: \n");

print(scale);

multiply(obj,scale,S);

printf("\nOutput matrix: \n");

print(S);

draw(S);

}

break;

case 5:

{

float scale[3][3],sx,sy;

print(obj);

printf("\nEnter values for sx and sy: ");

scanf("%f%f",&sx,&sy);

printf("enter fixed point\n");

scanf("%d%d",&x,&y);

for(i=0;i<3;i++)

{

for(j=0;j<3;j++);

scale[i][j]=0;

}

scale[2][0]=x\*(1-sx);

scale[2][1]=y\*(1-sy);

scale[0][0]=sx;

scale[1][1]=sy;

scale[2][2]=1;

printf("\nScaling matrix: \n");

print(scale);

multiply(obj,scale,S);

printf("\nOutput matrix: \n");

print(S);

draw(S);

}

break;

case 6:

{

float ref[3][3];

print(obj);

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

if(i==j)

ref[i][j]=1;

else

ref[i][j]=0;

}

}

ref[1][1]=-1;

printf("\n Reflection matrix: \n");

print(ref);

multiply(obj,ref,R);

printf("\nOutput matrix: \n");

print(R);

draw(R);

}

break;

case 7:

{

float ref[3][3];

print(obj);

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

if(i==j)

ref[i][j]=1;

else

ref[i][j]=0;

}

}

ref[0][0]=-1;

printf("\n Reflection matrix: \n");

print(ref);

multiply(obj,ref,R);

printf("\nOutput matrix: \n");

print(R);

draw(R);

}

break;

case 8:

{

float ref[3][3];

print(obj);

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

if(i==j)

ref[i][j]=1;

else

ref[i][j]=0;

}

}

ref[0][0]=ref[1][1]=-1;

printf("\nReflection matrix: \n");

print(ref);

multiply(obj,ref,R);

printf("\nOutput matrix: \n");

print(R);

draw(R);

}

break;

case 9:

{

float shear[3][3],shx;

printf("\nEnter shx: \n");

scanf("%f",&shx);

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

if(i==j)

shear[i][j]=1;

else

shear[i][j]=0;

}

}

shear[1][0]=shx;

printf("\nShearing matrix: \n");

print(shear);

multiply(obj,shear,Sh);

printf("Output matrix: \n");

print(Sh);

draw(Sh);

}

break;

case 10:

{

float shear[3][3],shy;

printf("\nEnter shy: \n");

scanf("%f",&shy);

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

if(i==j)

shear[i][j]=1;

else

shear[i][j]=0;

}

}

shear[0][1]=shy;

printf("\nShearing matrix: \n");

print(shear);

multiply(obj,shear,Sh);

printf("Output matrix: \n");

print(Sh);

draw(Sh);

}

break;

default:

printf("\n Invalid choice!");

}

getch();

}

void draw(float D[3][3])

{

line(320+D[0][0],240-D[0][1],320+D[1][0],240-D[1][1]);

line(320+D[1][0],240-D[1][1],320+D[2][0],240-D[2][1]);

line(320+D[2][0],240-D[2][1],320+D[0][0],240-D[0][1]);

}

void print(float A[3][3])

{

int i,j;

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

printf("%0.2f\t",A[i][j]);

}

printf("\n");

}

}

void multiply(float B[3][3],float C[3][3],float D[3][3])

{

int i,j,k;

float s=0.0;

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

s=0.0;

for(k=0;k<3;k++)

{

s=s+(B[i][k]\*C[k][j]);

}

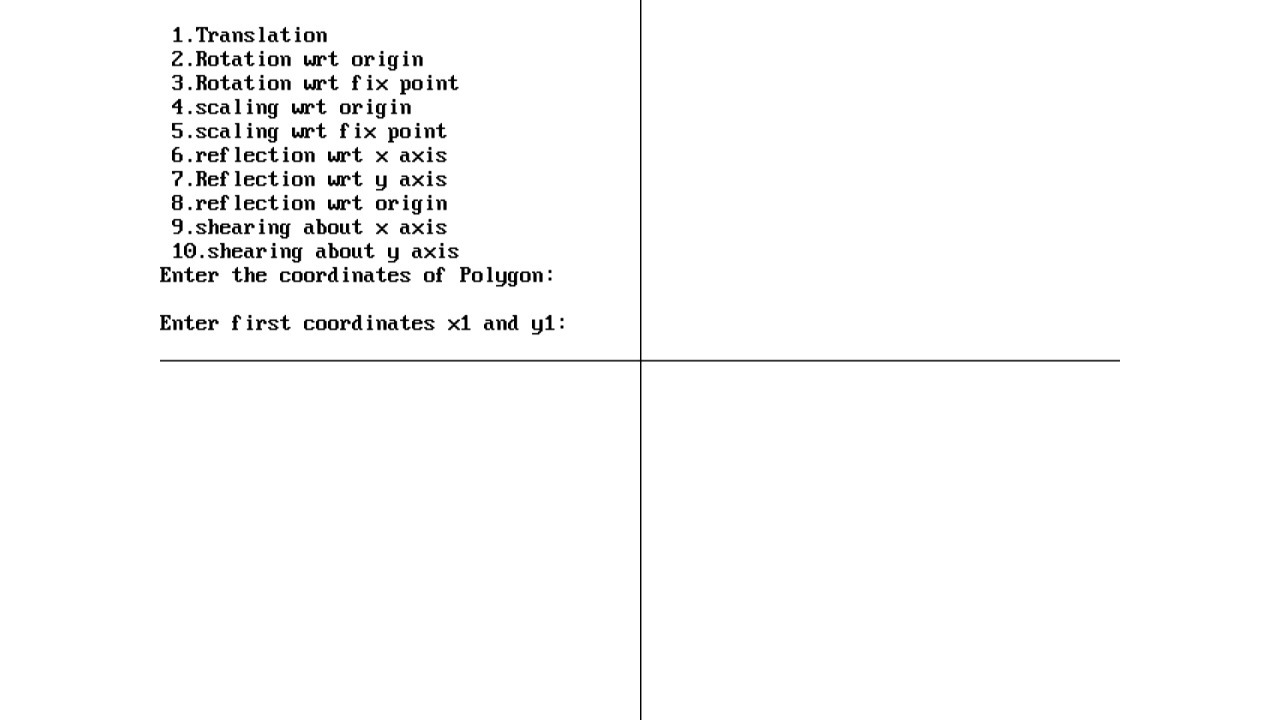
D[i][j]=s;

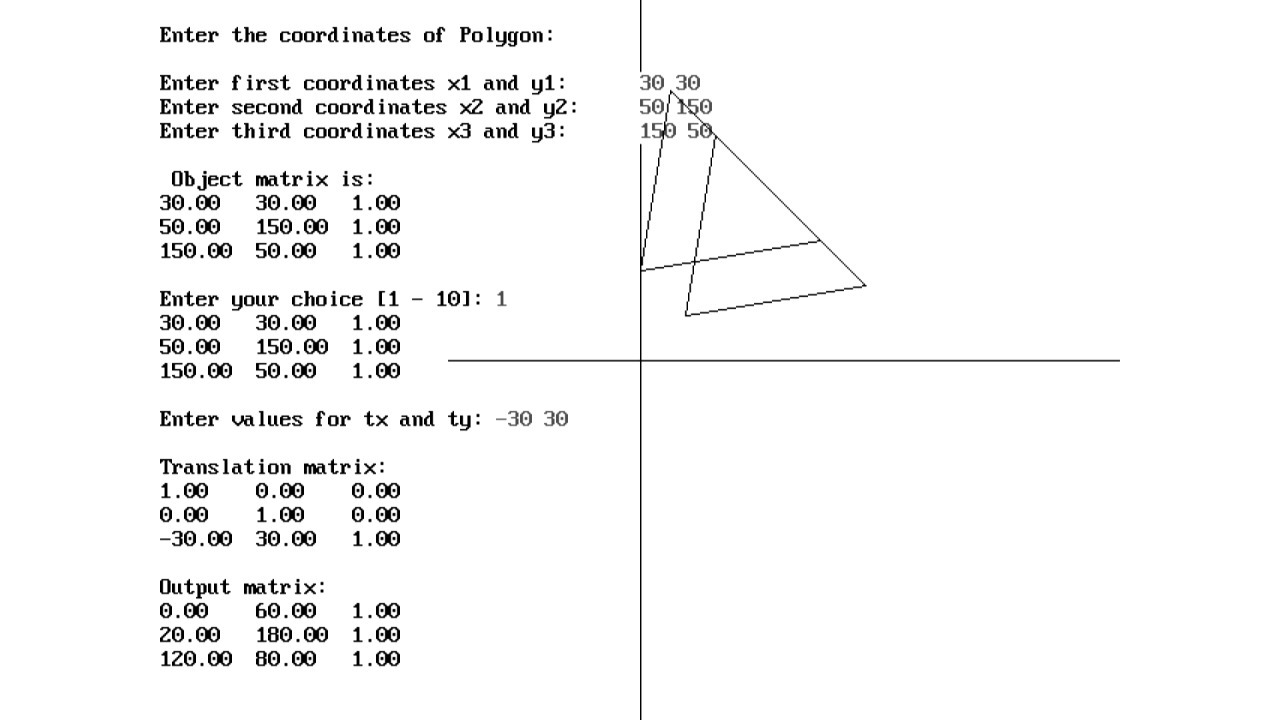
}

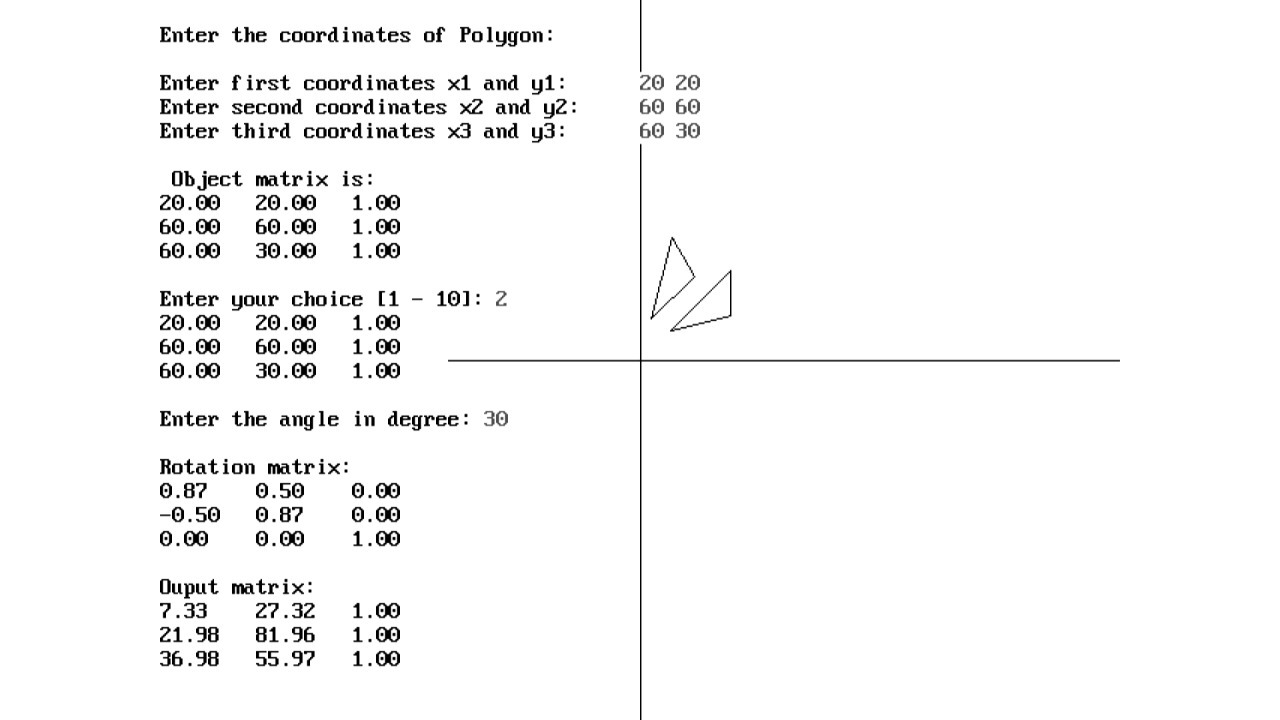
}

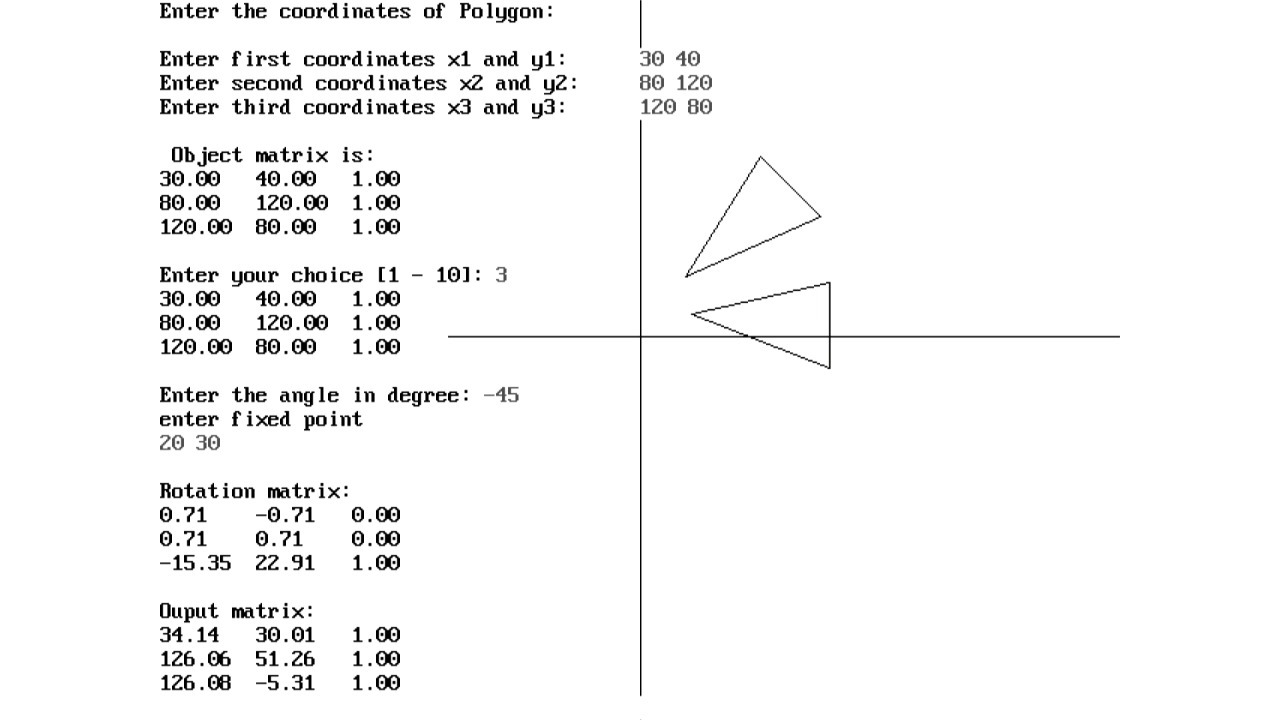
}

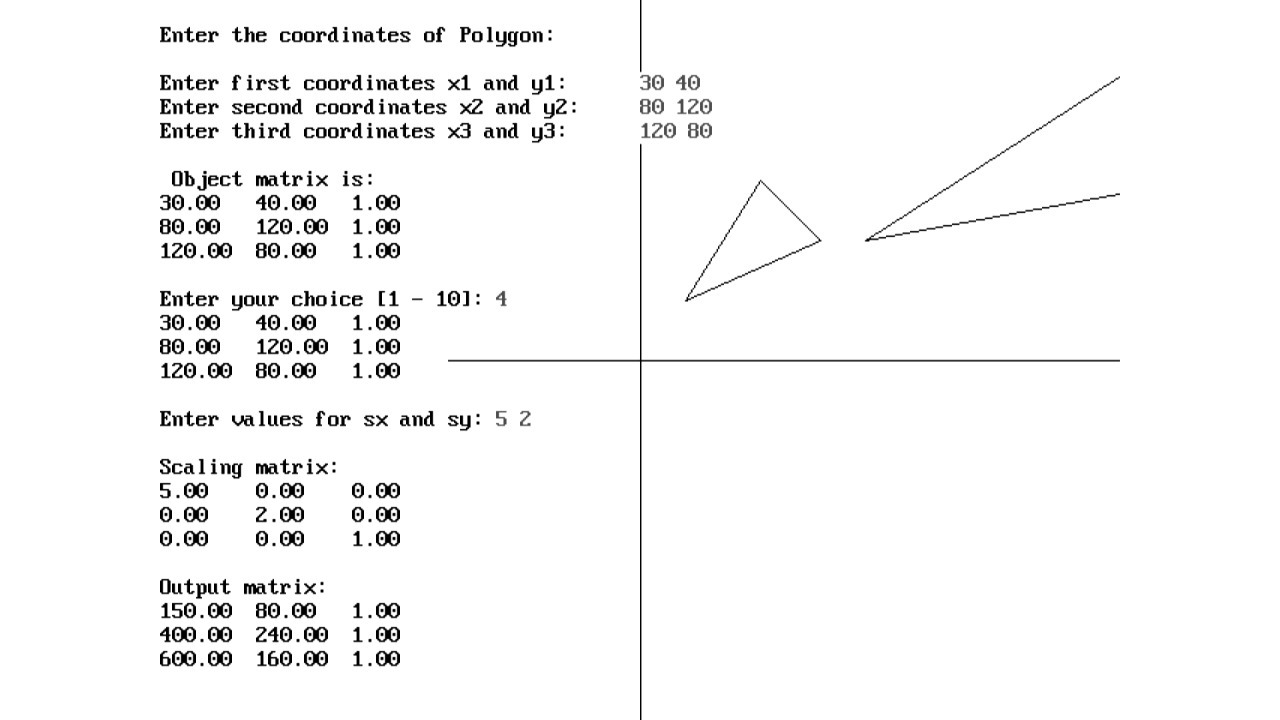
**Output:**

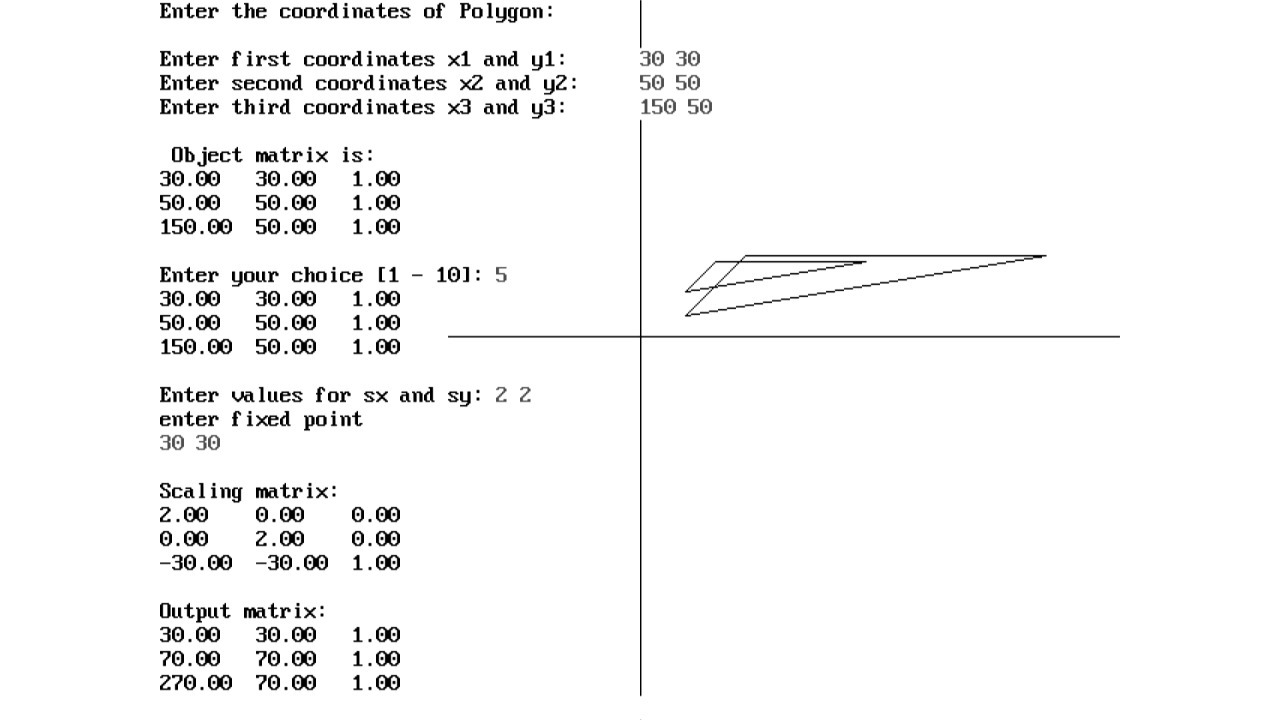
****

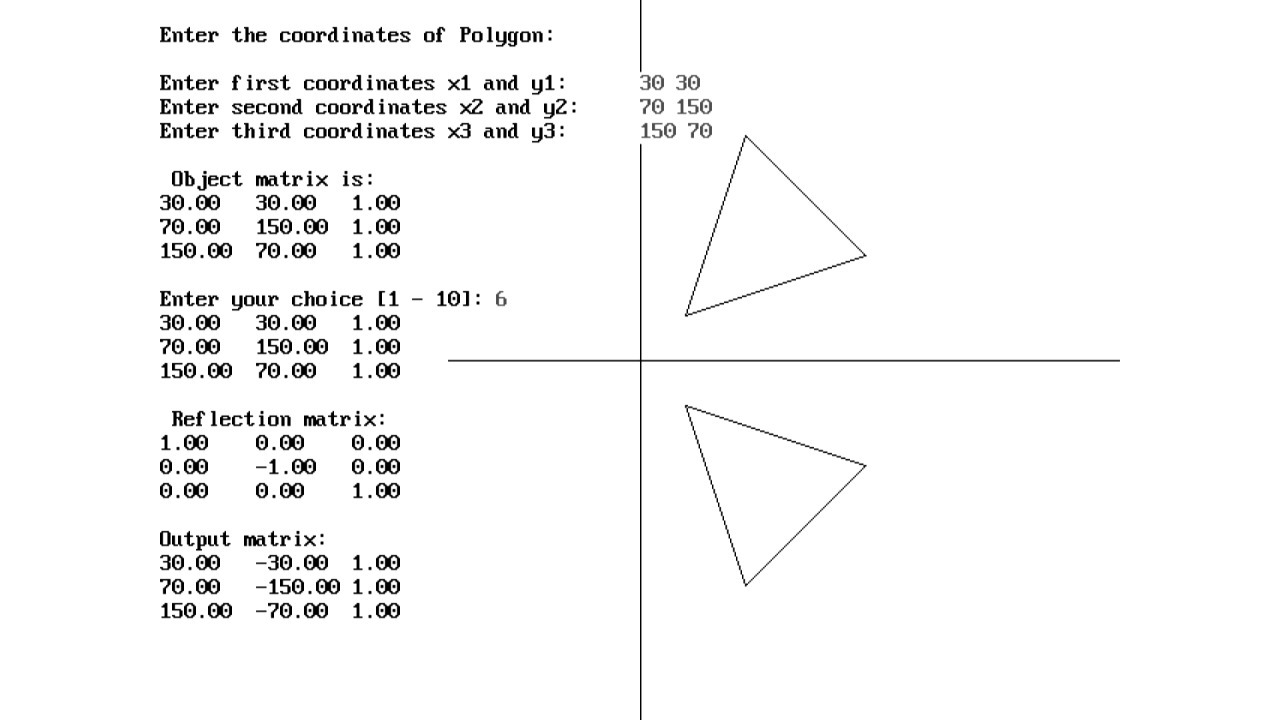
****

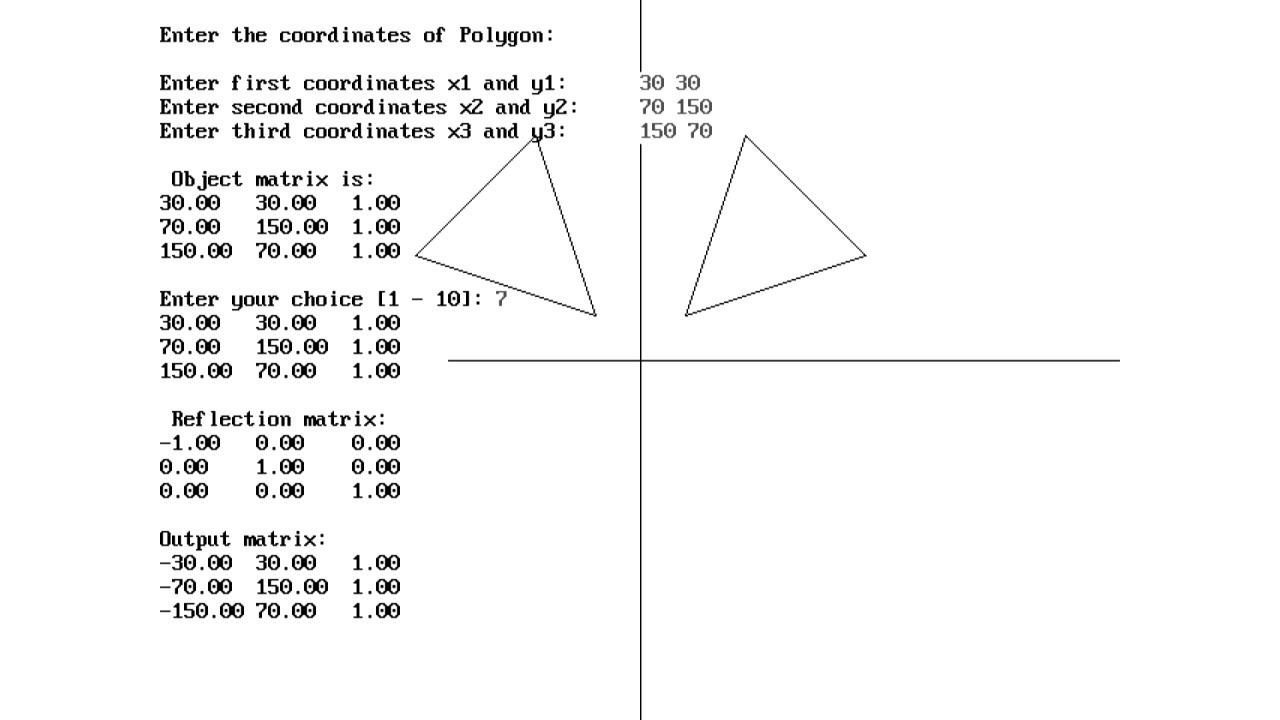
****

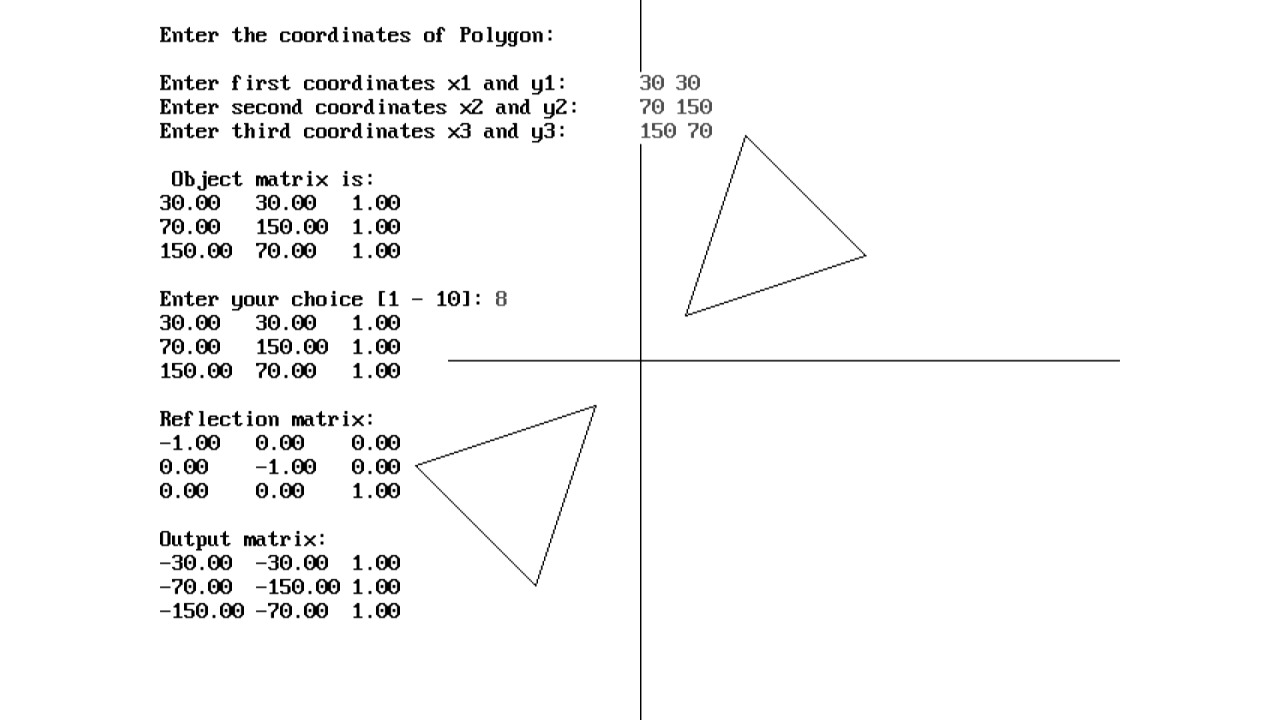
****

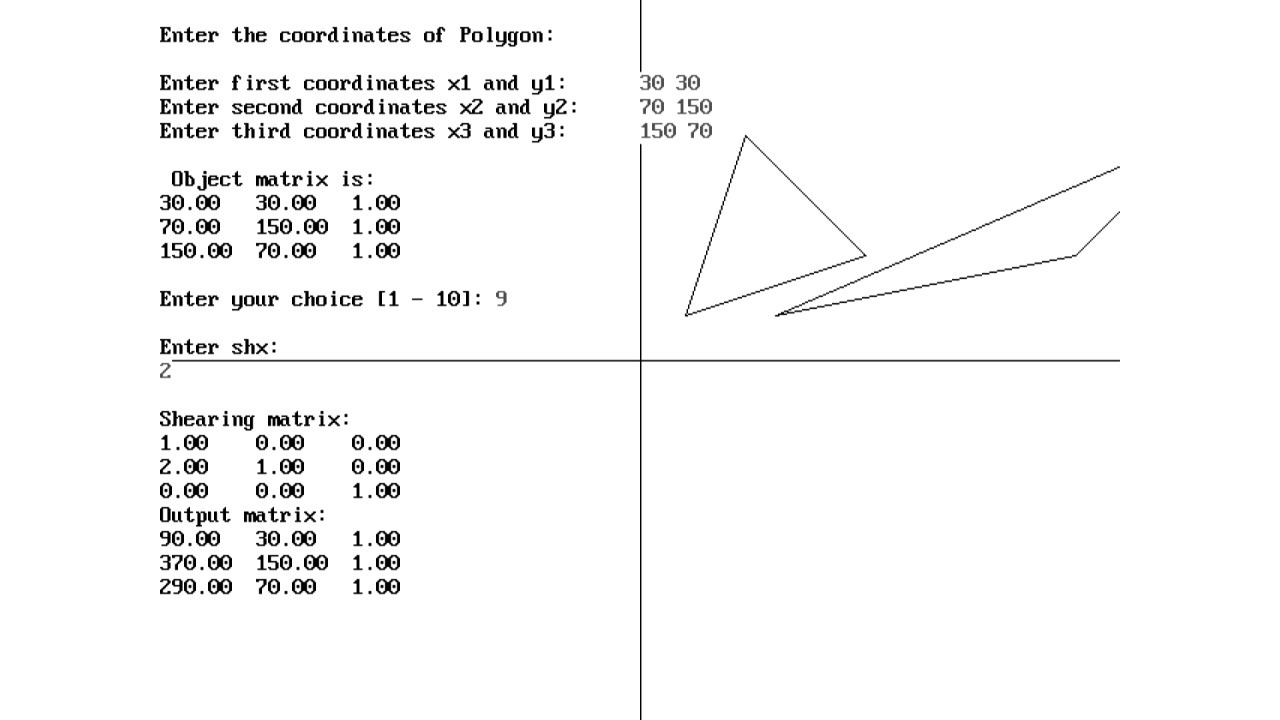
****

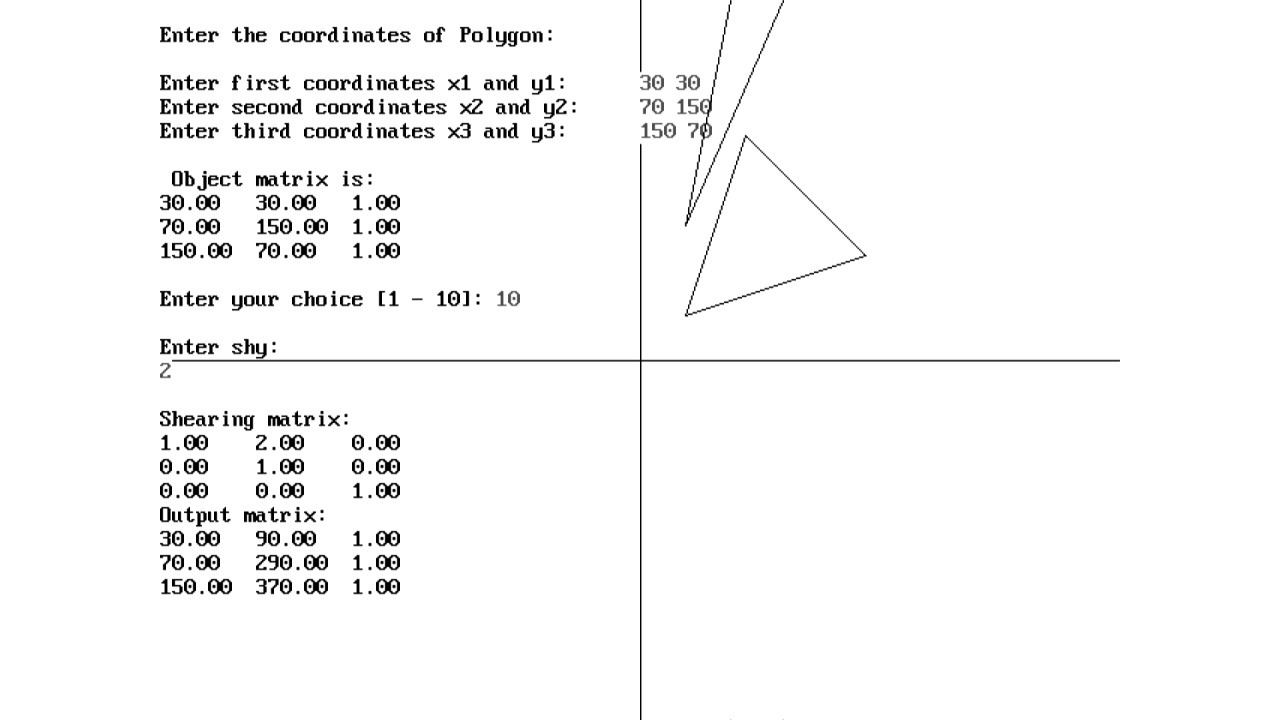
****

****

****

****

****

****