#include <cstdlib> // standard definitions

#include <iostream> // C++ I/O

#include <cstdio> // C I/O (for sprintf)

#include <cmath> // standard definitions

#include <Windows.h>

#include <MMSystem.h>

#include <GL/glut.h> // GLUT

#include <GL/glu.h> // GLU

#include <GL/gl.h> // OpenGL

#include <process.h>

using namespace std; // make std accessible

//-----------------------------------------------------------------------

// Global data

//-----------------------------------------------------------------------

//GLint TIMER\_DELAY = 10000; // timer delay (10 seconds)

GLint TIMER\_DELAY = 10;

int sure\_koyun=0;

GLfloat RED\_RGB[] = {1.0, 0.0, 0.0}; // drawing colors

GLfloat BLUE\_RGB[] = {0.0, 0.0, 1.0};

int mx=0;

int my=0;

int m1;

int m2;

double kopek\_a=0.0;

double kopek\_b=1.0;

double kurt\_a1=0.1;

double kurt\_b1=0.6;

double kurt\_hizi=0.3;

double kurt\_a2=0.4;

double kurt\_b2=0.7;

int kontrol=0;

double yer\_degisikligi = 0.01;

double kuzu1\_x = 0.05; double kuzu1\_y = 0.05;

double kuzu2\_x = 0.25; double kuzu2\_y = 0.95;

double kuzu3\_x = 0.45; double kuzu3\_y = 0.25;

double kuzu4\_x = 0.65; double kuzu4\_y = 0.65;

double kuzu5\_x = 0.85; double kuzu5\_y = 0.45;

double kuzu6\_x = 0.95; double kuzu6\_y = 0.85;

double rasgele1\_x = 0.05; double rasgele1\_y = 0.05;

double rasgele2\_x = 0.25; double rasgele2\_y = 0.95;

double rasgele3\_x = 0.45; double rasgele3\_y = 0.25;

double rasgele4\_x = 0.65; double rasgele4\_y = 0.65;

double rasgele5\_x = 0.85; double rasgele5\_y = 0.45;

double rasgele6\_x = 0.95; double rasgele6\_y = 0.85;

double koyun\_hizi\_1=0.2;

double koyun\_hizi\_2=0.2;

double koyun\_hizi\_3=0.2;

double koyun\_hizi\_4=0.2;

double koyun\_hizi\_5=0.2;

double koyun\_hizi\_6=0.2;

double kuzu\_koordinat[6][2];

int koyun\_yak=50;

int ses\_yak=200;

double gunes= 0.98;

double renk= 0.5;

int dizi[10000][2] = {0};

int sayac=0;

int sayac2=0;

double zaman\_tut=0.0;

bool mouseleftdown = false;

bool kopek\_hav = false;

GLvoid \*font\_style = GLUT\_BITMAP\_HELVETICA\_18;

//-----------------------------------------------------------------------

// Global variables

//-----------------------------------------------------------------------

static bool isReversed = false; // draw reversed colors?

void myReshape(int w, int h) {

glViewport (0, 0, w, h); // update the viewport

m1=w;

m2=h;

glMatrixMode(GL\_PROJECTION); // update projection

glLoadIdentity();

gluOrtho2D(0.0, 1.0, 0.0, 1.0); // map unit square to viewport

glMatrixMode(GL\_MODELVIEW);

//glutPostRedisplay(); // request redisplay

}

unsigned \_\_stdcall dog(void \*data)

{

PlaySound(TEXT("D:/dog1.wav"), NULL, SND\_FILENAME);

return 0;

}

unsigned \_\_stdcall sheep(void \*data)

{

PlaySound(TEXT("D:/sheep1.wav"), NULL, SND\_FILENAME);

return 0;

}

// draw diamond and rectangle

void drawObjects() {

sure\_koyun +=TIMER\_DELAY;

if(fmod((double)sure\_koyun,2000.0) == 0.0)

{

\_beginthreadex(NULL, 0, sheep, NULL, 0, NULL);

}

glBegin(GL\_POLYGON); // ahır

glColor3d(0.1,0.4,0.1);

glVertex2d(0.6, 0.5);

glVertex2d(0.6, 1.0);

glVertex2d(1.0, 1.0);

glVertex2d(1.0, 0.5);

glEnd();

if(gunes > 0.02)gunes-=(double)1/(7500);

else exit(0);

glPointSize(30); // güneş

glBegin(GL\_POINTS);

glColor3d(1,renk,0);

if(gunes<0.5)renk-=(double)1/(1875);

else renk+=(double)1/(1875);

glVertex2d(gunes , 0.98);

glEnd();

glColor3d(0,0,0);

glRasterPos2d (gunes-0.01 , 0.98-0.01);

glutBitmapCharacter(font\_style,'G');

int kuzu\_sayac=6;

for(int w=0 ; w<6;w++)

if(kuzu\_koordinat[w][0]==-2)

--kuzu\_sayac;

cout<< kuzu\_sayac<<endl;

glColor3d(0,0,0);

glRasterPos2d (0.05 , 0.97);

glutBitmapCharacter(font\_style,'K');

glRasterPos2d (0.07 , 0.97);

glutBitmapCharacter(font\_style,'A');

glRasterPos2d (0.09 , 0.97);

glutBitmapCharacter(font\_style,'L');

glRasterPos2d (0.11 , 0.97);

glutBitmapCharacter(font\_style,'A');

glRasterPos2d (0.13 , 0.97);

glutBitmapCharacter(font\_style,'N');

glRasterPos2d (0.15 , 0.97);

glutBitmapCharacter(font\_style,':');

if(kuzu\_sayac == 0)

{

glRasterPos2d (0.17, 0.97);

glutBitmapCharacter(font\_style,'0');

}

if(kuzu\_sayac == 1)

{

glRasterPos2d (0.17, 0.97);

glutBitmapCharacter(font\_style,'1');

}

if(kuzu\_sayac == 2)

{

glRasterPos2d (0.17, 0.97);

glutBitmapCharacter(font\_style,'2');

}

if(kuzu\_sayac == 3)

{

glRasterPos2d (0.17, 0.97);

glutBitmapCharacter(font\_style,'3');

}

if(kuzu\_sayac == 4)

{

glRasterPos2d (0.17, 0.97);

glutBitmapCharacter(font\_style,'4');

}

if(kuzu\_sayac == 5)

{

glRasterPos2d (0.17, 0.97);

glutBitmapCharacter(font\_style,'5');

}

if(kuzu\_sayac == 6)

{

glRasterPos2d (0.17, 0.97);

glutBitmapCharacter(font\_style,'6');

}

glPointSize(50); // hav hav butonu

glBegin(GL\_POINTS);

glColor3d(0.6,0.6,0.6);

glVertex2d((double)(m1-30)/m1 , 1-(double)(m2-30)/m2);

glEnd();

glColor3d(0,0,0);

glRasterPos2d ((double)(m1-30)/m1-0.01 , 1-(double)(m2-30)/m2-0.01);

glutBitmapCharacter(font\_style,'H');

//kuzular

if(kuzu1\_x >= 0.95 || kuzu1\_x <= 0.05 || kuzu1\_y >= 0.95 || kuzu1\_y <= 0.05 ) //kuzu1

{

rasgele1\_x = ( rand() % 11 ) \* 0.1;

rasgele1\_y = ( rand() % 11 ) \* 0.1;

koyun\_hizi\_1 = 0.1;

}

if(fabs(rasgele1\_x-kuzu1\_x) < (double)3/m1 && fabs(rasgele1\_y-kuzu1\_y) < (double)3/m1)

{

rasgele1\_x = ( rand() % 11 ) \* 0.1;

rasgele1\_y = ( rand() % 11 ) \* 0.1;

koyun\_hizi\_1 = 0.1;

}

if(fabs(kopek\_a-kuzu1\_x) < (double)koyun\_yak/m1 && fabs(kopek\_b-kuzu1\_y) < (double)koyun\_yak/m2)

{

rasgele1\_x = (kuzu1\_x-kopek\_a) + kuzu1\_x;

rasgele1\_y = (kuzu1\_y-kopek\_b) + kuzu1\_y;

koyun\_hizi\_1 = 1;

}

if( ( fabs(kopek\_a-kuzu1\_x) < (double)ses\_yak/m1 && fabs(kopek\_b-kuzu1\_y) < (double)ses\_yak/m2 ) && kopek\_hav )

{

rasgele1\_x = (kuzu1\_x-kopek\_a) + kuzu1\_x;

rasgele1\_y = (kuzu1\_y-kopek\_b) + kuzu1\_y;

koyun\_hizi\_1 = 1;

/\*kopek\_hav=false;\*/

}

if(kuzu1\_x < rasgele1\_x)

{

kuzu1\_x += (double)koyun\_hizi\_1/m1;

}

if(kuzu1\_x > rasgele1\_x)

{

kuzu1\_x -= (double)koyun\_hizi\_1/m1;

}

if(kuzu1\_y < rasgele1\_y)

{

kuzu1\_y += (double)koyun\_hizi\_1/m2;

}

if(kuzu1\_y > rasgele1\_y)

{

kuzu1\_y -= (double)koyun\_hizi\_1/m2;

}

if(kuzu2\_x >= 0.95 || kuzu2\_x <= 0.05 || kuzu2\_y >= 0.95 || kuzu2\_y <= 0.05 ) //kuzu2

{

rasgele2\_x = ( rand() % 11 ) \* 0.1;

rasgele2\_y = ( rand() % 11 ) \* 0.1;

koyun\_hizi\_2 = 0.1;

}

if(fabs(rasgele2\_x-kuzu2\_x) < (double)3/m1 && fabs(rasgele2\_y-kuzu2\_y) < (double)3/m1)

{

rasgele2\_x = ( rand() % 11 ) \* 0.1;

rasgele2\_y = ( rand() % 11 ) \* 0.1;

koyun\_hizi\_2 = 0.1;

}

if(fabs(kopek\_a-kuzu2\_x) < (double)koyun\_yak/m1 && fabs(kopek\_b-kuzu2\_y) < (double)koyun\_yak/m2)

{

rasgele2\_x = (kuzu2\_x-kopek\_a) + kuzu2\_x;

rasgele2\_y = (kuzu2\_y-kopek\_b) + kuzu2\_y;

koyun\_hizi\_2 = 1;

}

if( (fabs(kopek\_a-kuzu2\_x) < (double)ses\_yak/m1 && fabs(kopek\_b-kuzu2\_y) < (double)ses\_yak/m2 ) && kopek\_hav)

{

rasgele2\_x = (kuzu2\_x-kopek\_a) + kuzu2\_x;

rasgele2\_y = (kuzu2\_y-kopek\_b) + kuzu2\_y;

koyun\_hizi\_2 = 1;

//kopek\_hav=false;

}

if(kuzu2\_x < rasgele2\_x)

{

kuzu2\_x += (double)koyun\_hizi\_2/m1;

}

if(kuzu2\_x > rasgele2\_x)

{

kuzu2\_x -= (double)koyun\_hizi\_2/m1;

}

if(kuzu2\_y < rasgele2\_y)

{

kuzu2\_y += (double)koyun\_hizi\_2/m2;

}

if(kuzu2\_y > rasgele2\_y)

{

kuzu2\_y -= (double)koyun\_hizi\_2/m2;

}

if(kuzu3\_x >= 0.95 || kuzu3\_x <= 0.05 || kuzu3\_y >= 0.95 || kuzu3\_y <= 0.05 ) //kuzu3

{

rasgele3\_x = ( rand() % 11 ) \* 0.1;

rasgele3\_y = ( rand() % 11 ) \* 0.1;

koyun\_hizi\_3 = 0.1;

}

if(fabs(rasgele3\_x-kuzu3\_x) < (double)3/m1 && fabs(rasgele3\_y-kuzu3\_y) < (double)3/m1)

{

rasgele3\_x = ( rand() % 11 ) \* 0.1;

rasgele3\_y = ( rand() % 11 ) \* 0.1;

koyun\_hizi\_3 = 0.1;

}

if(fabs(kopek\_a-kuzu3\_x) < (double)koyun\_yak/m1 && fabs(kopek\_b-kuzu3\_y) < (double)koyun\_yak/m2)

{

rasgele3\_x = (kuzu3\_x-kopek\_a) + kuzu3\_x;

rasgele3\_y = (kuzu3\_y-kopek\_b) + kuzu3\_y;

koyun\_hizi\_3 = 1;

}

if( ( fabs(kopek\_a-kuzu3\_x) < (double)ses\_yak/m1 && fabs(kopek\_b-kuzu3\_y) < (double)ses\_yak/m2 ) && kopek\_hav)

{

rasgele3\_x = (kuzu3\_x-kopek\_a) + kuzu3\_x;

rasgele3\_y = (kuzu3\_y-kopek\_b) + kuzu3\_y;

koyun\_hizi\_3 = 1;

//kopek\_hav = false;

}

if(kuzu3\_x < rasgele3\_x)

{

kuzu3\_x += (double)koyun\_hizi\_3/m1;

}

if(kuzu3\_x > rasgele3\_x)

{

kuzu3\_x -= (double)koyun\_hizi\_3/m1;

}

if(kuzu3\_y < rasgele3\_y)

{

kuzu3\_y += (double)koyun\_hizi\_3/m2;

}

if(kuzu3\_y > rasgele3\_y)

{

kuzu3\_y -= (double)koyun\_hizi\_3/m2;

}

if(kuzu4\_x >= 0.95 || kuzu4\_x <= 0.05 || kuzu4\_y >= 0.95 || kuzu4\_y <= 0.05 ) //kuzu4

{

rasgele4\_x = ( rand() % 11 ) \* 0.1;

rasgele4\_y = ( rand() % 11 ) \* 0.1;

koyun\_hizi\_4 = 0.1;

}

if(fabs(rasgele4\_x-kuzu4\_x) < (double)3/m1 && fabs(rasgele4\_y-kuzu4\_y) < (double)3/m1)

{

rasgele4\_x = ( rand() % 11 ) \* 0.1;

rasgele4\_y = ( rand() % 11 ) \* 0.1;

koyun\_hizi\_4 = 0.1;

}

if(fabs(kopek\_a-kuzu4\_x) < (double)koyun\_yak/m1 && fabs(kopek\_b-kuzu4\_y) < (double)koyun\_yak/m2)

{

rasgele4\_x = (kuzu4\_x-kopek\_a) + kuzu4\_x;

rasgele4\_y = (kuzu4\_y-kopek\_b) + kuzu4\_y;

koyun\_hizi\_4 = 1;

}

if( ( fabs(kopek\_a-kuzu4\_x) < (double)ses\_yak/m1 && fabs(kopek\_b-kuzu4\_y) < (double)ses\_yak/m2 ) && kopek\_hav)

{

rasgele4\_x = (kuzu4\_x-kopek\_a) + kuzu4\_x;

rasgele4\_y = (kuzu4\_y-kopek\_b) + kuzu4\_y;

koyun\_hizi\_4 = 1;

//kopek\_hav = false;

}

if(kuzu4\_x < rasgele4\_x)

{

kuzu4\_x += (double)koyun\_hizi\_4/m1;

}

if(kuzu4\_x > rasgele4\_x)

{

kuzu4\_x -= (double)koyun\_hizi\_4/m1;

}

if(kuzu4\_y < rasgele4\_y)

{

kuzu4\_y += (double)koyun\_hizi\_4/m2;

}

if(kuzu4\_y > rasgele4\_y)

{

kuzu4\_y -= (double)koyun\_hizi\_4/m2;

}

if(kuzu5\_x >= 0.95 || kuzu5\_x <= 0.05 || kuzu5\_y >= 0.95 || kuzu5\_y <= 0.05 ) //kuzu5

{

rasgele5\_x = ( rand() % 11 ) \* 0.1;

rasgele5\_y = ( rand() % 11 ) \* 0.1;

koyun\_hizi\_5 = 0.1;

}

if(fabs(rasgele5\_x-kuzu5\_x) < (double)3/m1 && fabs(rasgele5\_y-kuzu5\_y) < (double)3/m1)

{

rasgele5\_x = ( rand() % 11 ) \* 0.1;

rasgele5\_y = ( rand() % 11 ) \* 0.1;

koyun\_hizi\_5 = 0.1;

}

if(fabs(kopek\_a-kuzu5\_x) < (double)koyun\_yak/m1 && fabs(kopek\_b-kuzu5\_y) < (double)koyun\_yak/m2)

{

rasgele5\_x = (kuzu5\_x-kopek\_a) + kuzu5\_x;

rasgele5\_y = (kuzu5\_y-kopek\_b) + kuzu5\_y;

koyun\_hizi\_5 = 1;

}

if( ( fabs(kopek\_a-kuzu5\_x) < (double)ses\_yak/m1 && fabs(kopek\_b-kuzu5\_y) < (double)ses\_yak/m2 ) && kopek\_hav)

{

rasgele5\_x = (kuzu5\_x-kopek\_a) + kuzu5\_x;

rasgele5\_y = (kuzu5\_y-kopek\_b) + kuzu5\_y;

koyun\_hizi\_5 = 1;

//kopek\_hav= false;

}

if(kuzu5\_x < rasgele5\_x)

{

kuzu5\_x += (double)koyun\_hizi\_5/m1;

}

if(kuzu5\_x > rasgele5\_x)

{

kuzu5\_x -= (double)koyun\_hizi\_5/m1;

}

if(kuzu5\_y < rasgele5\_y)

{

kuzu5\_y += (double)koyun\_hizi\_5/m2;

}

if(kuzu5\_y > rasgele5\_y)

{

kuzu5\_y -= (double)koyun\_hizi\_5/m2;

}

if(kuzu6\_x >= 0.95 || kuzu6\_x <= 0.05 || kuzu6\_y >= 0.95 || kuzu6\_y <= 0.05 ) //kuzu6

{

rasgele6\_x = ( rand() % 11 ) \* 0.1;

rasgele6\_y = ( rand() % 11 ) \* 0.1;

koyun\_hizi\_6 = 0.1;

}

if(fabs(rasgele6\_x-kuzu6\_x) < (double)3/m1 && fabs(rasgele6\_y-kuzu6\_y) < (double)3/m1)

{

rasgele6\_x = ( rand() % 11 ) \* 0.1;

rasgele6\_y = ( rand() % 11 ) \* 0.1;

koyun\_hizi\_6 = 0.1;

}

if(fabs(kopek\_a-kuzu6\_x) < (double)koyun\_yak/m1 && fabs(kopek\_b-kuzu6\_y) < (double)koyun\_yak/m2)

{

rasgele6\_x = (kuzu6\_x-kopek\_a) + kuzu6\_x;

rasgele6\_y = (kuzu6\_y-kopek\_b) + kuzu6\_y;

koyun\_hizi\_6 = 1;

}

if( ( fabs(kopek\_a-kuzu6\_x) < (double)ses\_yak/m1 && fabs(kopek\_b-kuzu6\_y) < (double)ses\_yak/m2 ) && kopek\_hav)

{

rasgele6\_x = (kuzu6\_x-kopek\_a) + kuzu6\_x;

rasgele6\_y = (kuzu6\_y-kopek\_b) + kuzu6\_y;

koyun\_hizi\_6 = 1;

//kopek\_hav = false;

}

if(kuzu6\_x < rasgele6\_x)

{

kuzu6\_x += (double)koyun\_hizi\_6/m1;

}

if(kuzu6\_x > rasgele6\_x)

{

kuzu6\_x -= (double)koyun\_hizi\_6/m1;

}

if(kuzu6\_y < rasgele6\_y)

{

kuzu6\_y += (double)koyun\_hizi\_6/m2;

}

if(kuzu6\_y > rasgele6\_y)

{

kuzu6\_y -= (double)koyun\_hizi\_6/m2;

}

kopek\_hav = false;

glPointSize(20);

glBegin(GL\_POINTS);

glColor3d(1.0,1.0,1.0);

if(kuzu\_koordinat[0][0] != -2)

{

glVertex2d(kuzu1\_x,kuzu1\_y);

}

if(kuzu\_koordinat[1][0] != -2)

{

glVertex2d(kuzu2\_x,kuzu2\_y);

}

if(kuzu\_koordinat[2][0] != -2)

{

glVertex2d(kuzu3\_x,kuzu3\_y);

}

if(kuzu\_koordinat[3][0] != -2)

{

glVertex2d(kuzu4\_x,kuzu4\_y);

}

if(kuzu\_koordinat[4][0] != -2)

{

glVertex2d(kuzu5\_x,kuzu5\_y);

}

if(kuzu\_koordinat[5][0] != -2)

{

glVertex2d(kuzu6\_x,kuzu6\_y);

}

glEnd();

if(kuzu\_koordinat[0][0] != -2){

kuzu\_koordinat[0][0] = kuzu1\_x; kuzu\_koordinat[0][1] = kuzu1\_y;

}

if(kuzu\_koordinat[1][0] != -2){

kuzu\_koordinat[1][0] = kuzu2\_x; kuzu\_koordinat[1][1] = kuzu2\_y;

}

if(kuzu\_koordinat[2][0] != -2){

kuzu\_koordinat[2][0] = kuzu3\_x; kuzu\_koordinat[2][1] = kuzu3\_y;

}

if(kuzu\_koordinat[3][0] != -2){

kuzu\_koordinat[3][0] = kuzu4\_x; kuzu\_koordinat[3][1] = kuzu4\_y;

}

if(kuzu\_koordinat[4][0] != -2){

kuzu\_koordinat[4][0] = kuzu5\_x; kuzu\_koordinat[4][1] = kuzu5\_y;

}

if(kuzu\_koordinat[5][0] != -2){

kuzu\_koordinat[5][0] = kuzu6\_x; kuzu\_koordinat[5][1] = kuzu6\_y;

}

glColor3d(0,0,0);

if(kuzu\_koordinat[0][0] != -2)

{

glRasterPos2d (kuzu1\_x-0.01 , kuzu1\_y-0.01);

glutBitmapCharacter(font\_style,'S');

}

if(kuzu\_koordinat[1][0] != -2)

{

glRasterPos2d (kuzu2\_x-0.01 , kuzu2\_y-0.01);

glutBitmapCharacter(font\_style,'S');

}

if(kuzu\_koordinat[2][0] != -2)

{

glRasterPos2d (kuzu3\_x-0.01 , kuzu3\_y-0.01);

glutBitmapCharacter(font\_style,'S');

}

if(kuzu\_koordinat[3][0] != -2)

{

glRasterPos2d (kuzu4\_x-0.01 , kuzu4\_y-0.01);

glutBitmapCharacter(font\_style,'S');

}

if(kuzu\_koordinat[4][0] != -2)

{

glRasterPos2d (kuzu5\_x-0.01 , kuzu5\_y-0.01);

glutBitmapCharacter(font\_style,'S');

}

if(kuzu\_koordinat[5][0] != -2)

{

glRasterPos2d (kuzu6\_x-0.01 , kuzu6\_y-0.01);

glutBitmapCharacter(font\_style,'S');

} //kuzu bitir

for(int i=sayac2 ; i<sayac-1 ; i++){ // kopeg yörünge

glLineWidth(3);

glBegin(GL\_LINE\_STRIP);

glColor3d(1.0,1.0,1.0);

glVertex2d((double)dizi[i][0]/m1 , 1-(double)dizi[i][1]/m2);

glVertex2d((double)dizi[i+1][0]/m1 , 1-(double)dizi[i+1][1]/m2);

glEnd();

}

glPointSize(50);

glBegin(GL\_POINTS);

glColor3d(0.0,0.0,0.0);

glVertex2d(kopek\_a , kopek\_b);

glEnd();

glColor3d(1,1,1);

glRasterPos2d (kopek\_a-0.01 , kopek\_b-0.01);

glutBitmapCharacter(font\_style,'D');

// kurt1

int i1=0;

int en\_yakin1=-1;

double uzaklik1=1;

for(i1=0;i1<6;i1++)

{

if(kuzu\_koordinat[i1][0] != -2)

{

if(kuzu\_koordinat[i1][0] < 0.6 || kuzu\_koordinat[i1][1] < 0.5)

{

if( uzaklik1 > sqrt( pow( kurt\_a1-kuzu\_koordinat[i1][0] , 2 ) + pow( kurt\_b1-kuzu\_koordinat[i1][1] , 2 ) ) )

{

uzaklik1=sqrt( pow( kurt\_a1-kuzu\_koordinat[i1][0] , 2 ) + pow( kurt\_b1-kuzu\_koordinat[i1][1] , 2 ) );

en\_yakin1= i1 ;

}

}

}

}

if(en\_yakin1 != -1)

{

if(kurt\_a1 < kuzu\_koordinat[en\_yakin1][0])

{

kurt\_a1 += (double)(kurt\_hizi)/m1;

}

if(kurt\_a1 > kuzu\_koordinat[en\_yakin1][0])

{

kurt\_a1 -= (double)(kurt\_hizi)/m1;

}

if(kurt\_b1 < kuzu\_koordinat[en\_yakin1][1])

{

kurt\_b1 += (double)(kurt\_hizi)/m2;

}

if(kurt\_b1 > kuzu\_koordinat[en\_yakin1][1])

{

kurt\_b1 -= (double)(kurt\_hizi)/m2;

}

if(fabs(kurt\_a1 - kuzu\_koordinat[en\_yakin1][0]) < (double)3/m1 && fabs(kurt\_b1 - kuzu\_koordinat[en\_yakin1][1]) < (double)3/m1)

{

kuzu\_koordinat[en\_yakin1][0]=-2;

}

}

glPointSize(50);

glBegin(GL\_POINTS);

glColor3d(1,0,0);

glVertex2d(kurt\_a1 , kurt\_b1);

glEnd();

glColor3d(0,0,0);

glRasterPos2d (kurt\_a1-0.01 , kurt\_b1-0.01);

glutBitmapCharacter(font\_style,'W');

// kurt2

int i2=0;

int en\_yakin2=-1;

double uzaklik2=1;

for(i2=0;i2<6;i2++)

{

if(en\_yakin1 != i2)

{

if(kuzu\_koordinat[i1][0] != -2)

{

if(kuzu\_koordinat[i2][0] < 0.6 || kuzu\_koordinat[i2][1] < 0.5)

{

if( uzaklik2 > sqrt( pow( kurt\_a2-kuzu\_koordinat[i2][0] , 2 ) + pow( kurt\_b2-kuzu\_koordinat[i2][1] , 2 ) ) )

{

uzaklik2=sqrt( pow( kurt\_a2-kuzu\_koordinat[i2][0] , 2 ) + pow( kurt\_b2-kuzu\_koordinat[i2][1] , 2 ) );

en\_yakin2 = i2;

}

}

}

}

}

if(en\_yakin2 != -1)

{

if(kurt\_a2 < kuzu\_koordinat[en\_yakin2][0])

{

kurt\_a2 += (double)(kurt\_hizi)/m1;

}

if(kurt\_a2 > kuzu\_koordinat[en\_yakin2][0])

{

kurt\_a2 -= (double)(kurt\_hizi)/m1;

}

if(kurt\_b2 < kuzu\_koordinat[en\_yakin2][1])

{

kurt\_b2 += (double)(kurt\_hizi)/m2;

}

if(kurt\_b2 > kuzu\_koordinat[en\_yakin2][1])

{

kurt\_b2 -= (double)(kurt\_hizi)/m2;

}

if(fabs(kurt\_a2 - kuzu\_koordinat[en\_yakin2][0]) < (double)3/m1 && fabs(kurt\_b2 - kuzu\_koordinat[en\_yakin2][1]) < (double)3/m1)

{

kuzu\_koordinat[en\_yakin2][0]=-2;

}

}

glPointSize(50);

glBegin(GL\_POINTS);

glColor3d(1,0,0);

glVertex2d(kurt\_a2 , kurt\_b2);

glEnd();

glColor3d(0,0,0);

glRasterPos2d (kurt\_a2-0.01 , kurt\_b2-0.01);

glutBitmapCharacter(font\_style,'W');

if(!mouseleftdown)

{

glPointSize(50);

glBegin(GL\_POINTS);

glColor3d(0.0,0.0,0.0);

if(fabs( ((double)dizi[sayac2][0]/m1) - kopek\_a ) < (double)3/m1 && fabs( (1-(double)dizi[sayac2][1]/m2) - kopek\_b ) <(double)3/m2 ){

if(sayac2<sayac-1){

++sayac2;

}

}else {

if((double)dizi[sayac2][0]/m1 > kopek\_a){

kopek\_a+=(double)3/m1;

}

if((double)dizi[sayac2][0]/m1 < kopek\_a){

kopek\_a-=(double)3/m1;

}

if(1-(double)dizi[sayac2][1]/m2 > kopek\_b){

kopek\_b+=(double)3/m2;

}

if(1-(double)dizi[sayac2][1]/m2 < kopek\_b){

kopek\_b-=(double)3/m2;

}

}

glVertex2d(kopek\_a , kopek\_b);

glEnd();

glColor3d(1,1,1);

glRasterPos2d (kopek\_a-0.01 , kopek\_b-0.01);

glutBitmapCharacter(font\_style,'D');

}

}

void myDisplay(void) { // display callback

/\*cout << "MyDisplay called" << endl;\*/

glClearColor(0.1,0.7,0.1, 1.0); // background is gray

glClear(GL\_COLOR\_BUFFER\_BIT); // clear the window

// draw the objects

drawObjects();

glutSwapBuffers(); // swap buffers

}

void myTimer(int id) { // timer callback

glutPostRedisplay(); // request redraw

glutTimerFunc(TIMER\_DELAY, myTimer, 0); // reset timer for 10 seconds

}

void myMouse(int b, int s, int x, int y) { // mouse click callback

mouseleftdown=false;

if (s == GLUT\_DOWN) {

if (b == GLUT\_LEFT\_BUTTON) {

mx=x;

my=y;

if(fabs(kopek\_a\*m1-x) < 30 && fabs((1-kopek\_b)\*m2-y) < 30 ){

if( x < (m1-70) || y < (m2-70) ){

dizi[sayac][0] = x;

dizi[sayac][1] = y;

++sayac;

mouseleftdown=true;

}

}

else {

mouseleftdown=false;

}

if( x > (m1-70) && y > (m2-70) )

{

kopek\_hav = true;

\_beginthreadex(NULL, 0, dog, NULL, 0, NULL);

}

else {

kopek\_hav=false;

}

}

}

}

void motion(int x, int y)

{

if (mouseleftdown){

mx = x;

my = y;

if( x < (m1-70) || y < (m2-70) ){

dizi[sayac][0] = x;

dizi[sayac][1] = y;

++sayac;

}

}

}

// keyboard callback

void myKeyboard(unsigned char c, int x, int y) {

switch (c) { // c is the key that is hit

case 'q': // 'q' means quit

exit(0);

break;

default:

cout << "Hit q to quit. All other characters ignored" << endl;

break;

}

}

int main(int argc, char\*\* argv)

{

glutInit(&argc, argv); // OpenGL initializations

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);// double buffering and RGB

glutInitWindowSize(800, 600); // create a 400x400 window

glutInitWindowPosition(0, 0); // ...in the upper left

glutCreateWindow(argv[0]); // create the window

glutDisplayFunc(myDisplay); // setup callbacks

glutReshapeFunc(myReshape);

glutMouseFunc(myMouse);

glutMotionFunc(motion);

glutKeyboardFunc(myKeyboard);

glutTimerFunc(TIMER\_DELAY, myTimer, 0);

glutMainLoop(); // start it running

return 0; // ANSI C expects this

}