1. **3D**

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<process.h>
#include<new.h>
#include<math.h>

void colorpixel(int x,int y,int color)
{
        putpixel(x+(getmaxx()/2),(getmaxy()/2)-y,color);
}

void drawaxis()
{
        int xmid=getmaxx()/2;
        int ymid=getmaxy()/2;
        line(xmid,0,xmid,getmaxy());
        line(0,ymid,getmaxx(),ymid);
}

class object
{
private:
        float vertex[8][4],final[8][4];
        int edge[12][2];
        float mult(float a[][4],float b[][4])
        {
                float sum=0.0;
                for(int i=0;i<8;i++)
                        for(int j=0;j<4;j++)
                        {
                                for(int k=0;k<4;k++)
                                        sum=sum+(a[i][k]*b[k][j]);
                                final[i][j]=sum;
                                sum=0.0;
                        }
        }
public:
        void input()
        {
                // VERTEX INFORMATION
                int n;
                for(int i=0;i<8;i++)
```

```
		{
			n=i;
			for(int j=2;j>=0;j--)
			{
				vertex[i][j]=(n%2)*50;
				n=n/2;
			}
			vertex[i][3]=1.0;

		}

		// EDGE INFORMATION
		edge[0][0]=0;
		edge[0][1]=4;
		edge[1][0]=0;
		edge[1][1]=1;
		edge[2][0]=0;
		edge[2][1]=2;
		edge[3][0]=1;
		edge[3][1]=5;
		edge[4][0]=1;
		edge[4][1]=3;
		edge[5][0]=2;
		edge[5][1]=3;
		edge[6][0]=2;
		edge[6][1]=6;
		edge[7][0]=3;
		edge[7][1]=7;
		edge[8][0]=4;
		edge[8][1]=5;
		edge[9][0]=4;
		edge[9][1]=6;
		edge[10][0]=5;
		edge[10][1]=7;
		edge[11][0]=6;
		edge[11][1]=7;
}

void display()
{
		//clrscr();
		int x=getmaxx()/2;
		int y=getmaxy()/2;
		int v1,v2;
```

```
	//cout<<"Previous object is"<<endl;
	drawaxis();
	for(int i=0;i<12;i++)
	{
		v1=edge[i][0];
		v2=edge[i][1];
		line(final[v1][0]+x,y-final[v1][1],x+final[v2][0],y-final[v2][1]);
	}
}

void orthographic()
{
	float transform[4][4];
	transform[0][0]=1.0;
	transform[0][1]=0.0;
	transform[0][2]=0.0;
	transform[0][3]=0.0;
	transform[1][0]=0.0;
	transform[1][1]=1.0;
	transform[1][2]=0.0;
	transform[1][3]=0.0;
	transform[2][0]=0.0;
	transform[2][1]=0.0;
	transform[2][2]=0.0;
	transform[2][3]=0.0;
	transform[3][0]=0.0;
	transform[3][1]=0.0;
	transform[3][2]=0.0;
	transform[3][3]=1.0;
	mult(vertex,transform);
	display();
}

void trimetric()		// first rotation with and then x
{
	float transform[4][4];
	float theta,cos_t,sin_t,pie,cos_p,sin_p;
	cout<<"enter the value of theta";
	cin>>theta;
	cout<<"enter the value of pie";
	cin>>pie;
	if(theta==90)
	{
		cos_t=0.0;
```

```
        sin_t=1.0;
}
else if(theta==180)
{
        cos_t=-1.0;
        sin_t=0.0;
}
else if(theta==270 || theta==-90)
{
        cos_t=0.0;
        sin_t=-1.0;
}
else if(theta==360)
{
        cos_t=1.0;
        sin_t=0.0;
}
else
{
        cos_t=cos((M_PI/180)*theta);
        sin_t=sin((M_PI/180)*theta);
}

if(pie==90)
{
        cos_p=0.0;
        sin_p=1.0;
}
else if(pie==180)
{
        cos_p=-1.0;
        sin_p=0.0;
}
else if(pie==270 || pie==-90)
{
        cos_p=0.0;
        sin_p=-1.0;
}
else if(pie==360)
{
        cos_p=1.0;
        sin_p=0.0;
}
else
```

```
	{
		cos_p=cos((M_PI/180)*pie);
		sin_p=sin((M_PI/180)*pie);
	}

	transform[0][0]=cos_p;
	transform[0][1]=sin_p*sin_t;
	transform[0][2]=0.0;
	transform[0][3]=0.0;
	transform[1][0]=0.0;
	transform[1][1]=cos_t;
	transform[1][2]=0.0;
	transform[1][3]=0.0;
	transform[2][0]=sin_p;
	transform[2][1]=-cos_p*sin_t;
	transform[2][2]=0.0;
	transform[2][3]=0.0;
	transform[3][0]=0.0;
	transform[3][1]=0.0;
	transform[3][2]=0.0;
	transform[3][3]=1.0;
	mult(vertex,transform);
	display();
}

void dimetric()		// first rotation with and then x
{
	float transform[4][4];
	float cos_t,sin_t,cos_p,sin_p,fz;
	cout<<"enter the forshortning factor along z-axis";
	cin>>fz;
	sin_t=fz/(sqrt(2));
	cos_t=sqrt(1-(pow(sin_t,2)));

	sin_p=fz/(sqrt(2-(pow(fz,2))));
	cos_p=sqrt(1-(pow(sin_p,2)));

	transform[0][0]=cos_p;
	transform[0][1]=sin_p*sin_t;
	transform[0][2]=0.0;
	transform[0][3]=0.0;
	transform[1][0]=0.0;
	transform[1][1]=cos_t;
	transform[1][2]=0.0;
```

```
        transform[1][3]=0.0;
        transform[2][0]=sin_p;
        transform[2][1]=-cos_p*sin_t;
        transform[2][2]=0.0;
        transform[2][3]=0.0;
        transform[3][0]=0.0;
        transform[3][1]=0.0;
        transform[3][2]=0.0;
        transform[3][3]=1.0;
        mult(vertex,transform);
        display();
}

void isometric()        // first rotation with and then x
{
        float transform[4][4];
        float cos_t,sin_t,cos_p,sin_p;
        sin_t=1/(sqrt(3));
        cos_t=sqrt(1-(pow(sin_t,2)));

        sin_p=1/(sqrt(2));
        cos_p=sqrt(1-(pow(sin_p,2)));

        transform[0][0]=cos_p;
        transform[0][1]=sin_p*sin_t;
        transform[0][2]=0.0;
        transform[0][3]=0.0;
        transform[1][0]=0.0;
        transform[1][1]=cos_t;
        transform[1][2]=0.0;
        transform[1][3]=0.0;
        transform[2][0]=sin_p;
        transform[2][1]=-cos_p*sin_t;
        transform[2][2]=0.0;
        transform[2][3]=0.0;
        transform[3][0]=0.0;
        transform[3][1]=0.0;
        transform[3][2]=0.0;
        transform[3][3]=1.0;
        mult(vertex,transform);
        display();
}

void cavalier()
```

```
{
        float transform[4][4];
        float cos_a,sin_a,alpha;
        cout<<"enter the hotrizontal inclination angle";
        cin>>alpha;
        cos_a=cos((M_PI/180)*alpha);
        sin_a=sin((M_PI/180)*alpha);


        transform[0][0]=1.0;
        transform[0][1]=0.0;
        transform[0][2]=0.0;
        transform[0][3]=0.0;
        transform[1][0]=0.0;
        transform[1][1]=1.0;
        transform[1][2]=0.0;
        transform[1][3]=0.0;
        transform[2][0]=-cos_a;         // for cavlier f=1,i.e -fcos_a=-cos_a
        transform[2][1]=-sin_a;
        transform[2][2]=0.0;
        transform[2][3]=0.0;
        transform[3][0]=0.0;
        transform[3][1]=0.0;
        transform[3][2]=0.0;
        transform[3][3]=1.0;
        mult(vertex,transform);
        display();
}

void cabinet()
{
        float transform[4][4];
        float cos_a,sin_a,alpha;
        cout<<"enter the hotrizontal inclination angle";
        cin>>alpha;
        cos_a=cos((M_PI/180)*alpha);
        sin_a=sin((M_PI/180)*alpha);


        transform[0][0]=1.0;
        transform[0][1]=0.0;
        transform[0][2]=0.0;
        transform[0][3]=0.0;
        transform[1][0]=0.0;
```

```
        transform[1][1]=1.0;
        transform[1][2]=0.0;
        transform[1][3]=0.0;
        transform[2][0]=-(0.5)*cos_a; // for cavlier f=0.5,i.e -fcos_a=-(0.5)*cos_a
        transform[2][1]=-(0.5)*sin_a;
        transform[2][2]=0.0;
        transform[2][3]=0.0;
        transform[3][0]=0.0;
        transform[3][1]=0.0;
        transform[3][2]=0.0;
        transform[3][3]=1.0;
        mult(vertex,transform);
        display();
}

void single_point()
{
        float transform[4][4];
        float cos_a,sin_a,alpha,zc;
        cout<<"enter the value of centre of projection about z-axis";
        cin>>zc;
        transform[0][0]=1.0;
        transform[0][1]=0.0;
        transform[0][2]=0.0;
        transform[0][3]=0.0;
        transform[1][0]=0.0;
        transform[1][1]=1.0;
        transform[1][2]=0.0;
        transform[1][3]=0.0;
        transform[2][0]=0.0;
        transform[2][1]=0.0;
        transform[2][2]=1.0;
        transform[2][3]=-(1.0/zc);
        transform[3][0]=0.0;
        transform[3][1]=0.0;
        transform[3][2]=0.0;
        transform[3][3]=1.0;
        mult(vertex,transform);

        for(int i=0;i<8;i++)
                for(int j=0;j<4;j++)
                        final[i][j]=final[i][j]/(final[i][3]);

        display();
```

```
}

void two_point()
{
	float transform[4][4];
	float cos_a,sin_a,alpha,xp,yq,p,q;
	cout<<"enter the value of centre of projection about x-axis";
	cin>>xp;
	cout<<"enter the value of centre of projection about y-axis";
	cin>>yq;
	p=-1.0/xp;
	q=-1.0/yq;
	transform[0][0]=1.0;
	transform[0][1]=0.0;
	transform[0][2]=0.0;
	transform[0][3]=p;
	transform[1][0]=0.0;
	transform[1][1]=1.0;
	transform[1][2]=0.0;
	transform[1][3]=q;
	transform[2][0]=0.0;
	transform[2][1]=0.0;
	transform[2][2]=1.0;
	transform[2][3]=0.0;
	transform[3][0]=0.0;
	transform[3][1]=0.0;
	transform[3][2]=0.0;
	transform[3][3]=1.0;
	mult(vertex,transform);
	for(int i=0;i<8;i++)
		for(int j=0;j<4;j++)
			final[i][j]=final[i][j]/(final[i][3]);
	display();
}

void three_point()
{
	float transform[4][4];
	float cos_a,sin_a,alpha,xp,yq,zc,p,q,r;
	cout<<"enter the value of centre of projection about x-axis";
	cin>>xp;
	cout<<"enter the value of centre of projection about y-axis";
	cin>>yq;
	cout<<"enter the value of centre of projection about z-axis";
```

```
                cin>>zc;
                p=-1.0/xp;
                q=-1.0/yq;
                r=-1.0/zc;
                transform[0][0]=1.0;
                transform[0][1]=0.0;
                transform[0][2]=0.0;
                transform[0][3]=p;
                transform[1][0]=0.0;
                transform[1][1]=1.0;
                transform[1][2]=0.0;
                transform[1][3]=q;
                transform[2][0]=0.0;
                transform[2][1]=0.0;
                transform[2][2]=1.0;
                transform[2][3]=r;
                transform[3][0]=0.0;
                transform[3][1]=0.0;
                transform[3][2]=0.0;
                transform[3][3]=1.0;
                mult(vertex,transform);
                for(int i=0;i<8;i++)
                        for(int j=0;j<4;j++)
                                final[i][j]=final[i][j]/(final[i][3]);
                display();
        }

}cube;

void main()
{
        int gd=DETECT,gm,op;
        initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
        cube.input();
        while(1)
        {
                clrscr();
                cout<<"This Program shows different projection on unit cube\n\n\n";
                cout<<"********** MENU ***********";
                cout<<"\n\t1. ORTHOGRAPHIC";
                cout<<"\n\t2. TRIMETRIC";
                cout<<"\n\t3. DIMETRIC";
                cout<<"\n\t4. ISOMETRIC";
                cout<<"\n\t5. CAVALIER";
```

```
                cout<<"\n\t6. CABINET";
                cout<<"\n\t7. SINGLE-POINT";
                cout<<"\n\t8. TWO-POINT";
                cout<<"\n\t9. THREE-POINT";
                cout<<"\n\t10.EXIT";
                cout<<" \n\tGIVE UR CHOICE";
                cin>>op;
                switch(op)
                {
                        case 1:cube.orthographic();
                                break;
                        case 2: cube.trimetric();
                                break;
                        case 3: cube.dimetric();
                                break;
                        case 4: cube.isometric();
                                break;
                        case 5: cube.cavalier();
                                break;
                        case 6: cube.cabinet();
                                break;
                        case 7:cube.single_point();
                                break;
                        case 8: cube.two_point();
                                break;
                        case 9: cube.three_point();
                                break;
                        case 10:break;
                        default:cout<<"INVALID CHOICE";
                }
                if(op==10)
                        break;
                getch();
        }
        closegraph();
}
```