**Question 7**

**Write a program to apply a various 3D transformations on a 3D object and then apply parallel and perspective projection on it.**

#include<iostream>

#include<math.h>

#include<conio.h>

#include<graphics.h>

using namespace std;

class matrix

{

int nodes[10][4];

float T[4][4];

int size;

public:

matrix(int s)

{

size = s;

cout << "\nThe number of nodes are : \n" << size;

for(int i =0;i<size;++i)

nodes[i][3] = 1;

}

void input()

{

cout << endl;

for(int i=0;i<size;++i)

{

cout << "Enter P" << i<<" : ";

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

cin >> nodes[i][j];

}

cout << "\nBefore\n";

drawMy(nodes);

cout << endl;

}

void drawMy(int x[][4])

{

cout << "\nTransformation"<<endl;

for(int i=0;i<size;++i)

{

line(200 + x[i%size][0],200 + x[i%size][1], 200 + x[(i+1)%size][0],200 + x[(i+1)%size][1]);

}

cout << endl;

}

void rotation()

{

setcolor(RED);

for(int i=0;i<size;++i)

{

for(int j=0;j<4;++j)

{

cout << nodes[i][j] << " ";

}

cout << endl;

}

int angle;

cout << "Enter angle along x axis : ";

cin >> angle;

float r = angle\*(3.14/180);

float T[4][4];

T[0][0] = 1;

for (int i=1;i<=3;++i)

T[0][i] = 0;

for(int i=1;i<=3;++i)

T[i][0] = 0;

T[3][1] = 0;

T[3][2] = 0;

T[3][3] = 1;

T[0][3] = 0;

T[1][3] = 0;

T[2][3] = 0;

T[1][1] = cos(r);

T[1][2] = sin(r);

T[2][1] = -1\*sin(r);

T[2][2] = cos(r);

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

{

for(int j=0;j<4;++j)

{

cout << T[i][j] << " ";

}

cout << endl;

}

float rr[3][4];

int n = size;

cout << "\nSize is : " << n << endl;

for(int i=0;i<n;i++)

for(int j=0;j<4;j++)

rr[i][j]=0;

for(int i=0;i<size;i++)

{

for(int j=0;j<4;j++)

{

rr[i][j] = 0;

for(int k=0;k<4;k++)

rr[i][j] += nodes[i][k]\*T[k][j];

}

}

cout << endl;

for(int i=0;i<size;++i)

{

for(int j=0;j<4;++j)

cout << rr[i][j] << " ";

cout << endl;

}

for(int i=0;i<size;++i)

{

line(round(200 + rr[i%size][0]),round(200 + rr[i%size][1]), round(200 + rr[(i+1)%size][0]),round(200 + rr[(i+1)%size][1]));

}

}

};

int main()

{

int gdriver = DETECT, gmode;

initgraph(&gdriver, &gmode, "");

matrix m(3);

m.input();

m.rotation();

getch();

return 0;

}

**Output:**



