***Form1.cs***

﻿using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace SAWs

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

//For Two Dimensional SAW

int sw=20, st=12,deltax=1,deltay=1,x=0,y=0;

int [,] visited = new int[2\*st+1, 2\*st+1];

//to keep record of visited sites,0 for un-visited and 1 for visited

Point [,] position = new Point[2 \* st + 1, 2 \* st + 1];

//To keep record of positions

float[] swr2ave = new float[st];

float[] temp = new float[st];

//Graphics

Graphics gg = CreateGraphics();

SolidBrush bred = new SolidBrush(Color.Red);

SolidBrush bblue = new SolidBrush(Color.Blue);

SolidBrush byellow = new SolidBrush(Color.Yellow);

Font f = new Font("arial", 14);

Pen pblue = new Pen(Color.Blue);

//Draw the region of SAWs

for (int i = 0; i < 2 \* st + 1; i++)

{

for (int j = 0; j < 2 \* st + 1; j++)

{

gg.FillEllipse(bblue, 300 + j \* 10, 100 + i \* 10, 5, 5);

visited[i, j] = 0;//un-visited

if (i == st && j == st)

{

gg.FillEllipse(bred, 300 + j \* 10, 100 + i \* 10, 5, 5);

visited[i, j] = 1;//starting position/site is visited always

}

position[i, j].X = (j - st) \* deltax;

position[i, j].Y = (i - st) \* deltay;

// System.Threading.Thread.Sleep(30);

}

}

//start the SAWs

int saw\_no = 0,check=0;

Random[] walkers = new Random[sw];

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

walkers[i] = new Random(i);

double prob;

int rejected = 0;

while (saw\_no <sw)//walkers loop

{

check = 0;

x = 0; y = 0;//starting position of Walker

visited[x+st, y+st] = 1;

for (int i = 0; i < st; i++)//steps loop

{

prob = walkers[saw\_no].NextDouble();

if (prob <= 0.25)//side 1

x=x+deltax;

if (prob > 0.25&&prob<=0.5)//side 2

x = x - deltax;

if (prob >0.5&&prob<=0.75)//side 3

y = y + deltay;

if (prob>0.75)//side 4

y = y - deltay;

gg.FillEllipse(bred, 300 + (y+st) \* 10, 100 + (x+st) \* 10, 5, 5);

//System.Threading.Thread.Sleep(5000);

temp[i]=temp[i]+x\*x+y\*y;

//check new (x,y) whether already visited or not visited already

if (visited[x + st, y + st] == 1)//already visited

{

gg.FillEllipse(byellow, 300 + (y + st) \* 10, 100 + (x + st) \* 10, 5, 5);

//System.Threading.Thread.Sleep(1000);

check++;

rejected++;

break;

}

visited[x + st, y + st] = 1;//make the accepted site to be visited

}//end of steps loop

saw\_no++;

if (check != 0)//check whether the walk has been rejected or not

{

label2.Text = "SAW is rejected="+rejected.ToString();

label2.Refresh();

// System.Threading.Thread.Sleep(200);

saw\_no--;

//refresh temp array

for (int i = 0; i < st; i++)//steps loop

temp[i] = 0;

//make all sites to be again un-visited

for (int i = 0; i < 2 \* st + 1; i++)

{

for (int j = 0; j < 2 \* st + 1; j++)

{

gg.FillEllipse(bblue, 300 + j \* 10, 100 + i \* 10, 5, 5);

visited[i, j] = 0;//un-visited

if (i == st && j == st)

{

gg.FillEllipse(bred, 300 + j \* 10, 100 + i \* 10, 5, 5);

visited[i, j] = 1;//starting position/site is visited always

}

}

}//refreshing visited array

}//rejected walk

else

{ //SAW has been completed successfully

//store the information of the SAW

label3.Text = "SAW No Accepted="+(saw\_no).ToString();

// System.Threading.Thread.Sleep(5000);

label3.Refresh();

for (int i = 0; i < st; i++)//steps loop

{

swr2ave[i] = swr2ave[i] + temp[i];

temp[i] = 0;

}

//make all sites to be again un-visited

for (int i = 0; i < 2 \* st + 1; i++)

{

for (int j = 0; j < 2 \* st + 1; j++)

{

gg.FillEllipse(bblue, 300 + j \* 10, 100 + i \* 10, 5, 5);

visited[i, j] = 0;//un-visited

if (i == st && j == st)

{

gg.FillEllipse(bred, 300 + j \* 10, 100 + i \* 10, 5, 5);

visited[i, j] = 1;//starting position/site is visited always

}

}

}//refreshing visited array

}//else when SAW is accepted

}//walker loop ends

//\*\*\*\*\*\*\*\*SAWs have been recorded\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*Now record Real Random Walks in order to compare them with SAWs\*\*\*\*\*\*\*\*\*\*

float[] r2ave = new float[st];

for (int w = 0; w < sw; w++)//loop for real random walks in 2D

{

x = 0; y = 0;

for (int s = 0; s < st; s++)

{

prob = walkers[w].NextDouble();

if (prob <= 0.25)//side 1

x = x + deltax;

if (prob > 0.25 && prob <= 0.5)//side 2

x = x - deltax;

if (prob > 0.5 && prob <= 0.75)//side 3

y = y + deltay;

if (prob > 0.75)//side 4

y = y - deltay;

r2ave[s] = r2ave[s] + x \* x + y \* y;

}//steps loop ends

}//walker loop ends here

//Plot mean squared displacements from SAWs and Real Random Walks

for (int s = 0; s < st; s++)

{

gg.FillEllipse(bblue, 400 + s \* 15, 700 - swr2ave[s] / saw\_no\*5, 8, 8);//Mean squared displacement from SAWs

gg.FillEllipse(bred, 400 + s \* 15, 700 - r2ave[s] / saw\_no\*5, 8, 8);

}

}

private void button2\_Click(object sender, EventArgs e)

{

Walker wk = new Walker(20, this);//walker is standing at the starting site

int count = 0,exit=0;

while (wk.n >= 0)//until all SAWs are complete

{

if (wk.n == 0) exit++;

if (wk.SAWcheck() == false)//if SAW is not completed

{

wk.Forward();

label1.Text = "n=" + wk.n.ToString();

label1.Refresh();

//System.Threading.Thread.Sleep(1000);

if (wk.flag == 0)

{

//if (wk.visited[wk.y, wk.x].dir == 3) check++;

wk.NextDir();

}

if (wk.visited[wk.y, wk.x].dir > 4)

{

wk.Backward();

// System.Threading.Thread.Sleep(1000);

}

if (exit == 5)//loop break

{

MessageBox.Show("Number of SAWs completed="+count.ToString());

break;

}

}

else//when SAW is complete

{ count++;

wk.SAWsave();

wk.Backward();

//System.Threading.Thread.Sleep(1000);

label1.Text = "n=" + wk.n.ToString();

label1.Refresh();

}

}

this.Refresh();

for (int i = 0; i < wk.steps; i++)

{

wk.gg.FillEllipse(wk.bblue, 400 + i \* 20, 600 - wk.r2ave[i] / count, 8, 8);

}

}//event handler of button1 closing bracket

}

}

***Program.cs***

﻿using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

namespace SAWs

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

***Walker.cs***

﻿using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Drawing;

namespace SAWs

{

class Walker

{

//Declare variables first

public Graphics gg;

public SolidBrush bred,bblue,byellow;

public Pen pred, pblue;

Font arial;

public int steps,n,x,y,flag;

public Sites [,] visited;

public float[] temp; public float[] r2ave;

//constructor

public Walker(int s,Form1 frm1)

{

steps = s; n = -1;//n=-1 means walker is outside the walking region

gg = frm1.CreateGraphics();

bblue = new SolidBrush(Color.Blue);

bred = new SolidBrush(Color.Red);

byellow = new SolidBrush(Color.Yellow);

pblue = new Pen(Color.Blue);

pred = new Pen(Color.Red);

arial = new Font("Arial", 16);

temp = new float[steps+1];

r2ave = new float[steps+1];

visited = new Sites[2 \* steps + 1, 2 \* steps + 1];

for (int i = 0; i < 2 \* steps + 1; i++)

{

for (int j = 0; j < 2 \* steps + 1; j++)

{

visited[i, j] = new Sites(j, i);

//Draw the region

gg.FillEllipse(bblue, 500 + j \* 20, 100 + i \* 20, 8, 8);

//Make the walker stand at the starting site

if(i==steps&&j==steps)

{

visited[i, j].status = 1;//central site is visited now

visited[i, j].back = 5;//exit the region

n++;//At the start, number of visited sites is one; here n=0

gg.FillEllipse(bred, 500 + j \* 20, 100 + i \* 20, 8, 8);

x = j; y = i;

temp[n] = (x - steps) \* (x - steps) + (y - steps) \* (y - steps);

}

}

}

}//end of constructor

//other functions; to move, to check the SAW completed or not, to go back

//after completing SAW, to go back during movement when SAW is not completed

//save SAW information if completed, to check whether all the SAWs are completed

//or not

public bool SAWcheck()//this function will check whther the SAW is completed or not

{

if (n == steps)

return true;

else

return false;

}

public void SAWsave()//save SAW information

{

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

r2ave[i] = r2ave[i] + temp[i];

}

public void Forward()//for moving forward

{

flag = 0;

if (visited[y, x].dir == 1 && visited[y,x+1].status==0)

{

n++; flag++;

visited[y, x].dir++;

visited[y, x+1].back = 3;//for backward guidance

x++;

gg.FillEllipse(bred, 500 + x \* 20, 100 + y \* 20, 8, 8);

visited[y, x].status = 1;

temp[n] = (x - steps) \* (x - steps) + (y - steps) \* (y - steps);

}

if (visited[y, x].dir == 2 && visited[y-1, x].status == 0)

{

n++; flag++;

visited[y, x].dir++;

visited[y-1, x].back = 4;//for backward guidance

y--;

visited[y, x].status = 1;

gg.FillEllipse(bred, 500 + x \* 20, 100 + y \* 20, 8, 8);

temp[n] = (x - steps) \* (x - steps) + (y - steps) \* (y - steps);

}

if (visited[y, x].dir == 3 && visited[y, x - 1].status == 0)

{

n++; flag++;

visited[y, x].dir++;

visited[y, x - 1].back = 1;//for backward guidance

x--;

visited[y, x].status = 1;

gg.FillEllipse(bred, 500 + x \* 20, 100 + y \* 20, 8, 8);

temp[n] = (x - steps) \* (x - steps) + (y - steps) \* (y - steps);

}

if (visited[y, x].dir == 4 && visited[y+1, x].status == 0)

{

n++; flag++;

visited[y, x].dir++;

visited[y+1, x].back = 2;//for backward guidance

y++;

visited[y, x].status = 1;

gg.FillEllipse(bred, 500 + x \* 20, 100 + y \* 20, 8, 8);

temp[n] = (x - steps) \* (x - steps) + (y - steps) \* (y - steps);

}

//if (flag == 0 && visited[y, x].dir < 4)

//{

// NextDir();

// Forward();

//}

}//Forward ends

public void Backward()

{

if (SAWcheck()==true) visited[y, x].status = 0;

if (visited[y, x].back == 1)

{

n--;

gg.FillEllipse(bblue, 500 + x \* 20, 100 + y \* 20, 8, 8);

x++;

}

if (visited[y, x].back == 2)

{

n--;

gg.FillEllipse(bblue, 500 + x \* 20, 100 + y \* 20, 8, 8);

y--;

}

if (visited[y, x].back == 3)

{

n--;

gg.FillEllipse(bblue, 500 + x \* 20, 100 + y \* 20, 8, 8);

x--;

}

if (visited[y, x].back == 4)

{

n--;

gg.FillEllipse(bblue, 500 + x \* 20, 100 + y \* 20, 8, 8);

y++;

}

}//Backward ends

public void NextDir()

{

visited[y, x].dir++;

}

}

}

***Sites.cs***

﻿using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace SAWs

{

class Sites

{

//Data Variables

public float x, y;

public int dir, back, status;

//Member Functions

public Sites(float x, float y)

{

this.x = x; this.y = y; dir = 1; back = 0; status = 0;

}

}

}