**Цель:** моделирование схемы дорог линейно-кусочным методом, программное представление данных, генерация случайного положения точки на карте, определение множеств пикселей одинаковых цветов и вероятности попадания случайной точки в каждый цвет.

**Этапы работы:**

1. Создание изображения, состоящего из трех цветов, и подсчет величины площади каждого цвета.
2. Моделирование схемы дорог линейно-кусочным методом на карте, определение длин отрезков дороги и программное представление данных.
3. Определение координат точек концов отрезков частей дорог, заданных линейно-кусочным образом и программное представление данных.
4. Определение множеств пикселей одинаковых цветов и вероятности попадания случайной точки в каждый цвет.
5. Генерация случайных координат точки и определение цвета местности попадания точки.

**Средства разработки**

Данное задание предполагало плотную работу с графическим интерфейсом, а также использование и создание изображений в реальном времени, поэтому необходимо было выбрать инструменты и средства, удобные для дальнейшего их использования в разработке. Было решено использовать Visual Studio Community 2017 и язык программирования C#, так как этот язык позволяет работать с графикой, содержит необходимые классы и интерфейсы. Также данные средства делают возможным проектирование приложений с помощью уже готовых инструментов, что значительно упрощает разработку.

**Фронт работы**

Так как данная работа выполнялась совместно с Подгорной Виолеттой, мы приняли решение разделить обязанности и распределить этапы работы между собой. Первый и второй этапы были сделаны мной, а пятый этап выполнялся совместно.

**Первый этап**

На данном этапе необходимо было понять, как выбранные средства разработки позволяют нам работать с графикой и изображениями.

*Создание изображения.* Visual Studio и C# дают возможность как использования готового изображения, так и его создания внутри среды. Для загрузки выбранного изображения в C# используется специальное средство работы с диалоговыми окнами OpenFileDialog. Реализация выглядит так: создаем загрузчик и прописываем действие по открытию изображения на кнопке.

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

{

if (this.openFileDialog1.ShowDialog() == DialogResult.OK)// ответ в диалоговом окне "Ок"

{

try

{

this.pictureBox1.Load(this.openFileDialog1.FileName);

bitmap = new Bitmap(this.pictureBox1.Image);

image = this.pictureBox1.Image;

pictureBox1.Invalidate();

площадьToolStripMenuItem.Enabled = true;

}

catch (InvalidOperationException exc)

{

MessageBox.Show(exc.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);//вывод ошибки

}

catch (System.IO.FileNotFoundException exc)

{

MessageBox.Show(exc.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);//вывод ошибки

}

catch (ArgumentException exc)

{

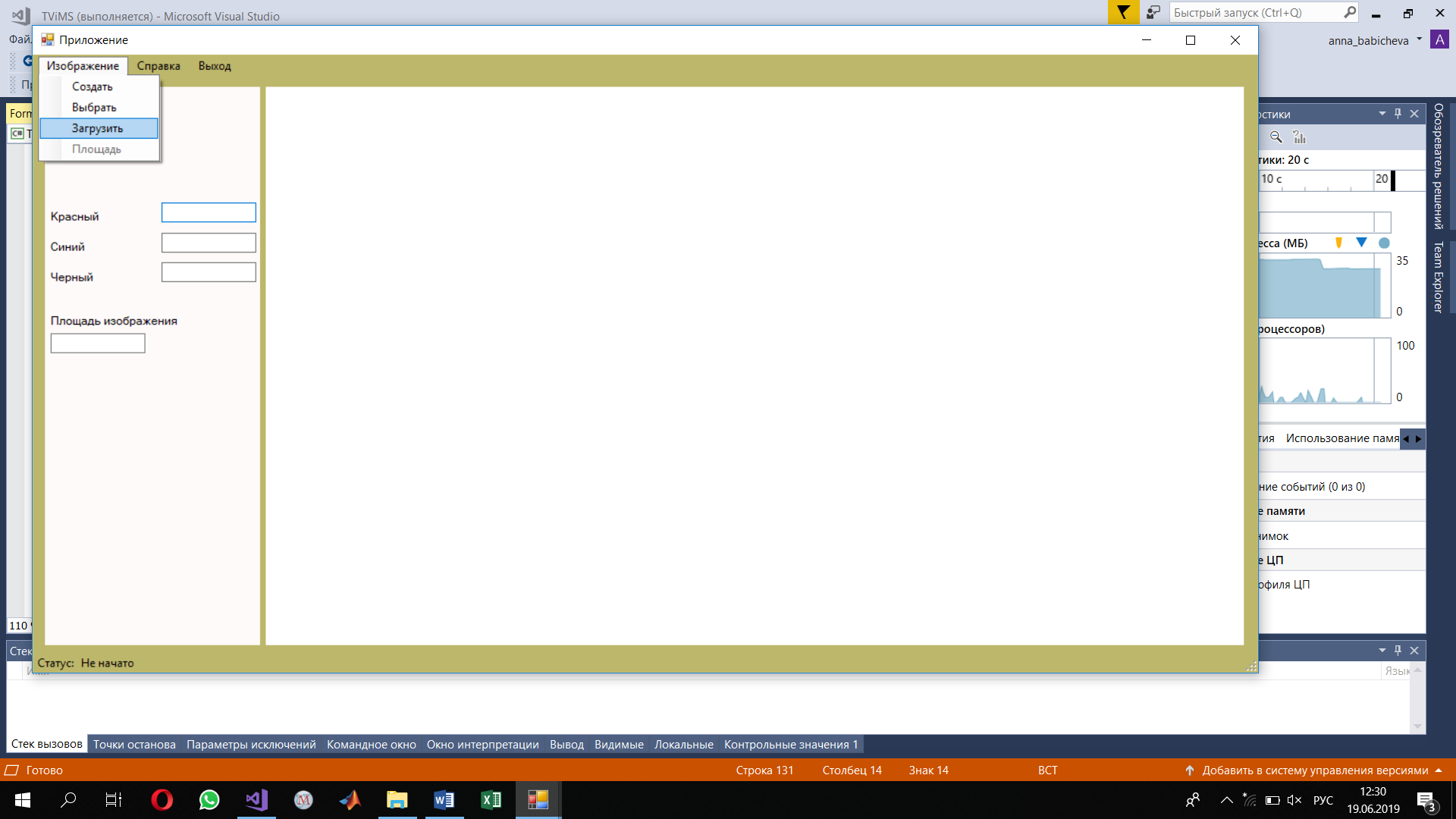
MessageBox.Show(exc.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);//вывод ошибки

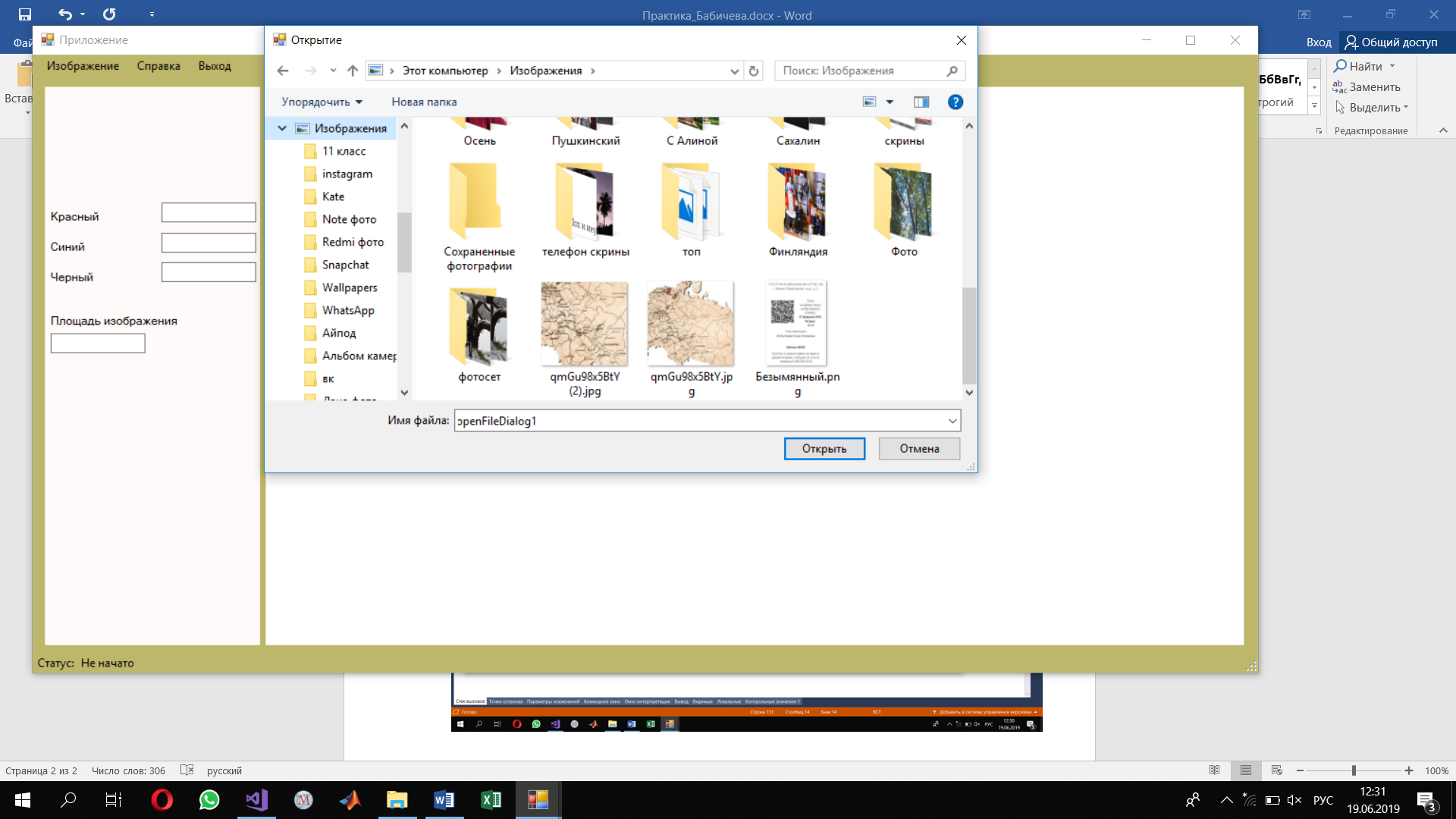
}

}

}

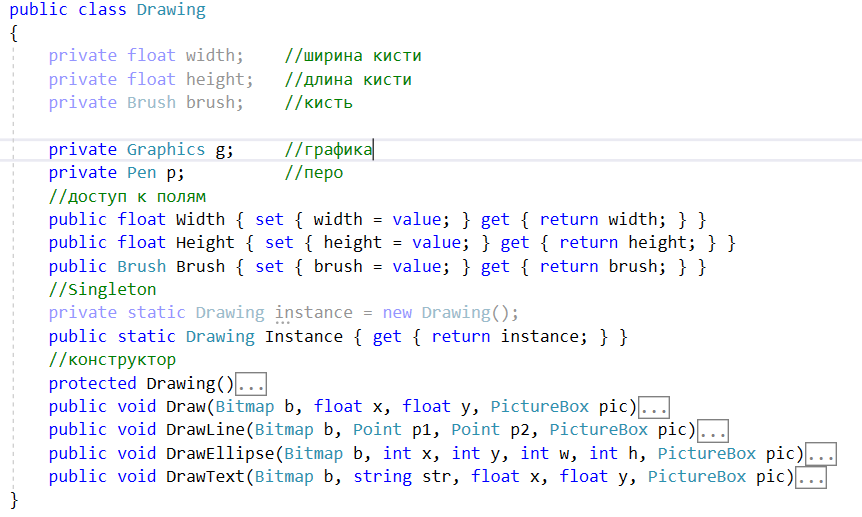
Таким образом, получаем следующее при работе:



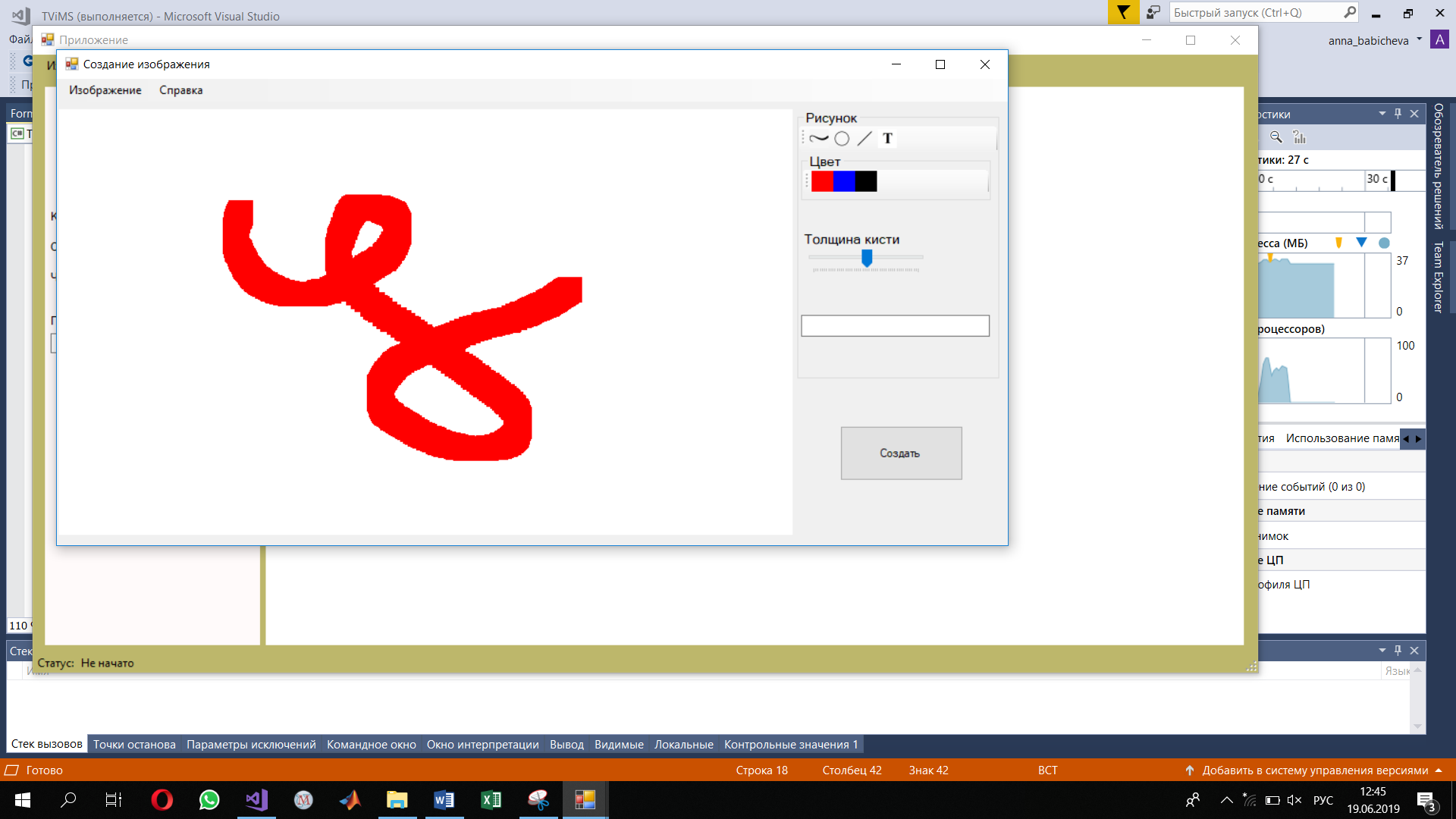


Для создания изображения необходимо использовать библиотеки System.Drawing и

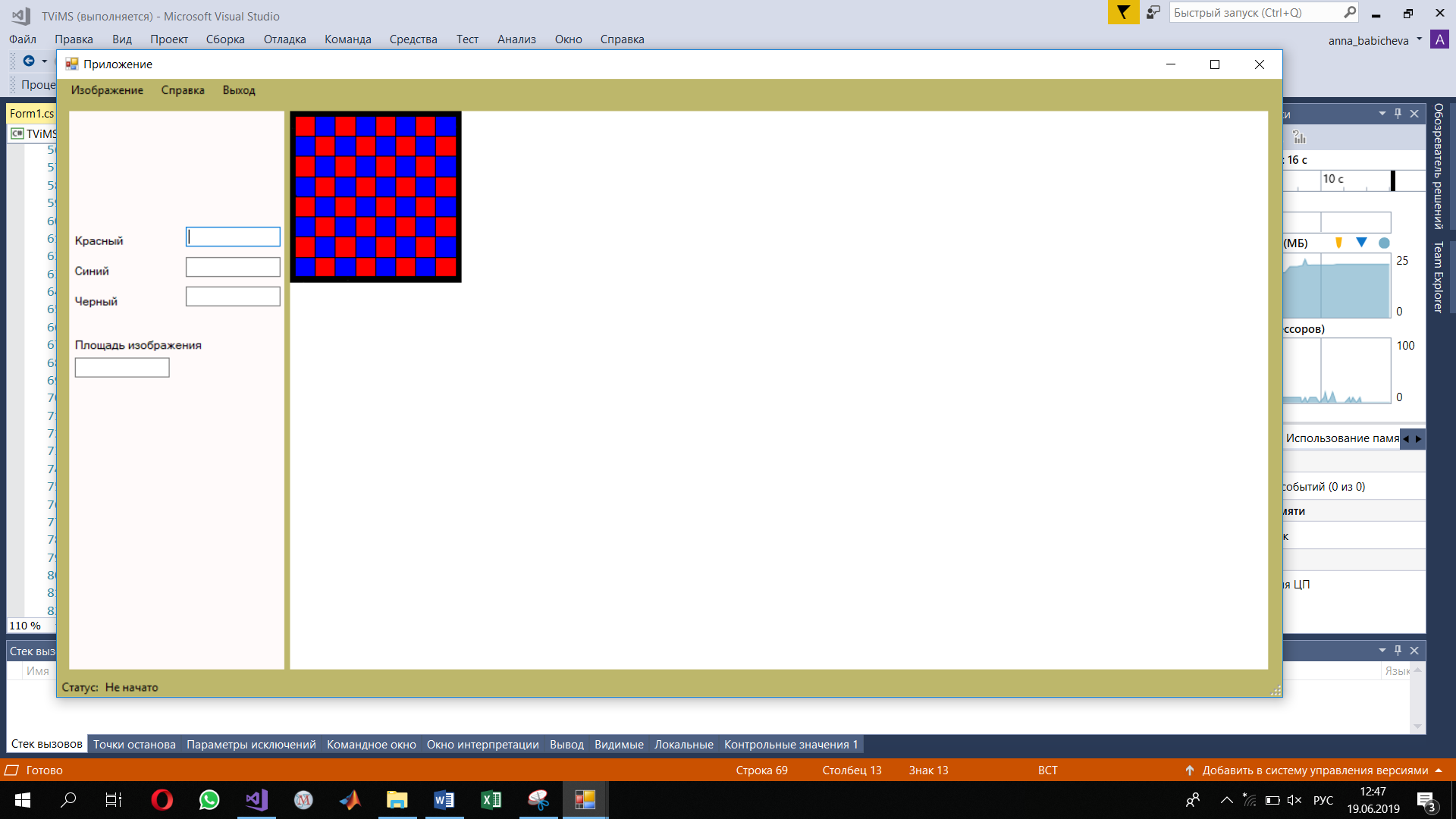
System.Drawing.Drawing2D, в которых есть необходимые классы и функции для работы с графикой. Однако было принято решение разделить работу графической и вычислительной составляющих программы. Для этого был выделен отдельный класс Drawing, работающий с графическим интерфейсом. В нем были прописаны функции для прорисовки различных фигур и кривых, а также атрибуты доступа к размерам и цвету кисти.



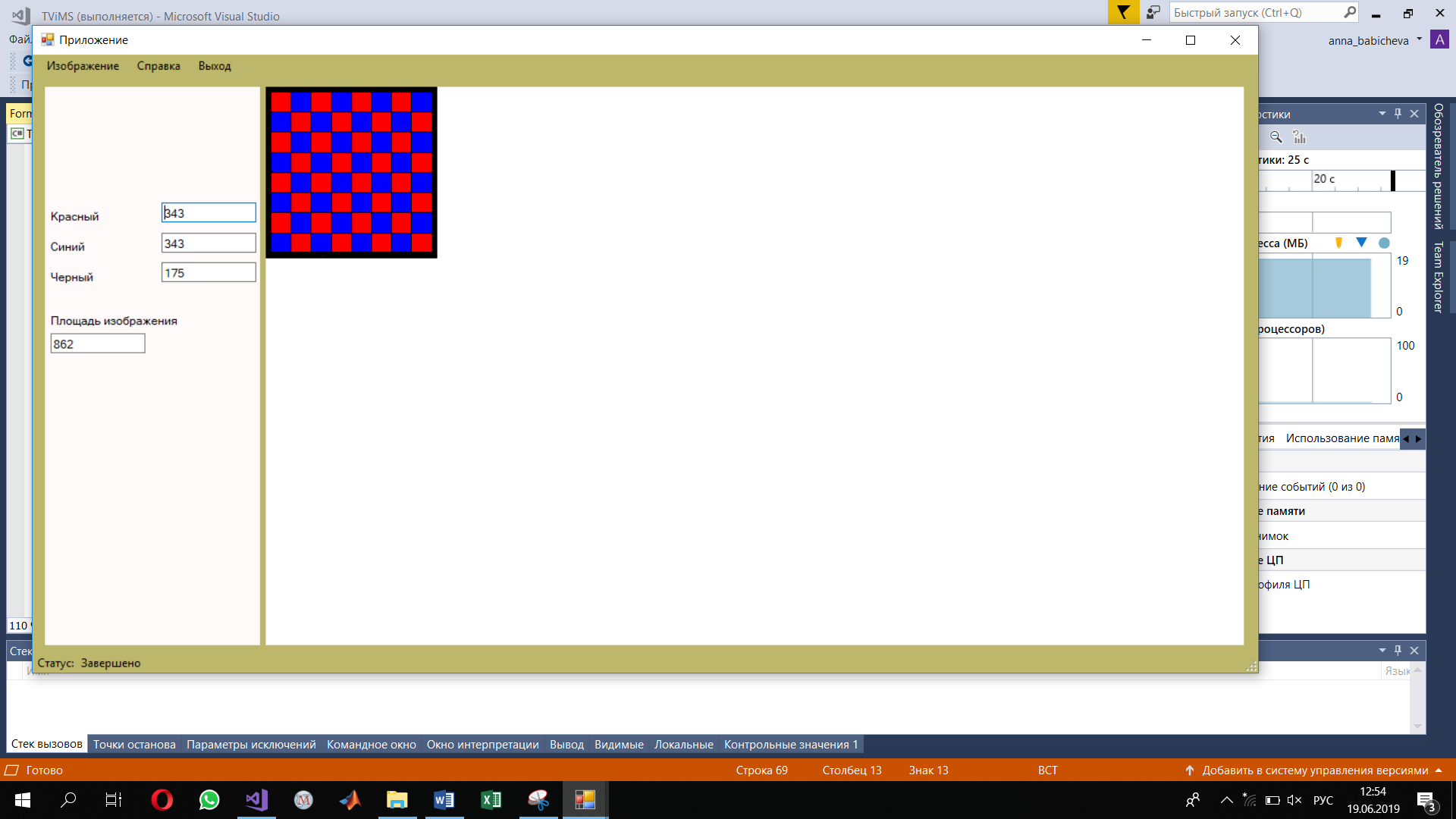
Программа вызывает данный класс с необходимой функцией для прорисовки тех или иных элементов, результатом чего является:



Вся прорисовка происходит на еще одном средстве C# - Bitmap, которое и передается на главную форму по завершению создания изображения:



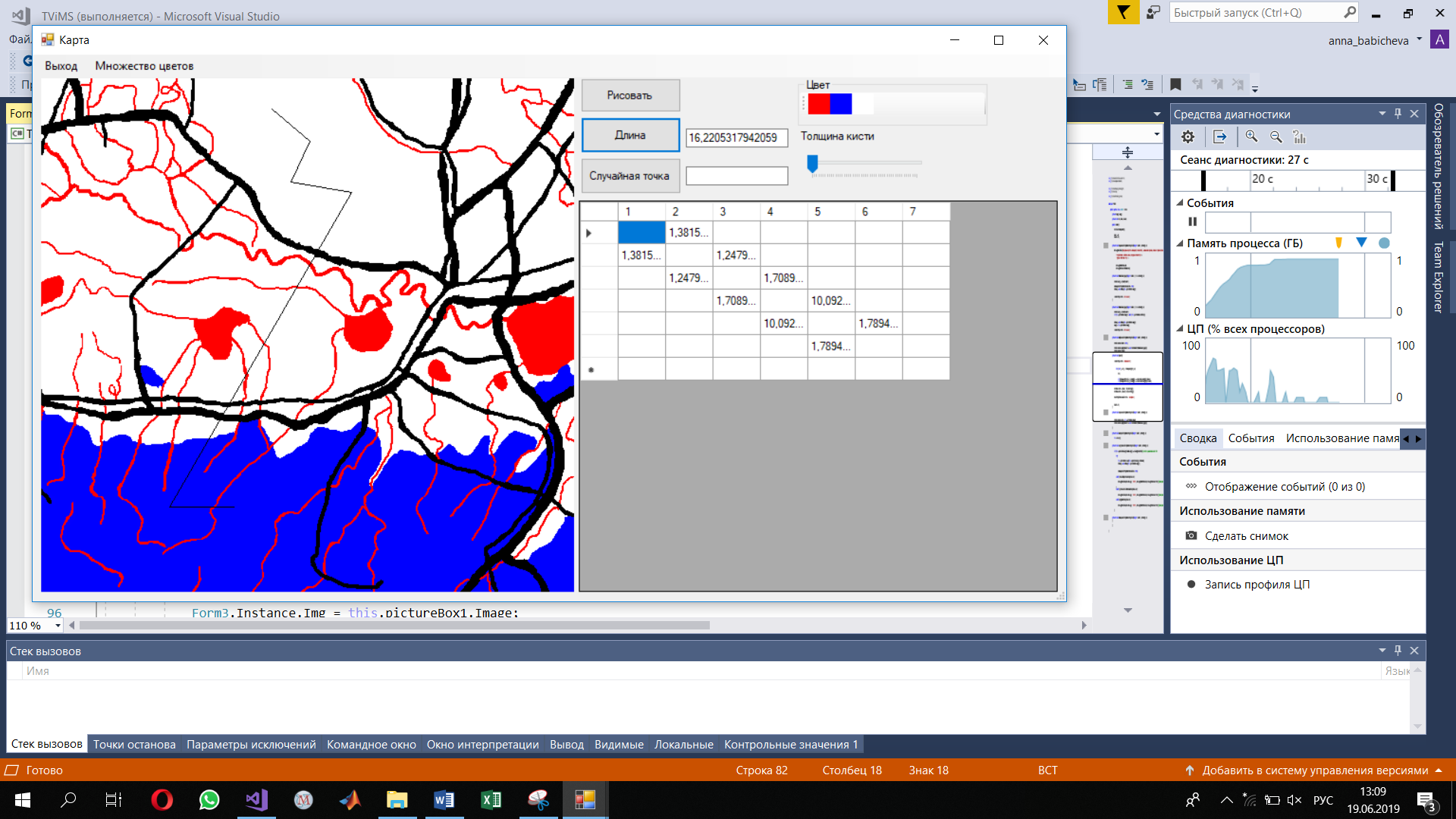
*Подсчет площади.*Данная часть программы является вычислительной, однако она использует свойства пикселей изображения, поэтому не следует забывать про необходимые библиотеки. Принцип подсчета прост: проходим по всем пикселям изображения, считая количество тех или иных цветов (в данном случае красного, синего и черного), что приводит к следующему результату:



**Второй этап**

Второй этап работы был направлен на изучение линейно-кусочного метода представления дорог в памяти компьютера. Он подразумевает, что дороги задаются множеством соединенных между собой коротких отрезков, моделирующих изгибы и повороты дороги. Этот метод позволяет хранить в памяти компьютера матрицу смежности для полученного графа, вершинами которого являются концы отрезков. На этом этапе матрица смежности заполнялась длинами в сантиметрах отрезков между двумя соединенными между собой вершинами. Длина вычислялась с помощью координат в пикселях начала и конца отрезка, а позже переводилась в сантиметры путем деления на 38 (приближенное значение количества пикселей в сантиметре). Сами же отрезки прорисовывались с помощью созданного класса Drawing и средств контроля времени и места прорисовки, определенных на форме. Матрица смежности была вынесена на форму в виде стандартного инструмента в C# - DataGridView. Начало прорисовки происходит после нажатия кнопки «Рисовать», окончание по двойному щелчку мыши.

Результат работы:



**Третий этап**

Выполнялся Подгорной Виолеттой.

**Четвертый этап**

Выполнялся Подгорной Виолеттой.

**Пятый этап**

(Выполнялся совместно)

Для генерации случайного положения точки необходимо сгенерировать случайным образом координаты этой точки. Случайная генерация чисел также предусмотрена средствами C# с помощью класса Random. Для работы с данным классом необходимо лишь определить область допустимых значений случайной величины. Значения будут получены в промежутке размеров изображения. Таким образом, генерируем координаты точки, а далее рисуем ее с помощью Drawing. Также определяем цвет точки.

private void button3\_Click(object sender, EventArgs e)

{

rand\_x = rand.Next(1, this.pictureBox1.Width);

rand\_y = rand.Next(1, this.pictureBox1.Height);

if (bmp.GetPixel(rand\_x, rand\_y).ToArgb() == Color.Red.ToArgb()) color = "Red";

if (bmp.GetPixel(rand\_x, rand\_y).ToArgb() == Color.Blue.ToArgb()) color = "Blue";

if (bmp.GetPixel(rand\_x, rand\_y).ToArgb() == Color.White.ToArgb()) color = "White";

this.textBox2.Text = "(" + rand\_x.ToString() + " , " + rand\_y.ToString() + ") " + color;

Drawing.Instance.Brush = new SolidBrush(Color.Yellow);

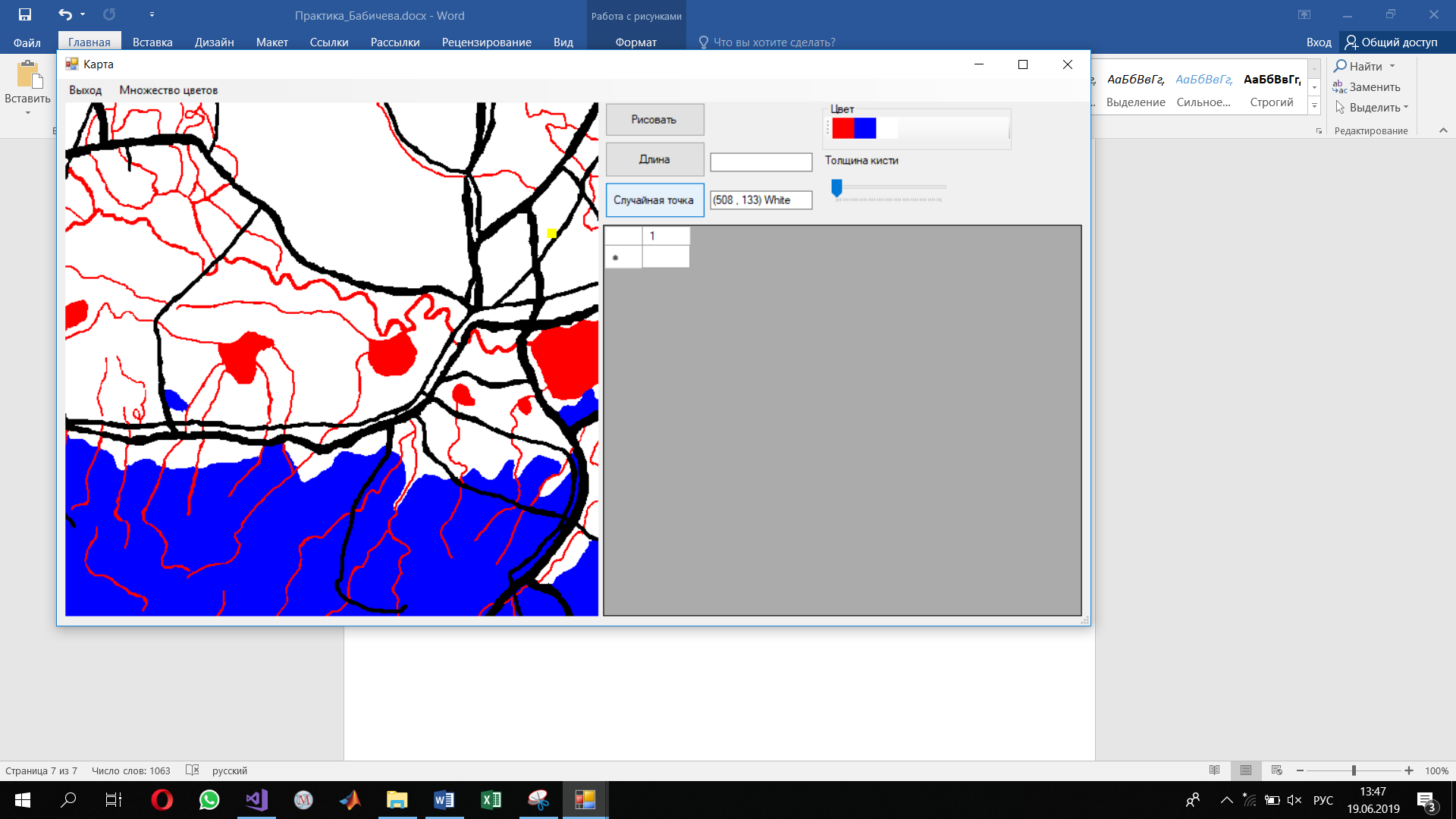
Drawing.Instance.Height = 10;

Drawing.Instance.Width = 10;

Drawing.Instance.Draw(bmp, rand\_x, rand\_y, this.pictureBox1);

}

Результат работы:



**Вывод**

Мы смоделировали схему дорог линейно-кусочным методом, программно представили данные, сгенерировали случайное положение точки на карте, определили множества пикселей одинаковых цветов и вероятности попадания случайной точки в каждый цвет.