**Федеральное государственное образовательное бюджетное учреждение высшего профессионального образования**

**Московский технический университет связи и информатики**

**КАФЕДРА ИНФОРМАТИКИ**

**Лабораторная работа по теме 3  
Проектирование различных дополнений для моделирования SIS/SIR на сети.**

**Выполнила:**

**Студентка группы БВТ1701**

**Сенькова Ольга**

**Вариант 22**

**Москва 2019**

**Цель работы:** реализовать различные дополнения, с помощью которых была бы наглядна продемонстрирована работа сети во 2 лабораторной работе.

**1 Формулировка задания на лабораторную работу**

Для сети, спроектированной в лабораторной работе №2, реализовать статистику, графики, удаление и восстановление узлов и ребер, изменение состояния узла при нажатии на холст.

**2 Техническое задание**

-Параметры ввода: выбор действия с узлом (изменить состояние или удалить/восстановить), с ребром (удалить или восстановить)

-Параметры вывода: диаграмма и графики изменения состояний узлов, статистика: номера узлов, состояние, количество заражений, последние время восстановления

-Класс Node: класс узлов, состоящий из i- номер узла по вертикали, j- номер узла по горизонтали, X- координаты узла по оси X, Y- координаты узла по осиY

**3 Функции программы**

- private void Form1\_MouseDown(object sender, MouseEventArgs e)

- private void DrawLineChart()

- private void DrawPieChart(int value1, int value2, int value3)

**4 Код программы**

private void Form1\_MouseDown(object sender, MouseEventArgs e)

{

int x = e.X;

int y = e.Y;

double minDest1 = double.PositiveInfinity;

double minDest2 = double.PositiveInfinity;

double Dest;

int k = 0;

int l = 0;

for (int i=0; i< Nodes.Count; i++)

{

Dest = Math.Sqrt(Math.Pow(x - Nodes[i].X, 2) + Math.Pow(y-Nodes[i].Y,2));

if (Dest < minDest1)

{

minDest1 = Dest;

k = i;

}

}

for (int i = 0; i < Nodes.Count; i++)

{

Dest = Math.Sqrt(Math.Pow(x - Nodes[i].X, 2) + Math.Pow(y - Nodes[i].Y, 2));

if (Dest < minDest2 && Dest!=minDest1)

{

minDest2 = Dest;

l = i;

}

}

var a = CreateGraphics();

SolidBrush mySolidBrushI = new SolidBrush(Color.Red);

SolidBrush mySolidBrushS = new SolidBrush(Color.Yellow);

SolidBrush mySolidBrushR = new SolidBrush(Color.Green);

SolidBrush mySolidBrushRem = new SolidBrush(Color.White);

if (checkBox2.Checked)

{ if (radioButton5.Checked)

{

if (radioButton1.Checked)

{

if (Nodes[k].S == true)

{

Nodes[k].I = true;

Nodes[k].S = false;

NI++;

NS--;

a.FillEllipse(mySolidBrushI, (int)Nodes[k].X - 50 / m, (int)Nodes[k].Y - 50 / m, (50 / m) \* 2, (50 / m) \* 2);

}

else if (Nodes[k].I == true)

{

Nodes[k].I = false;

Nodes[k].S = true;

NS++;

NI--;

a.FillEllipse(mySolidBrushS, (int)Nodes[k].X - 50 / m, (int)Nodes[k].Y - 50 / m, (50 / m) \* 2, (50 / m) \* 2);

}

}

else if (radioButton2.Checked)

{

if (Nodes[k].S == true)

{

Nodes[k].I = true;

Nodes[k].S = false;

NI++;

NS--;

a.FillEllipse(mySolidBrushI, (int)Nodes[k].X - 50 / m, (int)Nodes[k].Y - 50 / m, (50 / m) \* 2, (50 / m) \* 2);

}

else if (Nodes[k].I == true)

{

Nodes[k].I = false;

Nodes[k].R = true;

NI--;

NR++;

a.FillEllipse(mySolidBrushR, (int)Nodes[k].X - 50 / m, (int)Nodes[k].Y - 50 / m, (50 / m) \* 2, (50 / m) \* 2);

}

else if (Nodes[k].R == true)

{

Nodes[k].R = false;

Nodes[k].S = true;

NS++;

NR--;

a.FillEllipse(mySolidBrushS, (int)Nodes[k].X - 50 / m, (int)Nodes[k].Y - 50 / m, (50 / m) \* 2, (50 / m) \* 2);

}

}

}

else if (radioButton6.Checked)

{

if (Nodes[k].S == true)

{

Nodes[k].Rem = true;

Nodes[k].S = false;

NS--;

a.FillEllipse(mySolidBrushRem, (int)Nodes[k].X - 50 / m, (int)Nodes[k].Y - 50 / m, (50 / m) \* 2, (50 / m) \* 2);

}

else if (Nodes[k].Rem == true)

{

Nodes[k].Rem = false;

Nodes[k].S = true;

NS++;

a.FillEllipse(mySolidBrushS, (int)Nodes[k].X - 50 / m, (int)Nodes[k].Y - 50 / m, (50 / m) \* 2, (50 / m) \* 2);

}

}

}

if (checkBox3.Checked)

{

if (radioButton4.Checked)

{

Pen WhitePen = new Pen(Color.White, 3);

a.DrawLine(WhitePen, Nodes[k].X, Nodes[k].Y, Nodes[l].X, Nodes[l].Y);

WhitePen.Dispose();

}

else if (radioButton3.Checked)

{

Pen WhitePen = new Pen(Color.Green, 3);

a.DrawLine(WhitePen, Nodes[k].X, Nodes[k].Y, Nodes[l].X, Nodes[l].Y);

WhitePen.Dispose();

}

if (Nodes[k].I == true)

a.FillEllipse(mySolidBrushI, (int)Nodes[k].X - 50 / m, (int)Nodes[k].Y - 50 / m, (50 / m) \* 2, (50 / m) \* 2);

else if (Nodes[k].S == true)

a.FillEllipse(mySolidBrushS, (int)Nodes[k].X - 50 / m, (int)Nodes[k].Y - 50 / m, (50 / m) \* 2, (50 / m) \* 2);

else if (Nodes[k].Rem == true)

a.FillEllipse(mySolidBrushRem, (int)Nodes[k].X - 50 / m, (int)Nodes[k].Y - 50 / m, (50 / m) \* 2, (50 / m) \* 2);

else if (Nodes[k].R == true)

a.FillEllipse(mySolidBrushR, (int)Nodes[k].X - 50 / m, (int)Nodes[k].Y - 50 / m, (50 / m) \* 2, (50 / m) \* 2);

if (Nodes[l].I == true)

a.FillEllipse(mySolidBrushI, (int)Nodes[l].X - 50 / m, (int)Nodes[l].Y - 50 / m, (50 / m) \* 2, (50 / m) \* 2);

else if (Nodes[l].S == true)

a.FillEllipse(mySolidBrushS, (int)Nodes[l].X - 50 / m, (int)Nodes[l].Y - 50 / m, (50 / m) \* 2, (50 / m) \* 2);

else if (Nodes[l].Rem == true)

a.FillEllipse(mySolidBrushRem, (int)Nodes[l].X - 50 / m, (int)Nodes[l].Y - 50 / m, (50 / m) \* 2, (50 / m) \* 2);

else if (Nodes[l].R == true)

a.FillEllipse(mySolidBrushR, (int)Nodes[l].X - 50 / m, (int)Nodes[l].Y - 50 / m, (50 / m) \* 2, (50 / m) \* 2);

}

mySolidBrushS.Dispose();

mySolidBrushI.Dispose();

mySolidBrushR.Dispose();

mySolidBrushRem.Dispose();

}

private void DrawLineChart()

{

var series = new Series("S");

chart2.Series[0].Points.AddXY(NX, NS);

chart2.Series[1].Points.AddXY(NX, NI);

chart2.Series[2].Points.AddXY(NX, NR);

}

private void DrawPieChart(int value1, int value2, int value3)

{

//reset your chart series and legends

chart1.Series.Clear();

chart1.Legends.Clear();

chart1.Palette = ChartColorPalette.None;

chart1.PaletteCustomColors = new Color[] { Color.Gold, Color.Red, Color.Lime };

//Add a new chart-series

string seriesname = "MySeriesName";

chart1.Series.Add(seriesname);

//set the chart-type to "Pie"

chart1.Series[seriesname].ChartType = SeriesChartType.Pie;

//Add some datapoints so the series. in this case you can pass the values to this method

chart1.Series[seriesname].Points.AddXY("S", value1);

chart1.Series[seriesname].Points.AddXY("I", value2);

chart1.Series[seriesname].Points.AddXY("R", value3);

}

**5 Результат программы**

На рисунке 1 и 2 (SIS/SIR) изображены все дополнения ко второй программе.

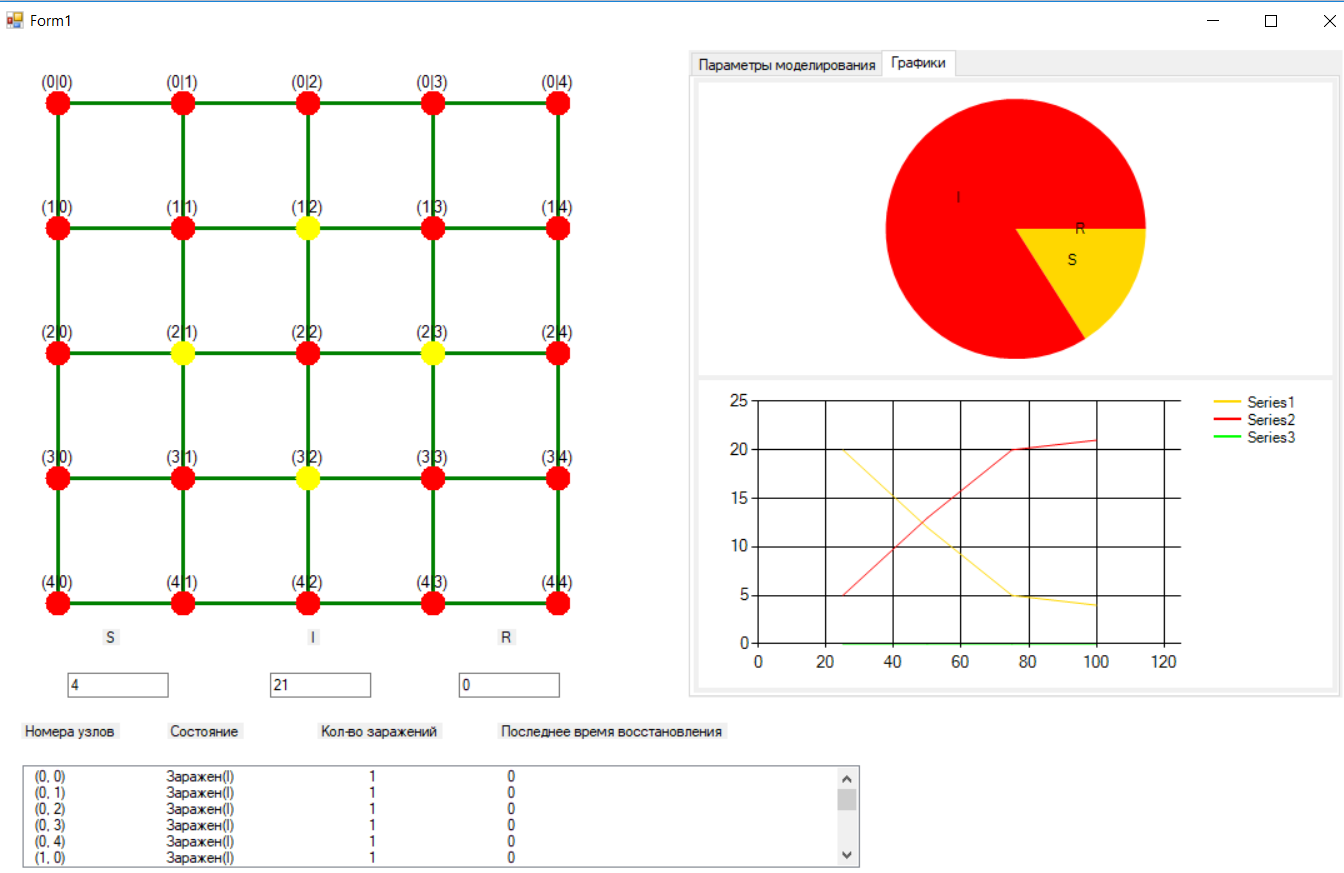


Рисунок 1 – SIS

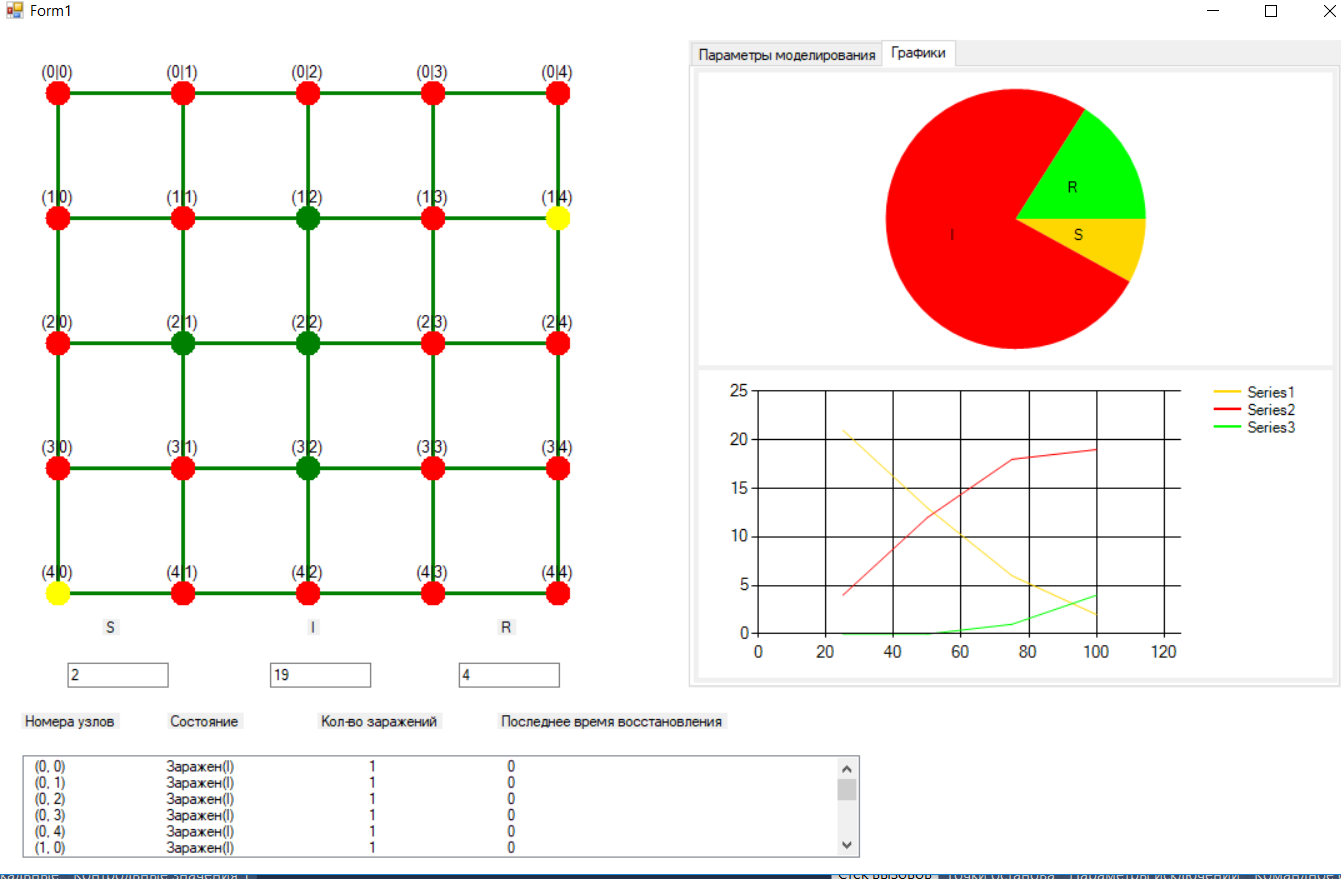


Рисунок 2 - SIR.

**Вывод:** для полноты картины и верной работы первой программы были осуществлены различные дополнения, научились стоить графики, диаграммы, удалять и восстанавливать ребра, менять состояния узлов.