**2D Transformation** #include <graphics.h> #include <stdio.h> #include <conio.h> #include<math.h> void main()

{

int gm,gd=DETECT,midx,midy,c;

float x1,x2,x3,y1,y2,y3,x11,x22,x33,y11,y22,y33,sfx,sfy,tpx,tpy,ang,t,a,b; clrscr();

initgraph(&gd,&gm,"c:\\turboc3\\bgi"); midx=getmaxx()/2;

midy=getmaxy()/2; line(midx,0,midx,getmaxy()); line(0,midy,getmaxx(),midy);

printf("2D Translation, Rotation and Scaling\n");

printf("Enter the points (x1,y1), (x2,y2) and (x3,y3) of triangle\n"); scanf("%f%f%f%f%f%f",&x1,&y1,&x2,&y2,&x3,&y3); setcolor(WHITE);

line(midx+x1,midy-y1,midx+x2,midy-y2); line(midx+x2,midy-y2,midx+x3,midy-y3); line(midx+x3,midy-y3,midx+x1,midy-y1);

printf("\n 1.Translation\n 2.Rotation\n 3.Scaling\n 4.exit\n"); a: printf("Enter your choice:");

scanf("%d",&c); switch(c)

{

case 1:

printf("Enter the translation factor\n"); scanf("%f%f",&tpx,&tpy);

x11=x1+tpx; y11=y1+tpy; x22=x2+tpx; y22=y2+tpy; x33=x3+tpx; y33=y3+tpy; setcolor(RED);

line(midx+x11,midy-y11,midx+x22,midy-y22); line(midx+x22,midy-y22,midx+x33,midy-y33); line(midx+x33,midy-y33,midx+x11,midy-y11); break;

case 2:

printf("Enter the angle of rotation\n"); scanf("%f",&ang);

printf("Enter rotetion point\n"); scanf("%f%f",&a,&b); t=3.14\*ang/180;

x11=abs(x1\*cos(t)-y1\*sin(t)+a\*(1-cos(t))+b\*sin(t)); y11=abs(x1\*sin(t)+y1\*cos(t)+b\*(1-cos(t))-a\*sin(t)); x22=abs(x2\*cos(t)-y2\*sin(t)+a\*(1-cos(t))+b\*sin(t)); y22=abs(x2\*sin(t)+y2\*cos(t)+b\*(1-cos(t))-a\*sin(t)); x33=abs(x3\*cos(t)-y3\*sin(t)+a\*(1-cos(t))+b\*sin(t)); y33=abs(x3\*sin(t)+y3\*cos(t)+b\*(1-cos(t))-a\*sin(t));

setcolor(BLUE);

line(midx+x11,midy-y11,midx+x22,midy-y22); line(midx+x22,midy-y22,midx+x33,midy-y33); line(midx+x33,midy-y33,midx+x11,midy-y11); break;

case 3:

printf("Enter the scalling factor\n"); scanf("%f%f",&sfx,&sfy); printf("Enter scaling point\n"); scanf("%f%f",&a,&b); x11=x1\*sfx+a\*(1-sfx); y11=y2\*sfy+b\*(1-sfy); x22=x2\*sfx+a\*(1-sfx); y22=y2\*sfy+b\*(1-sfy); x33=x3\*sfx+a\*(1-sfx); y33=y3\*sfy+b\*(1-sfy); setcolor(YELLOW);

line(midx+x11,midy-y11,midx+x22,midy-y22); line(midx+x22,midy-y22,midx+x33,midy-y33); line(midx+x33,midy-y33,midx+x11,midy-y11); break;

case 4: exit(0); break;

default:

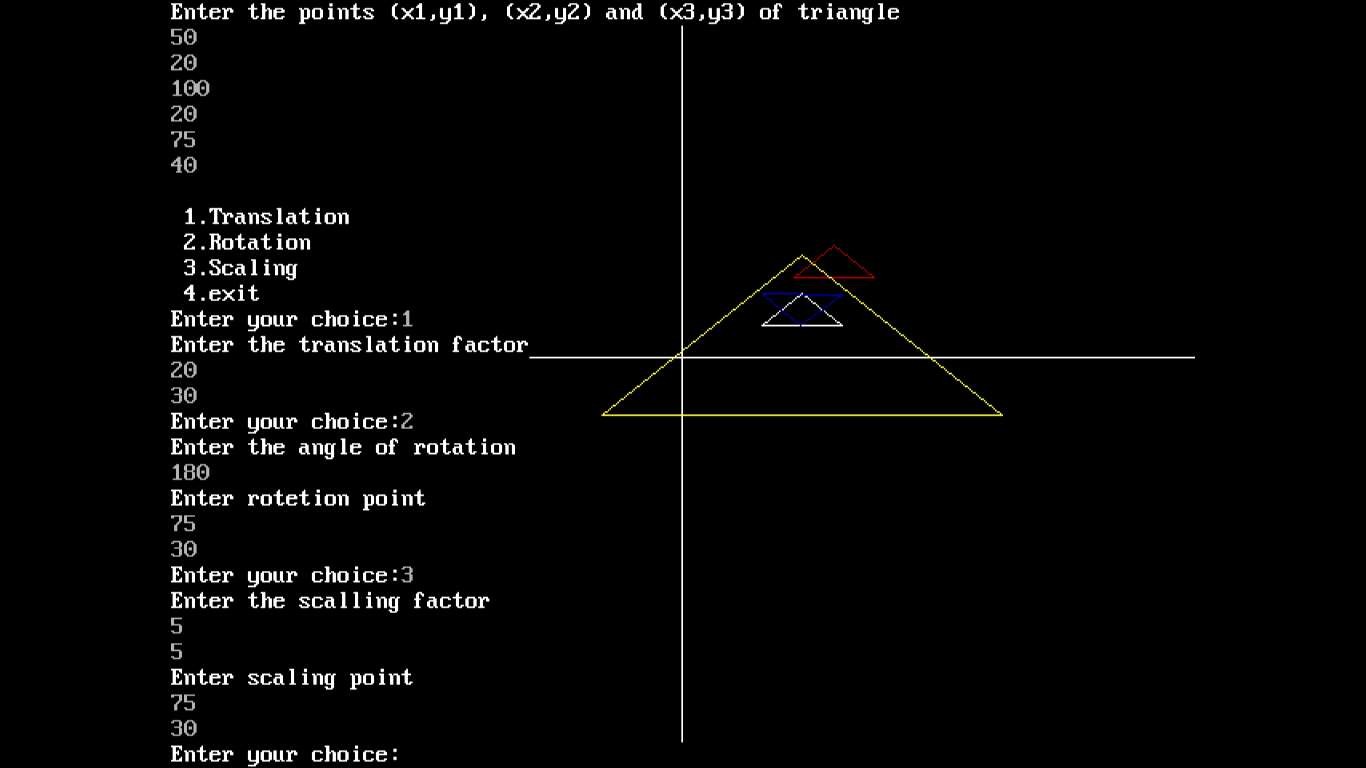
printf("Enter the correct choice\n");

}

goto a;

}

# Output



**3D Transformation** #include<stdio.h> #include<conio.h> #include<graphics.h> #include<math.h>

typedef struct { float x;

float y; float z;

}Point;

Point points;

float temp = 0;

void showPoint(){ printf("<<points.x<<","<<points.y<<","<<points.z<<");

}

void translate(float tx, float ty, float tz){ points.x += tx;

points.y += ty; points.z += tz;

printf("After Translation, new point is :"); showPoint();

}

void rotatex(float angle){

angle = angle \* M\_PI / 180.0; temp = points.y;

points.y = points.y \* cos(angle) - points.z \* sin(angle); points.z = temp \* sin(angle) + points.z \* cos(angle); printf("After rotation about x, new point is: "); showPoint();

}

void rotatey(float angle){

angle = (angle \* M\_PI) / 180.0; temp = points.z;

points.z = points.z \* cos(angle) - points.x \* sin(angle); points.x = temp \* sin(angle) + points.x \* cos(angle); printf("After rotation about y, new point is: "); showPoint();

}

void rotatez(float angle){

angle = angle \* M\_PI / 180.0; temp = points.x;

points.x = points.x \* cos(angle) - points.y \* sin(angle); points.y = temp \* sin(angle) + points.y \*cos(angle); printf("After rotation about z, new point is: "); showPoint();

}

void scale(float sf, float xf, float yf, float zf){ points.x = points.x \* sf + (1 - sf) \* xf; points.y = points.y \* sf + (1 - sf) \* yf; points.z = points.z \* sf + (1 - sf) \* zf; printf("After scaling, new point is: "); showPoint();

}

int main()

{

float tx = 0, ty = 0, tz = 0;

float sf = 0, xf = 0, yf = 0, zf = 0; int choose;

float angle;

printf("Enter the initial point you want to transform:"); scanf("%d",&points.x);

scanf("%d",&points.y);

scanf("%d",&points.z); printf("Choose the following: "); printf("1. Translate");

printf("2. Rotate about X axis"); printf("3. Rotate about Y axis"); printf("4. Rotate about Z axis"); printf("5. Scale");

scanf("%d",&choose); switch(choose){

case 1:

printf("Enter the value of tx, ty and tz: "); scanf("%d",&tx);

scanf("%d",&ty);

scanf("%d",&tz); translate(tx, ty, tz); break;

case 2:

printf("Enter the angle: "); scanf("%d",&angle); rotatex(angle);

break; case 3:

printf("Enter the angle: "); scanf("%d",&angle); rotatey(angle);

break; case 4:

printf("Enter the angle: "); scanf("%d",&angle); rotatez(angle);

break; case 5:

printf("Enter the value of sf, xf, yf and zf: "); scanf("%d",&sf);

scanf("%d",&xf);

scanf("%d",&yf);

scanf("%d",&zf);

scale(sf, xf, yf, zf); break;

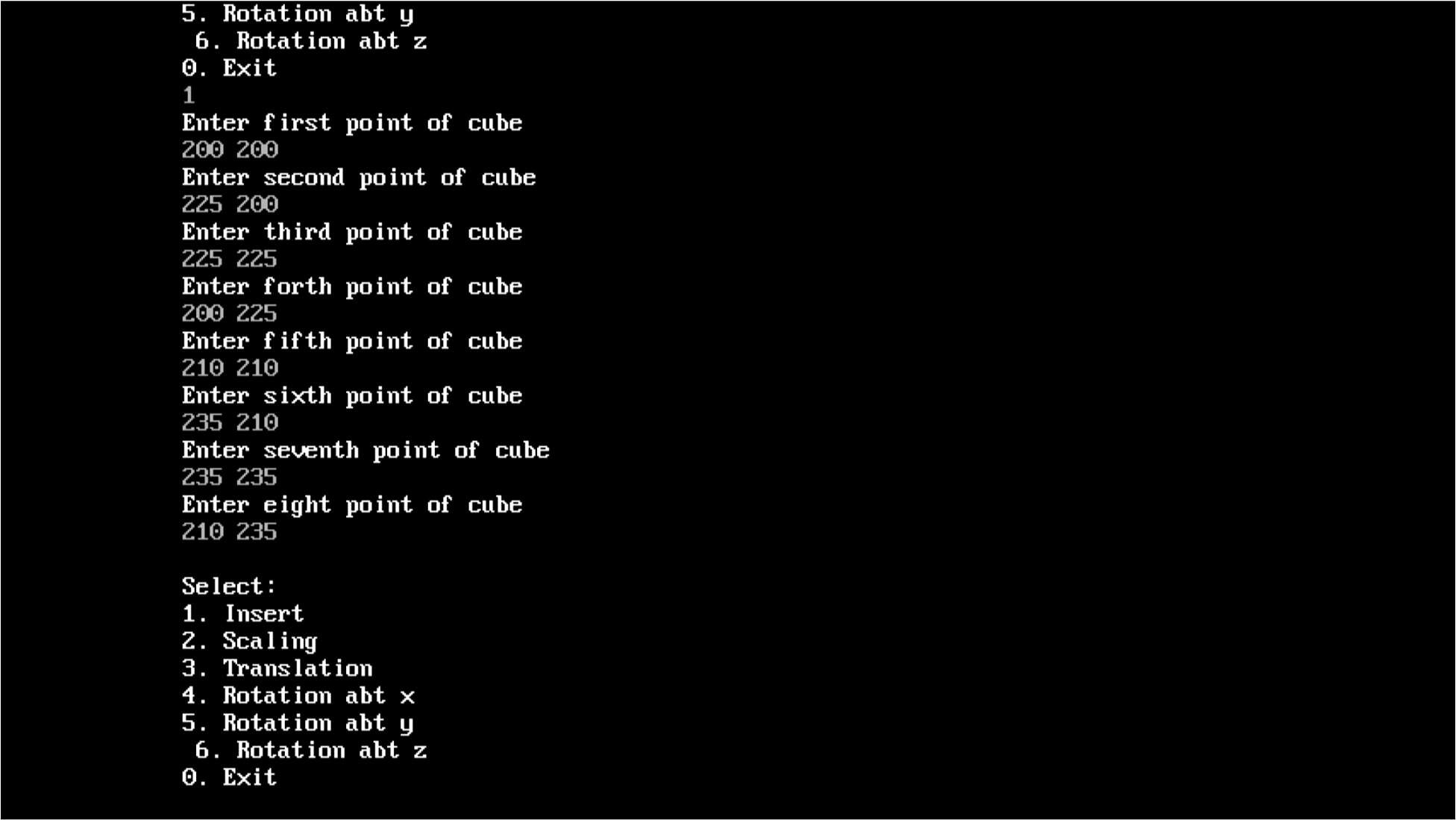
default: break;

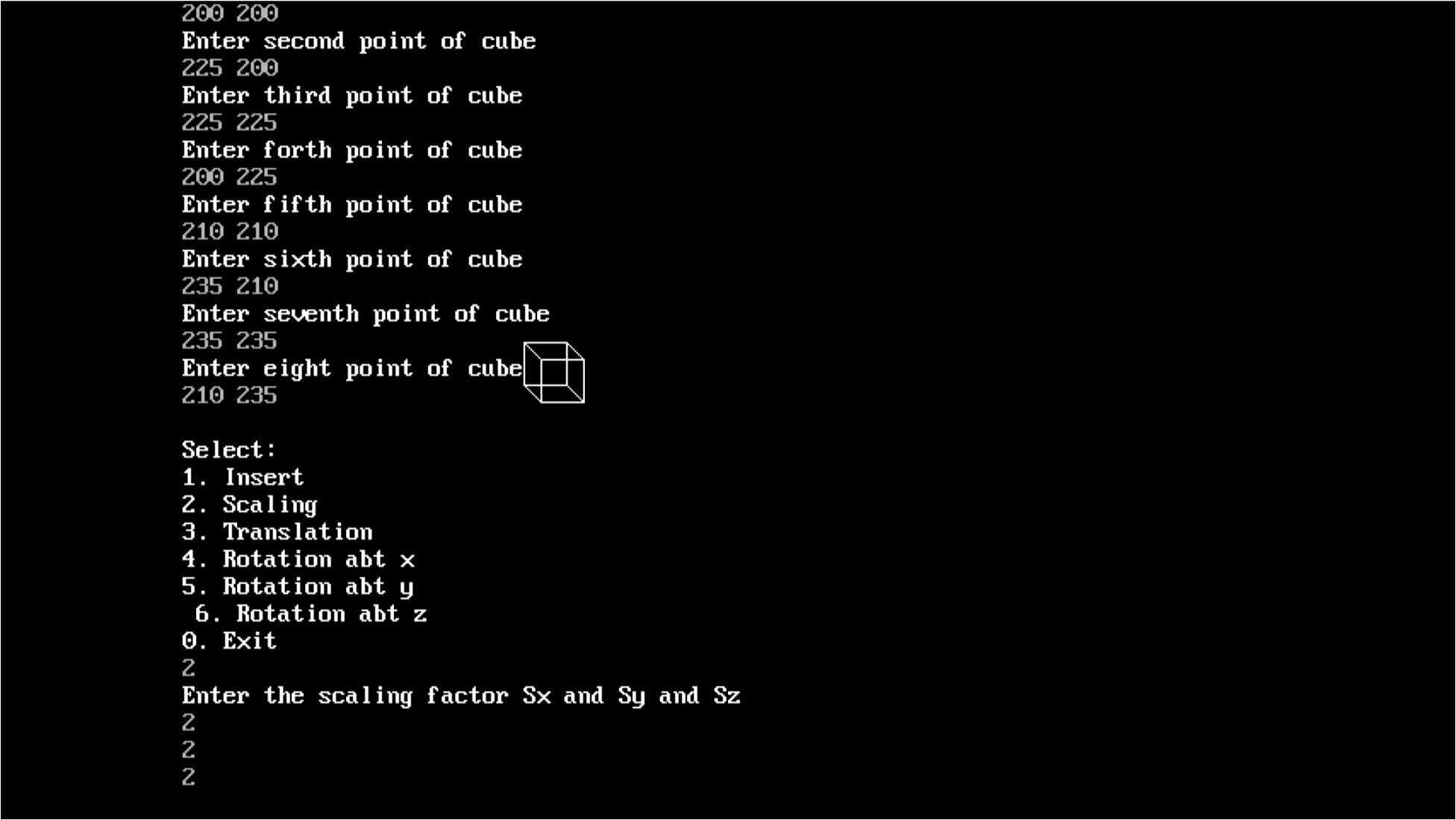
}

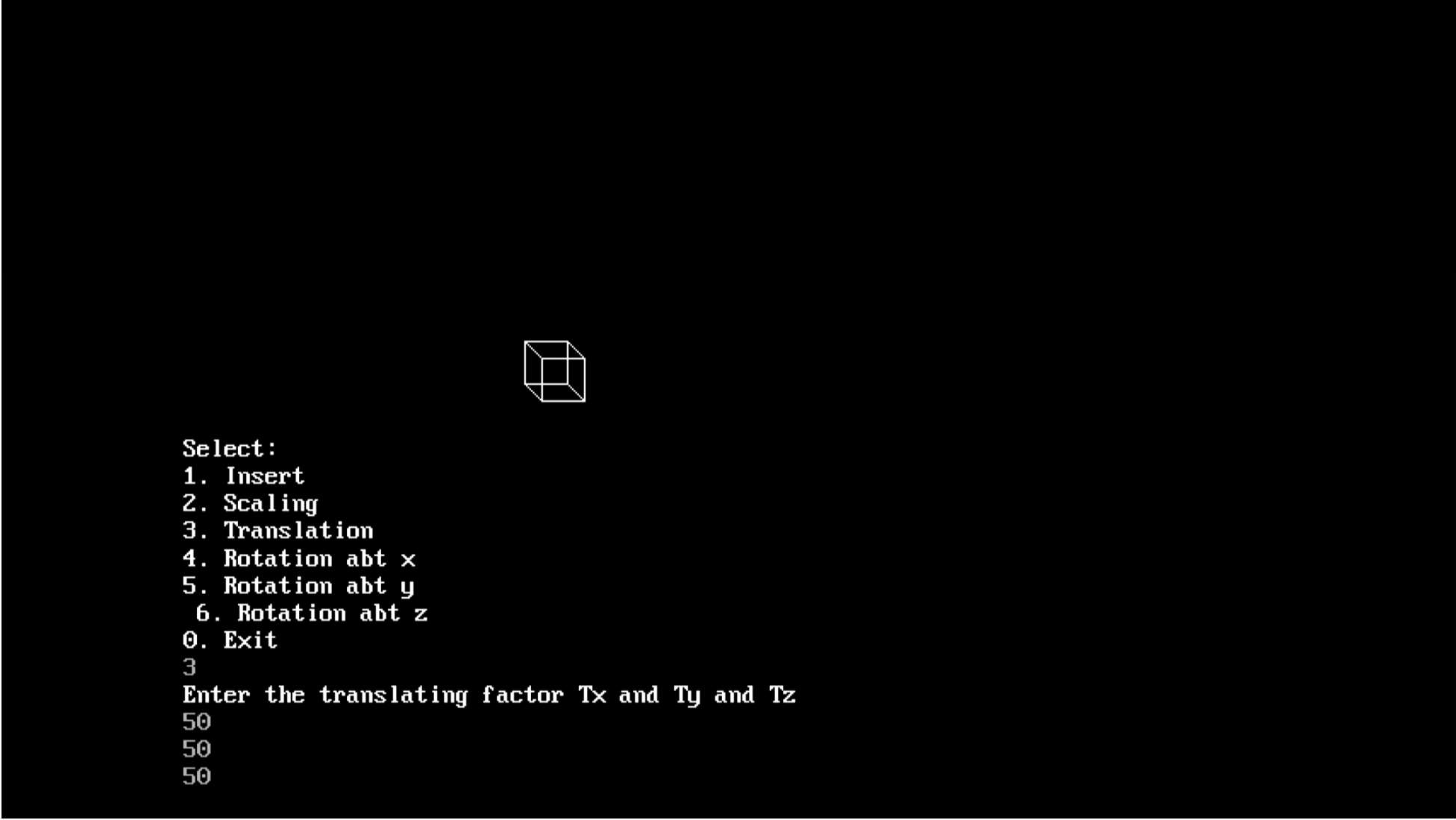
return 0;

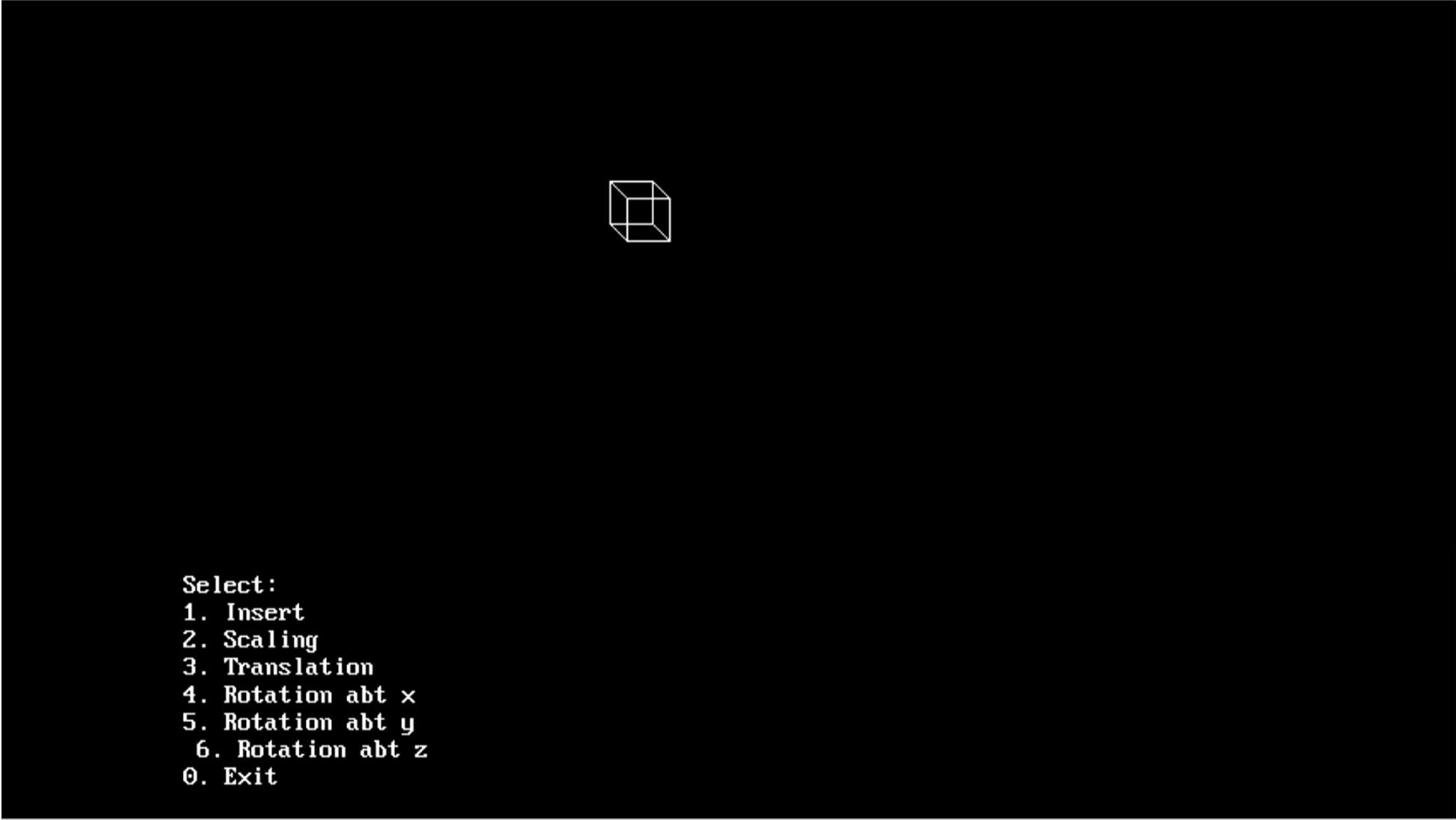
}

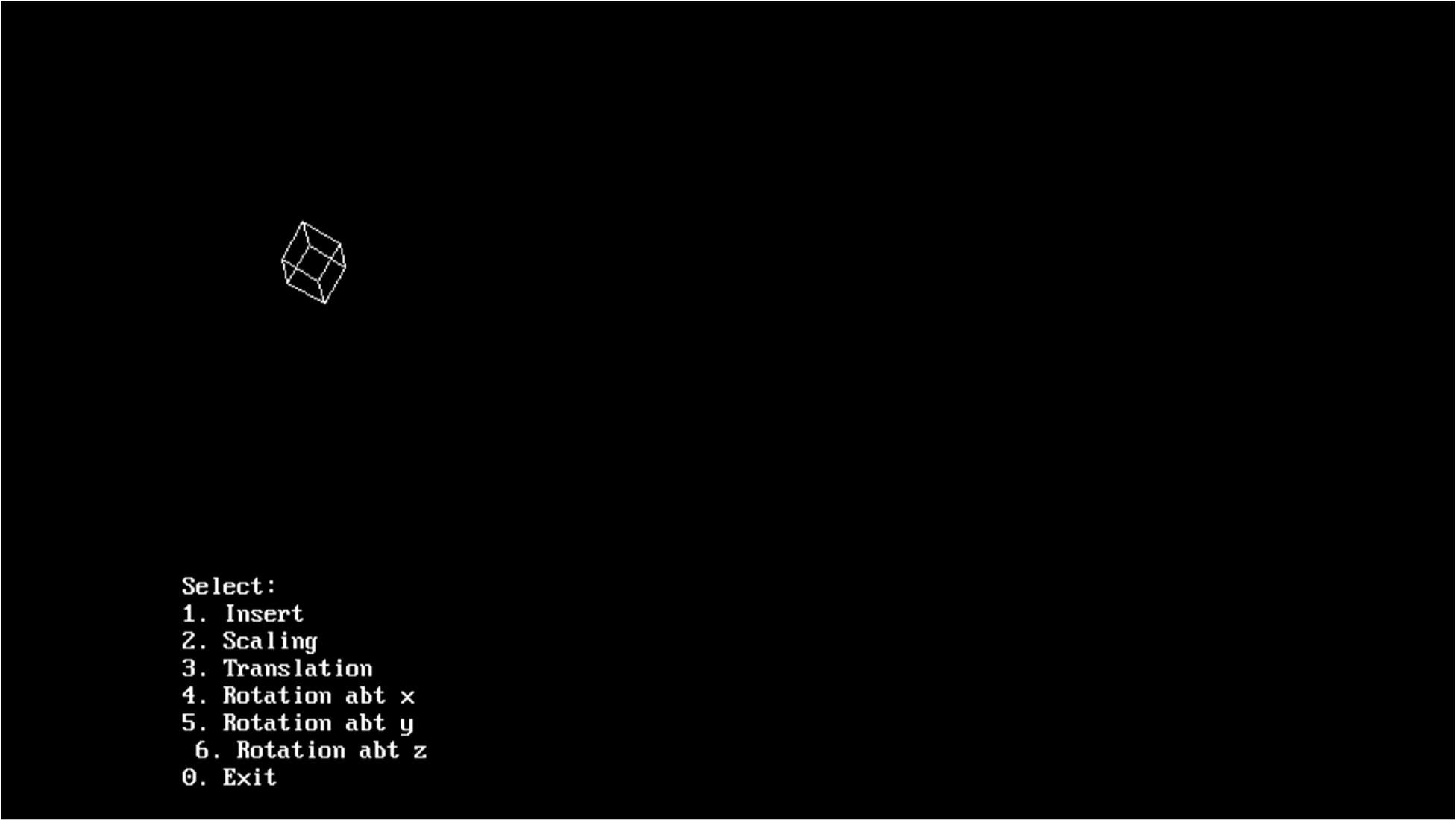
# Output

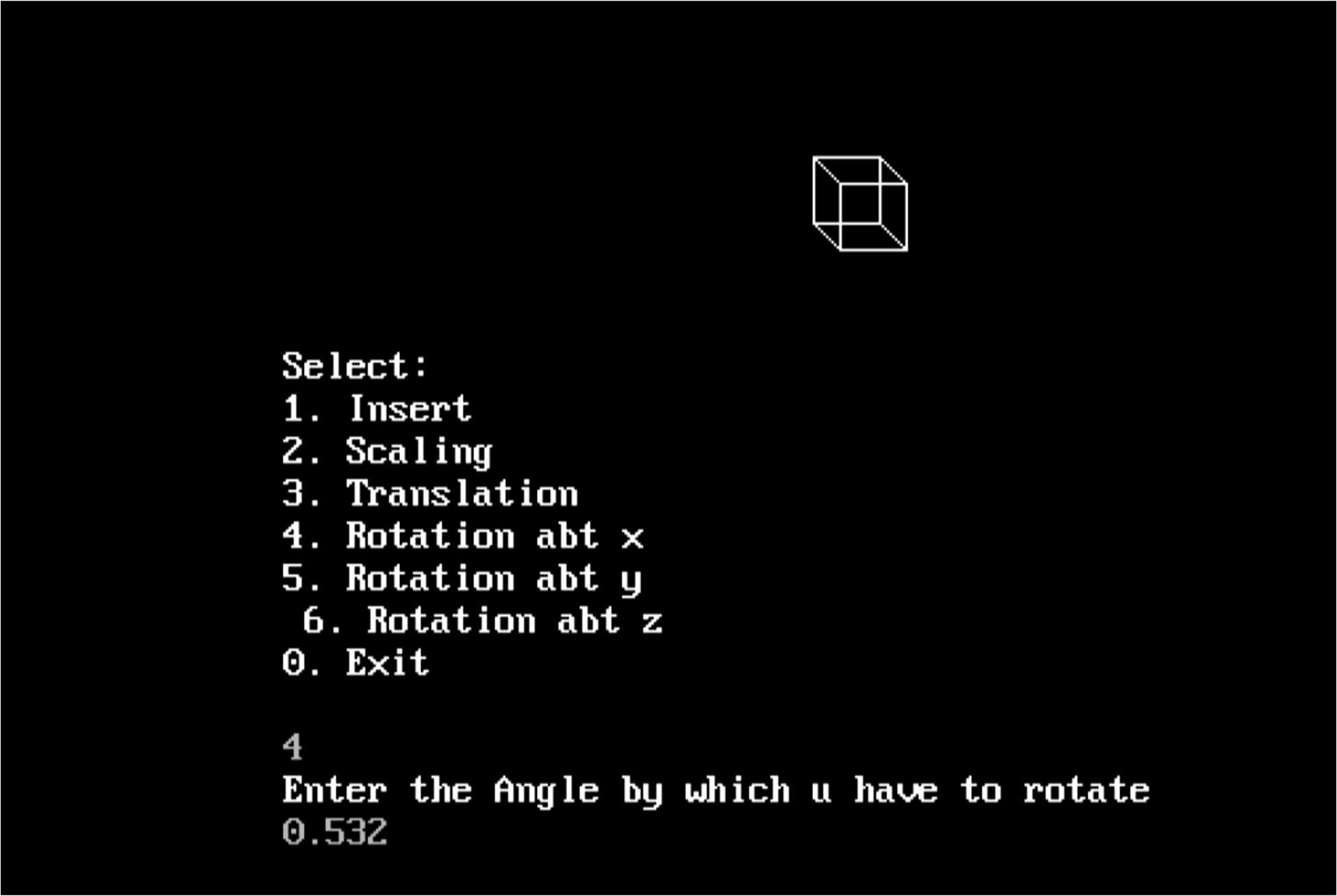












## Bezier

#include <stdio.h> #include <stdlib.h> #include <graphics.h> #include <math.h>

void bezier (int x[4], int y[4])

{

int gd = DETECT, gm; int i;

double t;

initgraph (&gd, &gm, "C:\\turboc3\\bgi");

for (t = 0.0; t < 1.0; t += 0.0005)

{

double xt = pow (1-t, 3) \* x[0] + 3 \* t \* pow (1-t, 2) \* x[1] + 3 \* pow (t, 2) \* (1-t) \* x[2] + pow (t, 3) \* x[3];

double yt = pow (1-t, 3) \* y[0] + 3 \* t \* pow (1-t, 2) \* y[1] + 3 \* pow (t, 2) \* (1-t) \* y[2] + pow (t, 3) \* y[3];

putpixel (xt, yt, RED);

}

for (i=0; i<4; i++)

putpixel (x[i], y[i], YELLOW);

getch(); closegraph(); return;

}

void main()

{

int x[4], y[4]; int i;

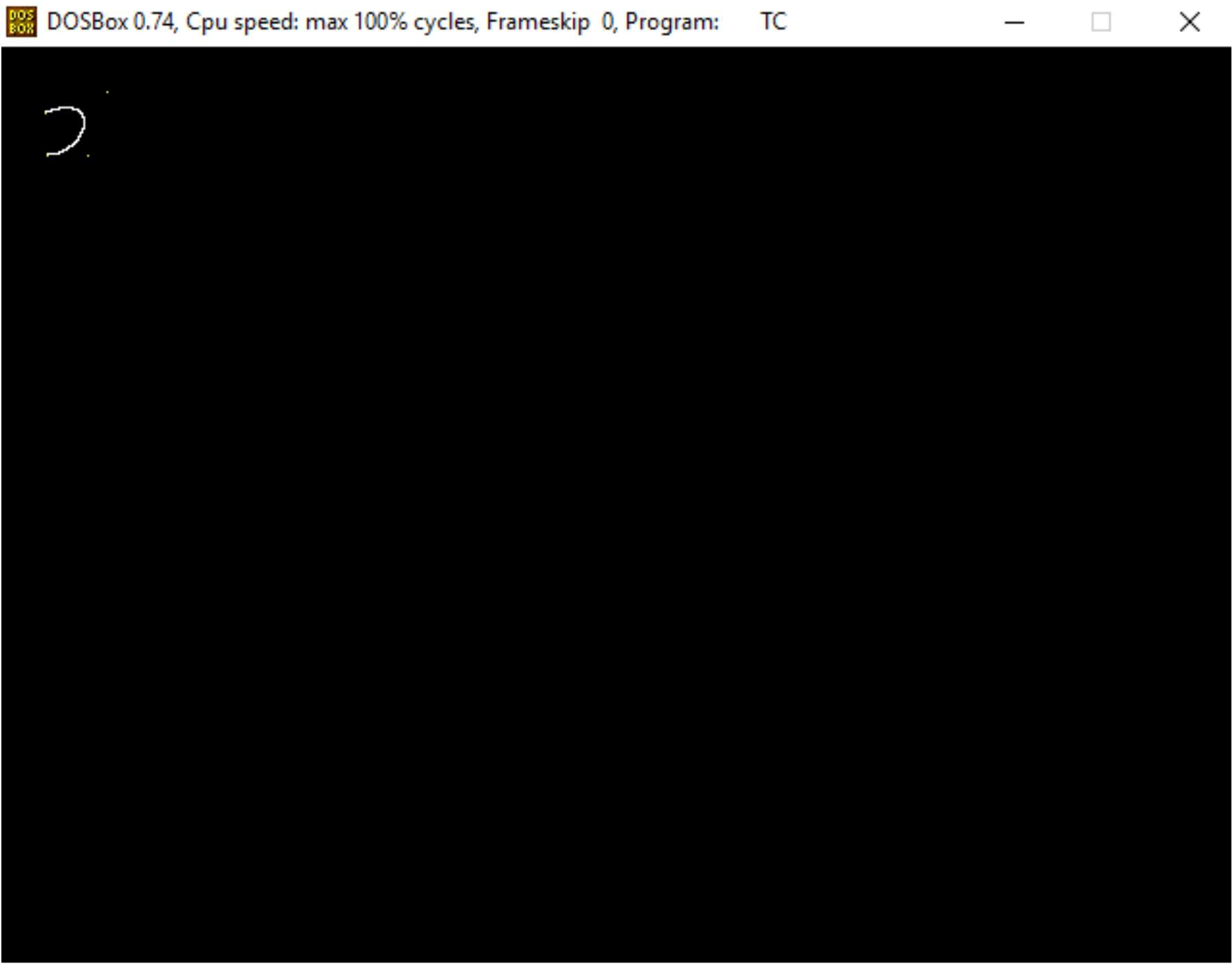
printf ("Enter the x- and y-coordinates of the four control points.\n"); for (i=0; i<4; i++)

scanf ("%d %d", &x[i], &y[i]);

bezier (x, y);

}

**Ouput**



## Projection

**Perspective Projection**

#include<stdio.h>

#include<math.h>

#include<graphics.h>

main()

{

int x1,y1,x2,y2,gd,gm;

int ymax,a[4][8];

float par[4][4],b[4][8];

int i,j,k,m,n,p;

int xp, yp, zp, x, y, z;

a[0][0] = 100; a[1][0] = 100; a[2][0] = -100;

a[0][1] = 200; a[1][1] = 100; a[2][1] = -100;

a[0][2] = 200; a[1][2] = 200; a[2][2] = -100;

a[0][3] = 100; a[1][3] = 200; a[2][3] = -100;

a[0][4] = 100; a[1][4] = 100; a[2][4] = -200;

a[0][5] = 200; a[1][5] = 100; a[2][5] = -200;

a[0][6] = 200; a[1][6] = 200; a[2][6] = -200;

a[0][7] = 100; a[1][7] = 200; a[2][7] = -200;

detectgraph(&gd,&gm);

initgraph(&gd,&gm, "c:\\tc\\bgi");

ymax = getmaxy();

xp = 300; yp = 320; zp = 100;

for(j=0; j<8; j++)

{

x = a[0][j]; y = a[1][j]; z = a[2][j];

b[0][j] = xp - ( (float)( x - xp )/(z - zp)) \* (zp);

b[1][j] = yp - ( (float)( y - yp )/(z - zp)) \* (zp);

}

/\*- front plane display -\*/

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

{

x1=(int) b[0][j]; y1=(int) b[1][j];

x2=(int) b[0][j+1]; y2=(int) b[1][j+1];

line( x1,ymax-y1,x2,ymax-y2);

}

x1=(int) b[0][3]; y1=(int) b[1][3];

x2=(int) b[0][0]; y2=(int) b[1][0];

line( x1, ymax-y1, x2, ymax-y2);

/\*- back plane display -\*/

setcolor(11);

for(j=4;j<7;j++)

{

x1=(int) b[0][j]; y1=(int) b[1][j];

x2=(int) b[0][j+1]; y2=(int) b[1][j+1];

line( x1, ymax-y1, x2, ymax-y2);

}

x1=(int) b[0][7]; y1=(int) b[1][7];

x2=(int) b[0][4]; y2=(int) b[1][4];

line( x1, ymax-y1, x2, ymax-y2); setcolor(7);

for(i=0;i<4;i++)

{

x1=(int) b[0][i]; y1=(int) b[1][i];

x2=(int) b[0][4+i]; y2=(int) b[1][4+i];

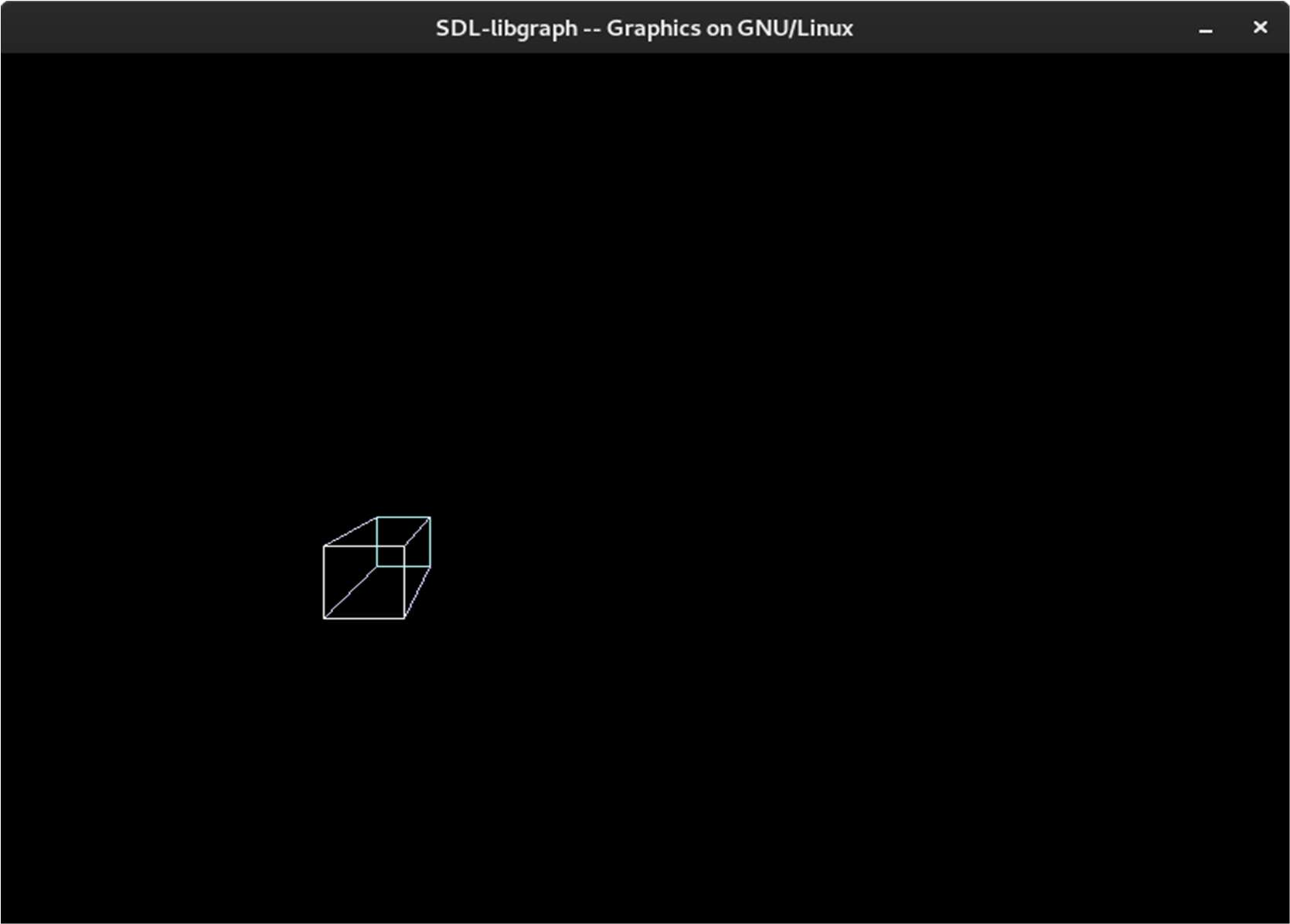
line( x1, ymax-y1, x2, ymax-y2);

}

getch();

}

## Output



**PARALLEL PROJECTION**

#include<stdio.h>

#include<conio.h> #include<graphics.h> void main()

{

int ch,x[10],y[10],z[10],a,b,c,j,xr[10],yr[10],zr[10]; int gd=DETECT,gm;

printf("\n\n parallel projection"); printf("\n\n to enter coordinates\n\n") ;

printf("\n else any other value would lead to default value \t:");

scanf("%d",&ch); if(ch==1)

{

printf("\n follow these steps"); for(int i=0;i<8;i++) scanf("%d",&x[i],&y[i],&z[i]);

}

else

{ x[0]=y[0]=z[0]=0;

x[1]=80;y[1]=z[1]=0;

x[2]=y[2]=80;z[2]=0;

x[3]=0;y[3]=80;z[3]=0;

x[4]=0;y[4]=z[4]=80;

x[5]=y[5]=0;z[5]=80;

x[6]=80;y[6]=0;z[6]=80;

x[7]=y[7]=z[7]=80;

printf("\n default values have been set");

}

printf("\n\n now enter the projection vector \t");

scanf("%d%d%d",&a,&b,&c); for(int i=0;i<8;i++)

{

xr[i]=x[i]-(a\*z[i]/c);

yr[i]=y[i]-(b\*z[i]/c); zr[i]=0;

}

initgraph(&gd,&gm,""); setcolor(CYAN); line(300,0,300,480); line(0,240,600,240);

setcolor(YELLOW); for(i=0,j=i+1;i<8;i++,j=(j+1)/8); line(xr[i]+300,240-yr[i],xr[j]+300,240-yr[i]); getch();

closegraph();

}

## Oblique Projection

#include<stdio.h>

#include<math.h>

#include<graphics.h>

main()

{

int x1,y1,x2,y2,gd,gm;

int ymax,a[4][8];

float par[4][4],b[4][8];

int i,j,k,m,n,p;

double L1,phi;

a[0][0] = 100; a[1][0] = 100; a[2][0] = 100;

a[0][1] = 200; a[1][1] = 100; a[2][1] = 100;

a[0][2] = 200; a[1][2] = 200; a[2][2] = 100;

a[0][3] = 100; a[1][3] = 200; a[2][3] = 100;

a[0][4] = 100; a[1][4] = 100; a[2][4] = 200;

a[0][5] = 200; a[1][5] = 100; a[2][5] = 200;

a[0][6] = 200; a[1][6] = 200; a[2][6] = 200;

a[0][7] = 100; a[1][7] = 200; a[2][7] = 200;

phi = (double) (3.14\*45.0)/180 ;

L1 = 0.5;

par[0][0] = 1; par[0][1] = 0;

par[0][2] = L1\*cos(phi); par[0][3] = 0;

par[1][0] = 0; par[1][1] = 1;

par[1][2] = L1\*sin(phi); par[1][3] = 0;

par[2][0] = 0; par[2][1] = 0;

par[2][2] = 0; par[2][3] = 0;

par[3][0] = 0; par[3][1] = 0;

par[3][2] = 0; par[3][3] = 1;

m=4; n=4; p=8;

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

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

b[i][k] = 0; for(i=0; i<m; i++)

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

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

b[i][k] += (float)par[i][j] \* a[j][k];

detectgraph(&gd,&gm);

initgraph(&gd,&gm, "c:\\tc\\bgi");

ymax = getmaxy();

/\*- front plane display -\*/

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

{

x1=(int) b[0][j]; y1=(int) b[1][j];

x2=(int) b[0][j+1]; y2=(int) b[1][j+1];

line( x1,ymax-y1,x2,ymax-y2);

}

x1=(int) b[0][3]; y1=(int) b[1][3];

x2=(int) b[0][0]; y2=(int) b[1][0];

line( x1,ymax-y1,x2,ymax-y2);

/\*- back plane display -\*/

setcolor(11);

for(j=4;j<7;j++)

{

x1=(int) b[0][j]; y1=(int) b[1][j];

x2=(int) b[0][j+1]; y2=(int) b[1][j+1];

line( x1,ymax-y1,x2,ymax-y2);

}

x1=(int) b[0][7]; y1=(int) b[1][7];

x2=(int) b[0][4]; y2=(int) b[1][4];

line( x1,ymax-y1,x2,ymax-y2);

setcolor(13);

for(i=0;i<4;i++)

{

x1=(int) b[0][i]; y1=(int) b[1][i];

x2=(int) b[0][4+i]; y2=(int) b[1][4+i];

line( x1,ymax-y1,x2,ymax-y2);

}

getch();

}

## Output

