***Program.cs***

﻿using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

namespace MeanFieldTheory

{

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());

}

}

}

***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 ClusterGrowthModels

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

ClusterGrowth ecgm = new ClusterGrowth(this,5);//0 for single point seed

//1 for x-axis as seed, 2 for y-axis as seed, 3 for y=x as seed

//4 for y=-x as seed, 5 as random seed, 6 for circle seed

while (true)

{

ecgm.perimeter();

ecgm.ECGM();

}

}

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

{

ClusterGrowth dla = new ClusterGrowth(this,5);

//0 for single point seed

//1 for x-axis as seed, 2 for y-axis as seed, 3 for y=x as seed

//4 for y=-x as seed, 5 as random seed, 6 for circle seed

while (true)

{

System.Threading.Thread.Sleep(10);

dla.perimeter();

dla.DLA();

}

}

}

}

***ClusterGrowth.cs***

﻿using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Drawing;

namespace ClusterGrowthModels

{

class ClusterGrowth

{

Random obj;

Graphics gg;

SolidBrush bblue, bred, bwhite;

Pen pblue, pred;

public int [,] Sites;

public int[,] Perims;

int nsites,x,y;

double prob;

//constructor

public ClusterGrowth(Form1 frm, int flag)

{

//flag=0 meens single seed

//flag=1 means x-axis as seed

//flag=2 means y-axis as seed

//flag=3 means y=x as seed

//flag=4 means y=-x as seed

obj = new Random();

gg = frm.CreateGraphics();

bblue = new SolidBrush(Color.Blue);

bred = new SolidBrush(Color.Red);

pblue = new Pen(Color.Blue);

pred = new Pen(Color.Red);

nsites = 30;

Sites = new int[2 \* nsites + 1, 2 \* nsites + 1];

Perims = new int[2 \* nsites + 1, 2 \* nsites + 1];

//make cluster region

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

{

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

{

gg.FillEllipse(bblue, 200 + j \* 10, 200 + i \* 10, 9, 9);

if (flag == 0)//single point seed

{

if (i == nsites && j == nsites)//seed is single point

{

gg.FillEllipse(bred, 200 + j \* 10, 200 + i \* 10, 9, 9);

Sites[i, j] = 1;//1 stands for affected/occupied

}

}

if (flag == 1)//x-axis as seed

{

if (i == nsites)//x-axis as seed

{

gg.FillEllipse(bred, 200 + j \* 10, 200 + i \* 10, 9, 9);

Sites[i, j] = 1;//1 stands for affected/occupied

}

}

if (flag == 2)//y-axis as seed

{

if (j == nsites)//y-axis as seed

{

gg.FillEllipse(bred, 200 + j \* 10, 200 + i \* 10, 9, 9);

Sites[i, j] = 1;//1 stands for affected/occupied

}

}

if (flag == 3)//y=x as seed

{

if ((i+j)==2\*nsites)//y=x as seed

{

gg.FillEllipse(bred, 200 + j \* 10, 200 + i \* 10, 9, 9);

Sites[i, j] = 1;//1 stands for affected/occupied

}

}

if (flag == 4)//y=-x as seed

{

if (i==j)//y=-x as seed

{

gg.FillEllipse(bred, 200 + j \* 10, 200 + i \* 10, 9, 9);

Sites[i, j] = 1;//1 stands for affected/occupied

}

}

if (flag == 5)//Random seed

{

if (obj.NextDouble()<0.003)//random seed

{

gg.FillEllipse(bred, 200 + j \* 10, 200 + i \* 10, 9, 9);

Sites[i, j] = 1;//1 stands for affected/occupied

}

}

if (flag == 6)//Circle seed

{

if (Math.Abs((j-nsites)\*(j-nsites)+(i-nsites)\*(i-nsites)-25)<0.01)

//circle seed

{

//x=j-nsites, y=i-nsites

gg.FillEllipse(bred, 200 + j \* 10, 200 + i \* 10, 9, 9);

Sites[i, j] = 1;//1 stands for affected/occupied

}

}

}

}

}//constructor ends

//other functions

public void perimeter()

{

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

{

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

{

if (Sites[i, j] == 1)

{

if (Sites[i - 1, j] == 0)

Perims[i - 1, j] = 1;

if (Sites[i + 1, j] == 0)

Perims[i + 1, j] = 1;

if (Sites[i, j-1] == 0)

Perims[i, j-1] = 1;

if (Sites[i, j+1] == 0)

Perims[i, j+1] = 1;

}

}

}

}//end of perimeter function

public void ECGM()

{

x = obj.Next(2 \* nsites+1);

y = obj.Next(2 \* nsites + 1);

if (Perims[y, x] == 1)

{

Sites[y, x] = 1;

Perims[y, x] = 0;

gg.FillEllipse(bred, 200 + x \* 10, 200 + y \* 10, 9, 9);

}

}

public void DLA()

{

x = obj.Next(2 \* nsites + 1);

y = obj.Next(2 \* nsites + 1);

while (Perims[y, x] != 1)

{

prob = obj.NextDouble();

if (prob < 0.25)

{

if(x<2\*nsites)

x++;

}

if (prob >= 0.25&&prob<0.5)

{

if (x>0)

x--;

}

if (prob>=0.5&&prob<0.75)

{

if (y<2\*nsites)

y++;

}

if (prob >= 0.75)

{

if (y>0)

y--;

}

}//while loop

Sites[y, x] = 1;

Perims[y, x] = 0;

gg.FillEllipse(bred, 200 + x \* 10, 200 + y \* 10, 9, 9);

}

}

}