Computer Graphics Practicals

Assignment 7

U18CO081
Krunal Rank

Write programs for designing simple animations using 2D transformation Concepts.

1. Circle moving from left to right and vice versa
2. Wind mill rotation
3. Man walking
4. Simple animation of football goal

```
#include <bits/stdc++.h>
#include <GL/glut.h>
#include <chrono>
#include <thread>

using namespace std;

// To Compile:-
// g++ -pthread 1.cpp -lglfw3 -lGLEW -lGLU -lGL -lXrandr -lXxf86vm -lXi -lXinerama
-lX11 -lrt -ldl -lglut


void debug(vector<vector<double>>& points){
   for(auto i: points){
       for(auto j: i) cout<<j<<" ";
       cout<<endl;
   }
}

void plot(int x, int y)
{
   glBegin(GL_POINTS);
   glVertex2i(x,y);
   glEnd();
}


vector<vector<double>> matrix_mul(vector<vector<double>> &mat1,
vector<vector<double>> &mat2)
{
   int p = mat1.size();
   int q = mat1[0].size();
   int r = mat2[0].size();
   vector<vector<double>> mat3(p, vector<double>(q, 0));
```

```cpp
    for (int i = 0; i < p; i++)
    {
        for (int k = 0; k < r; k++)
        {
            for (int j = 0; j < q; j++)
            {
                mat3[i][k] += mat1[i][j] * mat2[j][k];
            }
        }
    }
    return mat3;
}



vector<vector<double>> translateMath(vector<vector<double>> &points,double tx,double
ty)
{
    vector<vector<double>> translateMatrix = {{1, 0, 0}, {0, 1, 0}, {(double)tx,
(double)ty, 1}};
    return matrix_mul(points, translateMatrix);
}

vector<vector<double>> rotateMath(vector<vector<double>> &points,double angle)
{
    vector<vector<double>> rotateMatrix = {{cos(angle), sin(angle), 0}, {-sin(angle),
cos(angle), 0}, {0, 0, 1}};
    return matrix_mul(points, rotateMatrix);
}



vector<vector<double>> reflectMath(vector<vector<double>> &points,int axis)
{
    vector<vector<double>> translateMatrix = {{axis==0?-1.0:1.0, 0, 0}, {0,
axis==1?-1.0:1.0, 0}, {0, 0, 1}};
    return matrix_mul(points, translateMatrix);
}

vector<vector<double>> getHomogeneousPoints(vector<vector<double>> &points)
{

    vector<vector<double>> homogeneousPoints(points.size(), vector<double>(3, 1));
    for (int i = 0; i < points.size(); i++)
        for (int j = 0; j < 2; j++)
            homogeneousPoints[i][j] = points[i][j];
    return homogeneousPoints;
```

```
}


vector<vector<double>> translate_util(vector<vector<double>>& points,double tx,double
ty)
{
   vector<vector<double>> translatePoints = getHomogeneousPoints(points);
   return translateMath(translatePoints,tx,ty);
}

vector<vector<double>> rotate_util(vector<vector<double>>& points,double angle)
{
   vector<vector<double>> rotatePoints = getHomogeneousPoints(points);
   return rotateMath(rotatePoints,angle);
}
vector<vector<double>> reflect_util(vector<vector<double>>& points,double axis)
{
   vector<vector<double>> reflectPoints = getHomogeneousPoints(points);
   return reflectMath(reflectPoints,axis);
}


vector<vector<double>> rotate_blade(vector<vector<double>>& points,double
angle,double cx,double cy){
   vector<vector<double>> tempVec = getHomogeneousPoints(points);
   tempVec = translate_util(tempVec,-cx,-cy);
   tempVec = rotate_util(tempVec,angle);
   return translate_util(tempVec,cx,cy);

}

void plotVector(vector<vector<double>> &points)
{

   glColor3f(0, 0, 1);
   for (int i = 0; i < points.size(); i++)
   {
       glBegin(GL_LINES);
       glVertex2f(points[i][0], points[i][1]);
       glVertex2f(points[(i + 1) % points.size()][0], points[(i + 1) %
points.size()][1]);
       glEnd();
   }
}
```

```
void midpoint_circle_util(int r,int cx,int cy){

   int y = 0, x = r;
   int p = 1 - r;

   while(x>=y){
       plot(cx+x,cy+y);
       plot(cx-x,cy+y);
       plot(cx+x,cy-y);
       plot(cx-x,cy-y);
       plot(cx+y,cy+x);
       plot(cx-y,cy+x);
       plot(cx+y,cy-x);
       plot(cx-y,cy-x);
       if(p<=0){
           p += 2*(y+1)+1;
       }else{
           p += 2*(y+1)+1-2*(x+1);
           x--;
       }
       y++;
   }
}

void task1(){

   int cx=400,cy=300,r = 100;
   int incr = -5;
   while(true){
       glClear(GL_COLOR_BUFFER_BIT);
       glPointSize(1.0);
       glColor3f(0,0,1);
       midpoint_circle_util(r,cx,cy);
       if(cx+r>800 || cx-r<0) incr = -incr;

       cx += incr;
       glutSwapBuffers();
       this_thread::sleep_for(chrono::milliseconds(30) );
       glFlush();
   }
}

void task2(){
   #define PI 3.141592653589793238
   vector<vector<double>> blade1 = {{400,300},{500,400},{600,300}};
   vector<vector<double>> blade2 = rotate_blade(blade1,-2.0*PI/3.0,400,300);
   vector<vector<double>> blade3 = rotate_blade(blade1,2.0*PI/3.0,400,300);
   vector<vector<double>> hinge = {{400,300},{350,50},{450,50}};
   while(true){
       glClear(GL_COLOR_BUFFER_BIT);
       glPointSize(1.0);
       glColor3f(0,0,1);
       plotVector(blade1);
       plotVector(blade2);
       plotVector(blade3);
       plotVector(hinge);
       blade1 = rotate_blade(blade1,-PI/180.0,400,300);
       blade2 = rotate_blade(blade2,-PI/180.0,400,300);
       blade3 = rotate_blade(blade3,-PI/180.0,400,300);
       glutSwapBuffers();
       this_thread::sleep_for(chrono::milliseconds(30) );
       glFlush();
   }
}

void task3(){
   vector<vector<double>> leftImage =
{{15,25},{16,25},{17,25},{18,25},{19,25},{13,24},{14,24},{15,24},{16,24},{17,24},{18,
24},{19,24},{20,24},{21,24},{11,23},{12,23},{13,23},{14,23},{15,23},{16,23},{17,23},{
18,23},{19,23},{20,23},{21,23},{22,23},{10,22},{11,22},{12,22},{13,22},{14,22},{15,22
},{16,22},{17,22},{18,22},{19,22},{20,22},{21,22},{22,22},{9,21},{10,21},{11,21},{12,
21},{13,21},{14,21},{15,21},{16,21},{17,21},{18,21},{19,21},{20,21},{21,21},{22,21},{
23,21},{8,20},{9,20},{10,20},{11,20},{12,20},{13,20},{14,20},{15,20},{16,20},{17,20},
{18,20},{19,20},{20,20},{21,20},{22,20},{23,20},{7,19},{8,19},{9,19},{10,19},{11,19},
{12,19},{13,19},{14,19},{15,19},{16,19},{17,19},{18,19},{19,19},{20,19},{21,19},{22,1
9},{23,19},{7,18},{8,18},{9,18},{10,18},{11,18},{12,18},{13,18},{17,18},{18,18},{19,1
8},{20,18},{21,18},{22,18},{23,18},{7,17},{8,17},{9,17},{10,17},{11,17},{12,17},{14,1
7},{15,17},{16,17},{18,17},{19,17},{20,17},{21,17},{22,17},{23,17},{8,16},{10,16},{13
,16},{17,16},{19,16},{20,16},{21,16},{22,16},{23,16},{8,15},{9,15},{10,15},{11,15},{1
3,15},{17,15},{19,15},{20,15},{21,15},{22,15},{9,14},{10,14},{11,14},{12,14},{14,14},
{15,14},{16,14},{17,14},{18,14},{19,14},{20,14},{21,14},{22,14},{10,13},{11,13},{12,1
3},{14,13},{15,13},{17,13},{18,13},{19,13},{20,13},{21,13},{22,13},{15,12},{18,12},{1
9,12},{20,12},{21,12},{22,12},{17,11},{18,11},{19,11},{20,11},{21,11},{13,10},{14,10}
,{15,10},{16,10},{19,10},{21,10},{13,9},{14,9},{17,9},{21,9},{22,9},{13,8},{14,8},{16
,8},{17,8},{20,8},{21,8},{11,7},{13,7},{14,7},{15,7},{16,7},{17,7},{21,7},{22,7},{11,
6},{12,6},{13,6},{14,6},{15,6},{16,6},{17,6},{18,6},{20,6},{21,6},{12,5},{15,5},{16,5
},{17,5},{18,5},{19,5},{20,5},{12,4},{13,4},{14,4},{15,4},{16,4},{17,4},{18,4},{19,4}
,{13,3},{14,3},{15,3},{16,3},{17,3},{18,3},{14,2},{15,2},{16,2},{17,2},};
```

```cpp
    vector<vector<double>> leftImage1 =
{{16,25},{17,25},{18,25},{19,25},{15,24},{16,24},{17,24},{18,24},{19,24},{20,24},{21,
24},{12,23},{13,23},{14,23},{15,23},{16,23},{17,23},{18,23},{19,23},{20,23},{21,23},{
22,23},{10,22},{11,22},{12,22},{13,22},{14,22},{15,22},{16,22},{17,22},{18,22},{19,22
},{20,22},{21,22},{22,22},{10,21},{11,21},{12,21},{13,21},{14,21},{15,21},{16,21},{17
,21},{18,21},{19,21},{20,21},{21,21},{22,21},{23,21},{9,20},{10,20},{11,20},{12,20},{
13,20},{14,20},{15,20},{16,20},{17,20},{18,20},{19,20},{20,20},{21,20},{22,20},{23,20
},{24,20},{8,19},{9,19},{10,19},{11,19},{12,19},{13,19},{14,19},{15,19},{16,19},{17,1
9},{18,19},{19,19},{20,19},{21,19},{22,19},{23,19},{24,19},{7,18},{8,18},{9,18},{10,1
8},{11,18},{12,18},{13,18},{14,18},{17,18},{18,18},{19,18},{20,18},{21,18},{22,18},{2
3,18},{24,18},{7,17},{8,17},{9,17},{10,17},{11,17},{12,17},{18,17},{19,17},{20,17},{2
1,17},{22,17},{23,17},{24,17},{7,16},{8,16},{9,16},{10,16},{11,16},{12,16},{14,16},{1
5,16},{16,16},{17,16},{18,16},{19,16},{20,16},{21,16},{22,16},{23,16},{24,16},{8,15},
{9,15},{10,15},{11,15},{13,15},{17,15},{19,15},{20,15},{21,15},{22,15},{23,15},{9,14}
,{10,14},{11,14},{12,14},{13,14},{15,14},{17,14},{18,14},{19,14},{20,14},{21,14},{22,
14},{23,14},{10,13},{11,13},{12,13},{13,13},{15,13},{17,13},{18,13},{19,13},{20,13},{
21,13},{22,13},{10,12},{12,12},{15,12},{16,12},{18,12},{19,12},{20,12},{21,12},{22,12
},{18,11},{19,11},{20,11},{21,11},{16,10},{17,10},{18,10},{19,10},{20,10},{21,10},{13
,9},{14,9},{15,9},{16,9},{17,9},{18,9},{21,9},{22,9},{14,8},{15,8},{16,8},{18,8},{21,
8},{22,8},{15,7},{16,7},{17,7},{18,7},{19,7},{21,7},{22,7},{12,6},{15,6},{16,6},{17,6
},{18,6},{19,6},{21,6},{9,5},{10,5},{12,5},{14,5},{15,5},{16,5},{18,5},{19,5},{20,5},
{21,5},{9,4},{10,4},{11,4},{12,4},{13,4},{14,4},{15,4},{16,4},{18,4},{19,4},{20,4},{2
1,4},{10,3},{11,3},{12,3},{13,3},{14,3},{15,3},{16,3},{17,3},{18,3},{19,3},{20,3},{11
,2},{12,2},{13,2},{18,2},{19,2},{20,2},};

    vector<vector<double>> leftImage2 =
{{14,25},{15,25},{16,25},{17,25},{18,25},{19,25},{20,25},{21,25},{12,24},{13,24},{14,
24},{15,24},{16,24},{17,24},{18,24},{19,24},{20,24},{21,24},{22,24},{10,23},{11,23},{
12,23},{13,23},{14,23},{15,23},{16,23},{17,23},{18,23},{19,23},{20,23},{21,23},{22,23
},{10,22},{11,22},{12,22},{13,22},{14,22},{15,22},{16,22},{17,22},{18,22},{19,22},{20
,22},{21,22},{22,22},{23,22},{9,21},{10,21},{11,21},{12,21},{13,21},{14,21},{15,21},{
16,21},{17,21},{18,21},{19,21},{20,21},{21,21},{22,21},{23,21},{24,21},{8,20},{9,20},
{10,20},{11,20},{12,20},{13,20},{14,20},{15,20},{16,20},{17,20},{18,20},{19,20},{20,2
0},{21,20},{22,20},{23,20},{24,20},{7,19},{8,19},{9,19},{10,19},{11,19},{12,19},{13,1
9},{14,19},{15,19},{16,19},{17,19},{18,19},{19,19},{20,19},{21,19},{22,19},{23,19},{2
4,19},{7,18},{8,18},{9,18},{10,18},{11,18},{12,18},{13,18},{18,18},{19,18},{20,18},{2
1,18},{22,18},{23,18},{24,18},{7,17},{8,17},{9,17},{10,17},{11,17},{12,17},{14,17},{1
5,17},{16,17},{18,17},{19,17},{20,17},{21,17},{22,17},{23,17},{24,17},{8,16},{9,16},{
10,16},{11,16},{13,16},{17,16},{18,16},{19,16},{20,16},{21,16},{22,16},{23,16},{24,16
},{9,15},{10,15},{11,15},{12,15},{14,15},{17,15},{18,15},{19,15},{20,15},{21,15},{22,
15},{23,15},{10,14},{11,14},{12,14},{13,14},{15,14},{16,14},{17,14},{18,14},{19,14},{
20,14},{21,14},{22,14},{10,13},{12,13},{15,13},{16,13},{18,13},{19,13},{20,13},{21,13
},{22,13},{18,12},{19,12},{20,12},{21,12},{17,11},{18,11},{19,11},{20,11},{21,11},{12
,10},{13,10},{14,10},{15,10},{16,10},{20,10},{21,10},{22,10},{10,9},{11,9},{13,9},{14
,9},{15,9},{16,9},{20,9},{21,9},{22,9},{13,8},{14,8},{15,8},{16,8},{19,8},{20,8},{21,
8},{22,8},{10,7},{12,7},{13,7},{14,7},{15,7},{16,7},{20,7},{21,7},{22,7},{10,6},{11,6
},{12,6},{14,6},{15,6},{16,6},{17,6},{18,6},{20,6},{21,6},{9,5},{10,5},{11,5},{14,5},
{15,5},{16,5},{17,5},{18,5},{19,5},{20,5},{21,5},{9,4},{10,4},{11,4},{12,4},{13,4},{1
4,4},{15,4},{16,4},{17,4},{18,4},{19,4},{20,4},{21,4},{10,3},{11,3},{12,3},{16,3},{17
,3},{19,3},{20,3},{17,2},{18,2},{19,2},};

   int cx = 400,cy = 300;

   vector<vector<double>> rightImage = reflect_util(leftImage,0);
   rightImage = translate_util(rightImage,32,0);
   vector<vector<double>> rightImage1 = reflect_util(leftImage1,0);
   rightImage1 = translate_util(rightImage1,32,0);
   vector<vector<double>> rightImage2 = reflect_util(leftImage2,0);
   rightImage2 = translate_util(rightImage2,32,0);

   leftImage = translate_util(leftImage,cx,cy);
   leftImage1 = translate_util(leftImage1,cx,cy);
   leftImage2 = translate_util(leftImage2,cx,cy);
   rightImage = translate_util(rightImage,cx,cy);
   rightImage1 = translate_util(rightImage1,cx,cy);
   rightImage2 = translate_util(rightImage2,cx,cy);
   int incr = 0;
   int dcx = -8;
   while(true){
       glClear(GL_COLOR_BUFFER_BIT);
       glPointSize(1.0);
       glColor3f(0,0,1);
       if(dcx<0){
           if(incr%2==0){
               plotVector(leftImage);
           }else if(incr%4==1){
               plotVector(leftImage1);
           }else{
               plotVector(leftImage2);
           }
       }else{
           if(incr%2==0){
               plotVector(rightImage);
           }else if(incr%4==1){
               plotVector(rightImage1);
           }else{
               plotVector(rightImage2);
           }

       }
```

```
        if(cx+dcx<0 || cx+dcx>800) dcx = -dcx;
        cx+= dcx;
        incr++;
        leftImage = translate_util(leftImage,dcx,0);
        leftImage1 = translate_util(leftImage1,dcx,0);
        leftImage2 = translate_util(leftImage2,dcx,0);
        rightImage = translate_util(rightImage,dcx,0);
        rightImage1 = translate_util(rightImage1,dcx,0);
        rightImage2 = translate_util(rightImage2,dcx,0);
        glutSwapBuffers();
        this_thread::sleep_for(chrono::milliseconds(256) );
        glFlush();
    }
}

void task4(){
    vector<vector<double>> path =
{{345,455},{346,455},{347,455},{348,455},{349,455},{350,455},{351,455},{352,455},{353
,455},{354,455},{355,455},{356,455},{357,455},{358,455},{359,455},{360,455},{361,455}
,{362,455},{363,455},{364,455},{365,455},{366,455},{367,455},{368,455},{369,455},{370
,455},{371,455},{372,455},{373,455},{374,455},{375,455},{376,455},{377,455},{378,455}
,{379,455},{380,455},{381,455},{382,455},{383,455},{384,455},{385,455},{386,455},{387
,455},{388,455},{389,455},{390,455},{391,455},{392,455},{393,455},{394,455},{395,455}
,{396,455},{397,455},{398,455},{399,455},{400,455},{401,455},{402,455},{403,455},{404
,455},{405,455},{406,455},{407,455},{408,455},{409,455},{410,455},{411,455},{412,455}
,{413,455},{414,455},{415,455},{416,455},{417,455},{418,455},{419,455},{420,455},{421
,455},{422,455},{423,455},{424,455},{425,455},{426,455},{427,455},{428,455},{429,455}
,{430,455},{431,455},{432,455},{433,455},{434,455},{435,455},{436,455},{437,455},{438
,455},{439,455},{440,455},{441,455},{442,455},{443,455},{444,455},{445,455},{446,455}
,{447,455},{448,455},{332,454},{333,454},{334,454},{335,454},{336,454},{337,454},{338
,454},{339,454},{340,454},{341,454},{342,454},{343,454},{344,454},{449,454},{450,454}
,{451,454},{452,454},{453,454},{454,454},{455,454},{315,453},{316,453},{317,453},{318
,453},{319,453},{320,453},{321,453},{322,453},{323,453},{324,453},{325,453},{326,453}
,{327,453},{328,453},{329,453},{330,453},{331,453},{456,453},{457,453},{458,453},{459
,453},{460,453},{461,453},{307,452},{308,452},{309,452},{310,452},{311,452},{312,452}
,{313,452},{314,452},{462,452},{463,452},{464,452},{465,452},{466,452},{467,452},{300
,451},{301,451},{302,451},{303,451},{304,451},{305,451},{306,451},{468,451},{469,451}
,{470,451},{471,451},{472,451},{295,450},{296,450},{297,450},{298,450},{299,450},{473
,450},{474,450},{475,450},{476,450},{477,450},{290,449},{291,449},{292,449},{293,449}
,{294,449},{478,449},{479,449},{480,449},{481,449},{286,448},{287,448},{288,448},{289
,448},{482,448},{483,448},{484,448},{485,448},{283,447},{284,447},{285,447},{486,447}
,{487,447},{488,447},{489,447},{279,446},{280,446},{281,446},{282,446},{490,446},{491
,446},{492,446},{493,446},{494,446},{276,445},{277,445},{278,445},{495,445},{496,445}
,{497,445},{498,445},{272,444},{273,444},{274,444},{275,444},{499,444},{500,444},{501
,444},{502,444},{269,443},{270,443},{271,443},{503,443},{504,443},{505,443},{506,443}
```

```
,{266,442},{267,442},{268,442},{507,442},{508,442},{509,442},{510,442},{263,441},{264
,441},{265,441},{511,441},{512,441},{513,441},{261,440},{262,440},{514,440},{515,440}
,{516,440},{517,440},{258,439},{259,439},{260,439},{518,439},{519,439},{520,439},{521
,439},{256,438},{257,438},{522,438},{523,438},{524,438},{253,437},{254,437},{255,437}
,{525,437},{526,437},{527,437},{250,436},{251,436},{252,436},{528,436},{529,436},{530
,436},{247,435},{248,435},{249,435},{531,435},{532,435},{533,435},{245,434},{246,434}
,{534,434},{535,434},{536,434},{242,433},{243,433},{244,433},{537,433},{538,433},{539
,433},{239,432},{240,432},{241,432},{540,432},{541,432},{236,431},{237,431},{238,431}
,{542,431},{543,431},{544,431},{233,430},{234,430},{235,430},{545,430},{546,430},{230
,429},{231,429},{232,429},{547,429},{548,429},{549,429},{228,428},{229,428},{550,428}
,{551,428},{225,427},{226,427},{227,427},{552,427},{223,426},{224,426},{553,426},{554
,426},{220,425},{221,425},{222,425},{555,425},{556,425},{218,424},{219,424},{557,424}
,{216,423},{217,423},{558,423},{559,423},{214,422},{215,422},{560,422},{212,421},{213
,421},{561,421},{562,421},{210,420},{211,420},{563,420},{208,419},{209,419},{564,419}
,{565,419},{205,418},{206,418},{207,418},{566,418},{567,418},{203,417},{204,417},{568
,417},{569,417},{201,416},{202,416},{570,416},{199,415},{200,415},{571,415},{572,415}
,{197,414},{198,414},{573,414},{574,414},{195,413},{196,413},{575,413},{192,412},{193
,412},{194,412},{576,412},{577,412},{190,411},{191,411},{578,411},{188,410},{189,410}
,{579,410},{580,410},{187,409},{581,409},{185,408},{186,408},{582,408},{583,408},{183
,407},{184,407},{584,407},{182,406},{585,406},{586,406},{180,405},{181,405},{587,405}
,{178,404},{179,404},{588,404},{589,404},{177,403},{590,403},{175,402},{176,402},{591
,402},{592,402},{174,401},{593,401},{173,400},{594,400},{595,400},{171,399},{172,399}
,{596,399},{170,398},{597,398},{598,398},{168,397},{169,397},{599,397},{600,397},{167
,396},{601,396},{166,395},{602,395},{603,395},{164,394},{165,394},{604,394},{605,394}
,{163,393},{606,393},{162,392},{607,392},{608,392},{160,391},{161,391},{609,391},{610
,391},{159,390},{611,390},{158,389},{612,389},{613,389},{156,388},{157,388},{614,388}
,{155,387},{615,387},{616,387},{154,386},{617,386},{618,386},{152,385},{153,385},{619
,385},{151,384},{620,384},{621,384},{150,383},{622,383},{148,382},{149,382},{623,382}
,{624,382},{147,381},{625,381},{146,380},{626,380},{627,380},{145,379},{628,379},{629
,379},{143,378},{144,378},{630,378},{142,377},{631,377},{632,377},{141,376},{633,376}
,{634,376},{140,375},{635,375},{138,374},{139,374},{636,374},{637,374},{137,373},{638
,373},{639,373},{136,372},{640,372},{641,372},{134,371},{135,371},{642,371},{643,371}
,{133,370},{644,370},{645,370},{131,369},{132,369},{646,369},{130,368},{647,368},{648
,368},{129,367},{649,367},{650,367},{127,366},{128,366},{651,366},{126,365},{652,365}
,{653,365},{125,364},{654,364},{655,364},{124,363},{656,363},{122,362},{123,362},{657
,362},{658,362},{121,361},{659,361},{660,361},{120,360},{661,360},{118,359},{119,359}
,{662,359},{663,359},{117,358},{664,358},{116,357},{665,357},{666,357},{115,356},{667
,356},{113,355},{114,355},{668,355},{112,354},{669,354},{670,354},{111,353},{671,353}
,{109,352},{110,352},{672,352},{673,352},{108,351},{674,351},{107,350},{675,350},{106
,349},{676,349},{677,349},{105,348},{678,348},{103,347},{104,347},{679,347},{680,347}
,{102,346},{681,346},{101,345},{682,345},{100,344},{683,344},{99,343},{684,343},{685,
343},{98,342},{686,342},{97,341},{687,341},{96,340},{688,340},{95,339},{689,339},{690
,339},{93,338},{94,338},{691,338},{92,337},{692,337},{693,337},{91,336},{694,336},{90
,335},{695,335},{89,334},{696,334},{697,334},{88,333},{698,333},{87,332},{699,332},{8
5,331},{86,331},{700,331},{701,331},{84,330},{702,330},{83,329},{703,329},{82,328},{7
04,328},{705,328},{81,327},{706,327},{80,326},{707,326},{79,325},{708,325},{709,325},
{78,324},{710,324},{77,323},{711,323},{76,322},{712,322},{713,322},{74,321},{75,321},
{714,321},{73,320},{715,320},{73,319},{716,319},{71,318},{72,318},{717,318},{718,318}
,{70,317},{719,317},{69,316},{720,316},{721,316},{68,315},{722,315},{67,314},{723,314
},{724,314},{66,313},{725,313},{65,312},{726,312},{727,312},{64,311},{728,311},{63,31
0},{729,310},{730,310},{62,309},{731,309},{61,308},{732,308},{60,307},{733,307},{734,
307},{59,306},{735,306},{58,305},{736,305},{737,305},{57,304},{738,304},{56,303},{739
,303},{55,302},{740,302},{54,301},{741,301},{742,301},{53,300},{743,300},{52,299},{74
4,299},{51,298},{745,298},{50,297},{746,297},{49,296},{747,296},{48,295},{748,295},{4
7,294},{749,294},{46,293},{750,293},{44,292},{45,292},{751,292},{43,291},{752,291},{4
2,290},{753,290},{41,289},{754,289},{39,288},{40,288},{755,288},{38,287},{756,287},{3
7,286},{757,286},{36,285},{758,285},{34,284},{35,284},{759,284},{760,284},{33,283},{7
61,283},{31,282},{32,282},{761,282},{30,281},{31,281},{762,281},{763,281},{29,280},{7
64,280},{28,279},{765,279},{766,278},{767,277},{768,276},{769,275},{770,274},{771,273
},{771,272},{771,271},{772,270}};


   sort(path.begin(),path.end());

   int cx = path[0][0],cy=path[0][1],r=20;

   vector<vector<double>> net = {{650,500},{794,400},{794,200},{650,100}};
   int idx = 0;
   while(true){
       glClearColor(0 ,1 ,0.5 , 1);
       glClear(GL_COLOR_BUFFER_BIT);
       glPointSize(1.0);
       glColor3f(0,0,1);
       midpoint_circle_util(r,cx,cy);
       plotVector(net);
       glutSwapBuffers();
       idx+=2;
       this_thread::sleep_for(chrono::milliseconds(30) );

       cx = path[idx%path.size()][0];
       cy = path[idx%path.size()][1];
       glFlush();
   }

}

void init(int argc,char** argv){
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
   glutInitWindowSize(800, 600);
```

```
    glutCreateWindow("Assignment 7");
    glShadeModel(GLU_FLAT);
    glClearColor(1.0, 1.0, 1.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0, 800.0, 0.0, 600.0);
}




int main(int argc, char *argv[]){
    cout<<"Question 1:"<<endl;
    cout<<"1. Circle Moving From Left to Right and Vice Versa"<<endl;
    cout<<"2. Wind mill rotation"<<endl;
    cout<<"3. Man walking"<<endl;
    cout<<"4. Simple animation of football goal"<<endl;
    cout<<">>";
    int choice;
    cin >> choice;
    init(argc,argv);

    if(choice==1){
        glutDisplayFunc(task1);
    }else if(choice==2){
        glutDisplayFunc(task2);
    }else if(choice==3){
        glutDisplayFunc(task3);
    }else if(choice==4){
        glutDisplayFunc(task4);
    }else{
        cout<<"Invalid Choice!"<<endl;
        return 0;
    }
    glutMainLoop();
}
```

2. Write a menu driven Program to implement set of basic Transformations on Polygon:
Program should include: Translation, Rotation and Scaling.

```
#include <bits/stdc++.h>
#include <GL/glut.h>

using namespace std;

// To Compile:-
// g++ -pthread 2.cpp -lglfw3 -lGLEW -lGLU -lGL -lXrandr -lXxf86vm -lXi -lXinerama
-lX11 -lrt -ldl -lglut

int n;
vector<vector<double>> points;
int choice;
double angle, tx, ty, s, shear;
int isx;
double axis;
vector<vector<vector<double>>> displayPoints;
void debug(){
   for(auto i: points){
       for(auto j: i) cout<<j<<" ";
       cout<<endl;
   }
}

vector<vector<double>> matrix_mul(vector<vector<double>> &mat1,
vector<vector<double>> &mat2)
{
   int p = mat1.size();
   int q = mat1[0].size();
   int r = mat2[0].size();
   vector<vector<double>> mat3(p, vector<double>(q, 0));
   for (int i = 0; i < p; i++)
   {
       for (int k = 0; k < r; k++)
       {
           for (int j = 0; j < q; j++)
           {
               mat3[i][k] += mat1[i][j] * mat2[j][k];
           }
       }
   }
   return mat3;
```

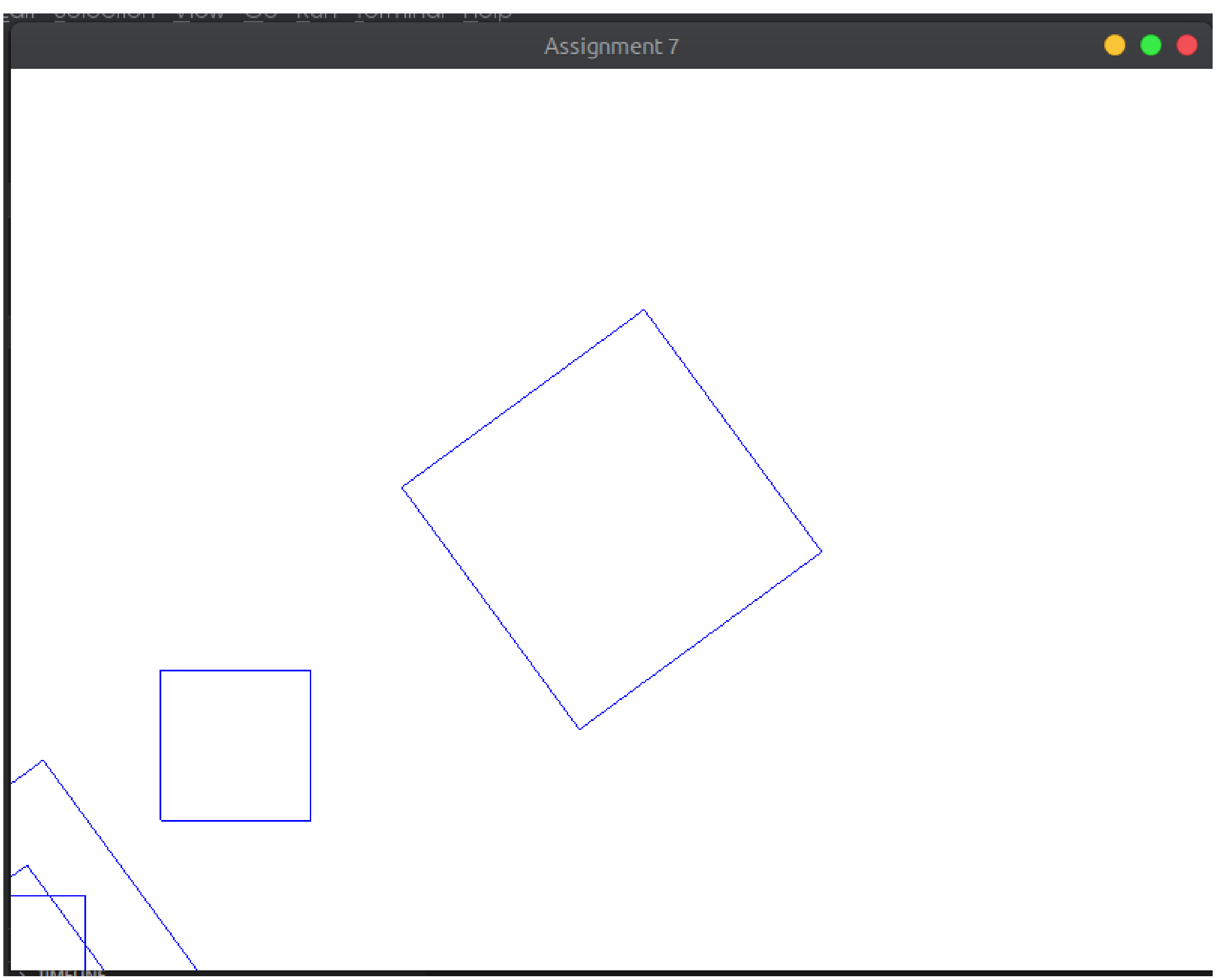
```
}

vector<vector<double>> translateMath(vector<vector<double>> &points, int tX, int tY)
{
   vector<vector<double>> translateMatrix = {{1, 0, 0}, {0, 1, 0}, {(double)tX,
(double)tY, 1}};
   return matrix_mul(points, translateMatrix);
}

vector<vector<double>> rotateMath(vector<vector<double>> &points, double angle)
{
   vector<vector<double>> rotateMatrix = {{cos(angle), sin(angle), 0}, {-sin(angle),
cos(angle), 0}, {0, 0, 1}};
   return matrix_mul(points, rotateMatrix);
}

vector<vector<double>> scaleMath(vector<vector<double>> &points, double scale)
{
   vector<vector<double>> scaleMatrix = {{scale, 0, 0}, {0, scale, 0}, {0, 0, 1}};
   return matrix_mul(points, scaleMatrix);
}

vector<vector<double>> shearMath(vector<vector<double>> &points, double shear, bool
isX)
{
   vector<vector<double>> scaleMatrix = {{1.0, isX == false ? shear : 0, 0}, {isX ==
true ? shear : 0, 1, 0}, {0, 0, 1}};
   return matrix_mul(points, scaleMatrix);
}


vector<vector<double>> reflectMath(vector<vector<double>> &points,int axis)
{
   vector<vector<double>> translateMatrix = {{axis==0?-1.0:1.0, 0, 0}, {0,
axis==1?-1.0:1.0, 0}, {0, 0, 1}};
   return matrix_mul(points, translateMatrix);
}

vector<vector<double>> getHomogeneousPoints(vector<vector<double>> &points)
{

   vector<vector<double>> homogeneousPoints(n, vector<double>(3, 1));
   for (int i = 0; i < n; i++)
       for (int j = 0; j < 2; j++)
           homogeneousPoints[i][j] = points[i][j];
```

```
    return homogeneousPoints;
}

void plotVector(vector<vector<double>> &points)
{


    glColor3f(0, 0, 1);
    for (int i = 0; i < n; i++)
    {
        glBegin(GL_LINES);
        glVertex2f(points[i][0], points[i][1]);
        glVertex2f(points[(i + 1) % n][0], points[(i + 1) % n][1]);
        glEnd();
    }
}

void normal_util()
{
    plotVector(points);
}

void translate_util()
{
    vector<vector<double>> translatePoints = getHomogeneousPoints(points);
    points = translateMath(translatePoints, tx, ty);
}

void rotate_util()
{
    vector<vector<double>> rotatePoints = getHomogeneousPoints(points);
    points = rotateMath(rotatePoints, angle);
}

void scale_util()
{
    vector<vector<double>> scalePoints = getHomogeneousPoints(points);
    points = scaleMath(scalePoints, s);
}

void shear_util()
{
    vector<vector<double>> scalePoints = getHomogeneousPoints(points);
    points = shearMath(scalePoints, shear, isx);
}
```

```
void reflect_util()
{
   vector<vector<double>> reflectPoints = getHomogeneousPoints(points);
   points = reflectMath(reflectPoints,axis);
}


void display()
{
   glClear(GL_COLOR_BUFFER_BIT);
   glPointSize(1.0);
   glColor3f(0, 0, 1);
   for(auto points: displayPoints)
       plotVector(points);
   glutSwapBuffers();
}


int main(int argc, char *argv[])
{
   cout << "Enter Number of Points: ";
   cin >> n;
   points = vector<vector<double>>(n, vector<double>(2, 0));
   for (int i = 0; i < n; i++)
   {
       cout << "Enter point[" << i << "].x: ";
       cin >> points[i][0];
       cout << "Enter point[" << i << "].y: ";
       cin >> points[i][1];
   }

   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
   glutInitWindowSize(800, 600);
   glutCreateWindow("Assignment 7");
   glShadeModel(GLU_FLAT);
   glClearColor(1.0, 1.0, 1.0, 0.0);
   glMatrixMode(GL_PROJECTION);
   gluOrtho2D(0.0, 800.0, 0.0, 600.0);
   while (true)
   {
       displayPoints.push_back(points);
       cout << "Enter Choice:\n1. Translate\n2. Rotate\n3. Scale\n4. Shear\n5.
Reflect\n6. Display\n>>";
```

```cpp
    cin >> choice;
    cout<<choice<<endl;
    if (choice == 1)
    {
        cout << "TX: ";
        cin >> tx;
        cout << "TY: ";
        cin >> ty;
        translate_util();
    }
    else if (choice == 2)
    {
        cout << "Angle: ";
        cin >> angle;
        angle *= 180/(3.14159358979);
        rotate_util();
    }
    else if (choice == 3)
    {
        cout << "Scale: ";
        cin >> s;
        scale_util();
    }
    else if (choice == 4)
    {
        cout << "Shear: ";
        cin >> shear;
        cout << "1 if X Shear, 0 otherwise: ";
        cin >> isx;
        shear_util();
    }else if (choice == 5)
    {
        cout << "Reflecting Axis (0 if X,otherwise 1): ";
        cin >> axis;
        reflect_util();
    }
    else if (choice == 6)
    {
        break;
    }
    else{
        cout << "Invalid Choice!" << endl;
    }
}
```

```
    glutDisplayFunc(display);

    glutMainLoop();
}
```


Assignment 7