МИНЕСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
МОСКОВСКИЙ ЭНЕРГЕТИЧЕСКИЙ ИНСТИТУТ

Отчет к лабораторной работе № 8

Дисциплина: «Системное программирование»

Михаил Чуворкин
19.12.2021

Лабораторная работа 8

Процессы и потоки

Задание

Для выполнения работы создать приложение на языке C#.

1. Программа должна выдать информацию о первичном (главном) потоке, создать еще три потока.

   По кнопке в поле label будем выводить информацию о первичном потоке (имя потока добавлено вручную):

```
private void btnAboutMainThread_Click(object sender, EventArgs e)
{
    Thread mainThread = Thread.CurrentThread;
    mainThread.Name = "Первичный поток";
    labelAboutMainThread.Text = "Имя домена: " + Thread.GetDomain().FriendlyName +
                "\nУровень приоритета: " + mainThread.Priority +
                "\nСостояние потока: " + mainThread.ThreadState.ToString() +
                "\nИмя потока: " + mainThread.Name +
                "\nПоток запущен? " + (mainThread.IsAlive ? "Да" : "Нет");
}
```

Еще три потока будут создаваться по другой кнопке (чтобы можно было перезапускать нарезку)

```
hammersThread = new Thread(hammersImgDoWork);
wywhThread = new Thread(wywhImgDoWork);
infoThread = new Thread(infoDoWork);
```

2. Два вторичных потока должны быть созданы на основе двух разных функций, каждая из которых реализует метод случайного представления фрагментов одной из двух картинок (**каждая картинка режется на квадратики – не меньше девяти**). Поток должен завершиться, если «соберет» свою картинку в правильном порядке.

   Добавим необходимые объекты:

```
Bitmap hammers;
Bitmap wywh;
```

```
        Graphics pb1Graphics;
        Graphics pb2Graphics;
        Graphics pb3Graphics;
        List<Bitmap> hammersSlices;
        List<Bitmap> wywhSlices;
```

Напишем функцию, которая нарежет картинки на части и сформирует список из частей:

```
        private List<Bitmap> SliceImage(Bitmap img) {
            int fragmentHeight = img.Height / 3;
            int fragmentWidth = img.Width / 3;
            List<Bitmap> imgList = new List<Bitmap> { };
            /*
             * Нарезка построчно, индексы
             * 0 1 2
             * 3 4 5
             * 6 7 8
             */
            for (int i = 0; i < fragmentHeight * 3; i += fragmentHeight) {
                for (int j = 0; j < fragmentWidth * 3; j += fragmentWidth) {
                    Rectangle rect = new Rectangle(j, i, fragmentWidth, fragmentHeight);
                    imgList.Add(img.Clone(rect, img.PixelFormat));
                }
            }
            return imgList;
        }
```

В конструкторе формы загрузим картинки, создадим объекты Graphics для каждого PictureBox и нарежем картинки:

```
hammers = new Bitmap("hammers.jpg");
wywh = new Bitmap("wywh.jpg");
pb1Graphics = pbImage1.CreateGraphics();
pb2Graphics = pbImage2.CreateGraphics();
pb3Graphics = pbImage3.CreateGraphics();
hammersSlices = SliceImage(hammers);
wywhSlices = SliceImage(wywh);
```

Добавим некоторые вспомогательные функции. Будем генерировать перестановки длины 9 из элементов от 0 до 8. Таких перестановок всего 362880. Перемешаем их для того, чтобы первой перестановкой в списке не была упорядоченная.

```
// генерация перестановок длины length из элементов list
https://stackoverflow.com/a/10630026
        private IEnumerable<IEnumerable<T>> GetPermutations<T>(IEnumerable<T>
list, int length)
        {
            if (length == 1) return list.Select(t => new T[] { t });
```

```
            return GetPermutations(list, length - 1)
                .SelectMany(t => list.Where(e => !t.Contains(e)),
                    (t1, t2) => t1.Concat(new T[] { t2 }));
        }

        // перемешивание полученых перестановок
https://stackoverflow.com/a/5807191
        private IEnumerable<T> Shuffle<T>(IEnumerable<T> enumerable)
        {
            var r = new Random();
            return enumerable.OrderBy(x => r.Next()).ToList();
        }

        // возвращает пару (значение, индекс) в коллекции
https://thomaslevesque.com/2019/11/18/using-foreach-with-index-in-c/
        private IEnumerable<(T item, int index)> WithIndex<T>(IEnumerable<T>
source)
        {
            return source.Select((item, index) => (item, index));
        }
```

Добавим функции, на основе которых создаются потоки. В них будем замерять время с помощью Stopwatch.

```
        private void hammersImgDoWork() {
            hammersStopWatch.Restart();
            GoThroughPermutations(pb1Graphics, hammersSlices);
            hammersTimeSpan = hammersStopWatch.Elapsed;
        }

        private void wywhImgDoWork() {
            wywhStopWatch.Restart();
            GoThroughPermutations(pb2Graphics, wywhSlices);
            wywhTimeSpan = wywhStopWatch.Elapsed;
        }
```

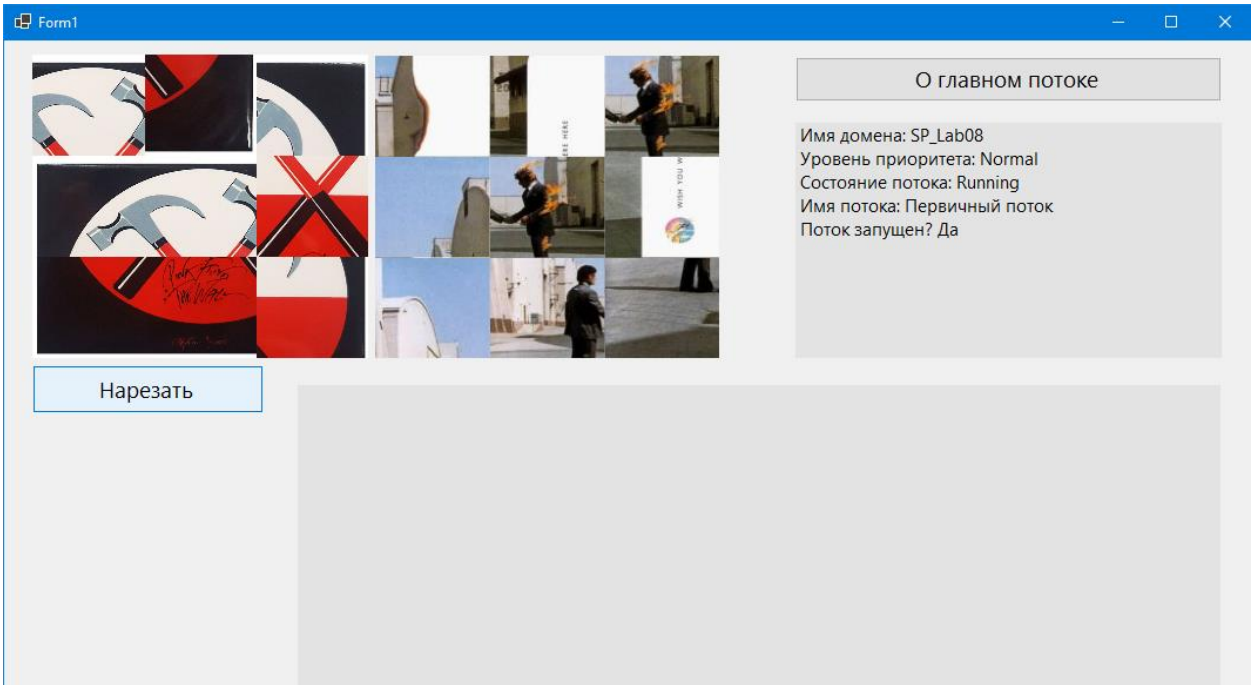
Создадим функции, которые вызывать создание перестановок, их перемешивание, проход по перестановкам и отрисовку частей картинки в порядке, указанном в перестановке. Для того, чтобы не происходила частая перерисовка, будем отрисовывать каждый 300 вариант нарезки.

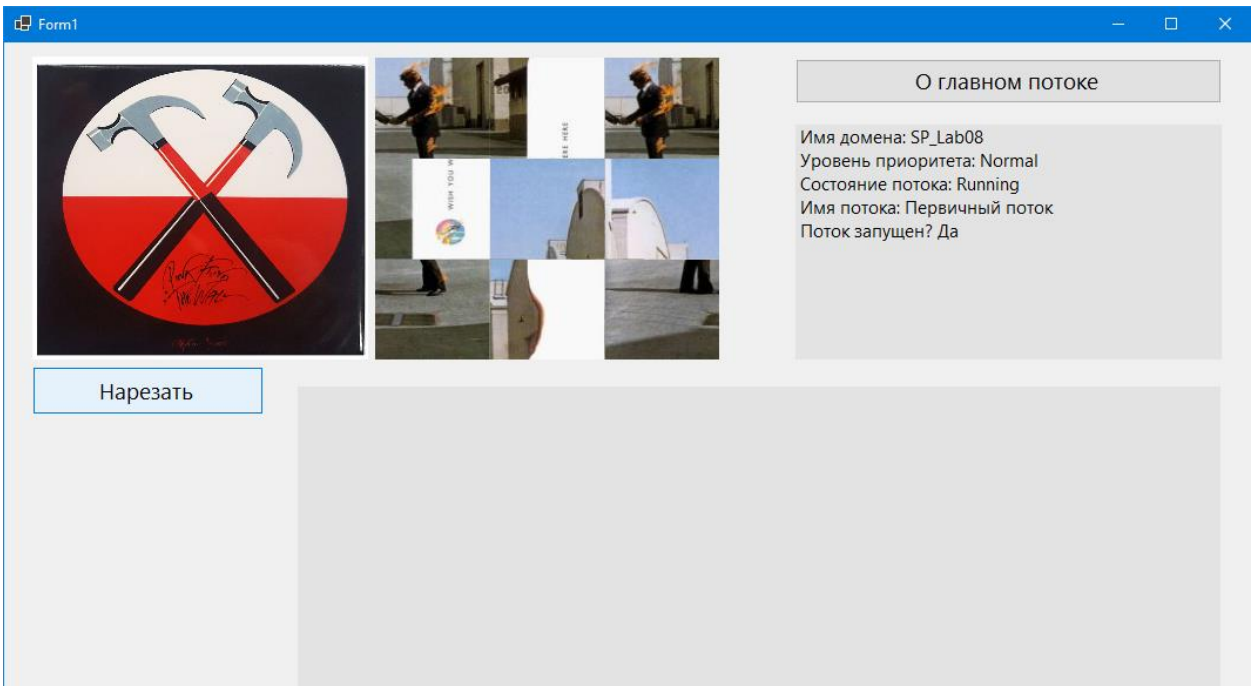
```
// отрисовка нарезанной картинки
        private void DrawSliced(Graphics g, List<Bitmap> imgList, List<int>
permutation) {
            // для укладывания картинки в размер отсечения
```

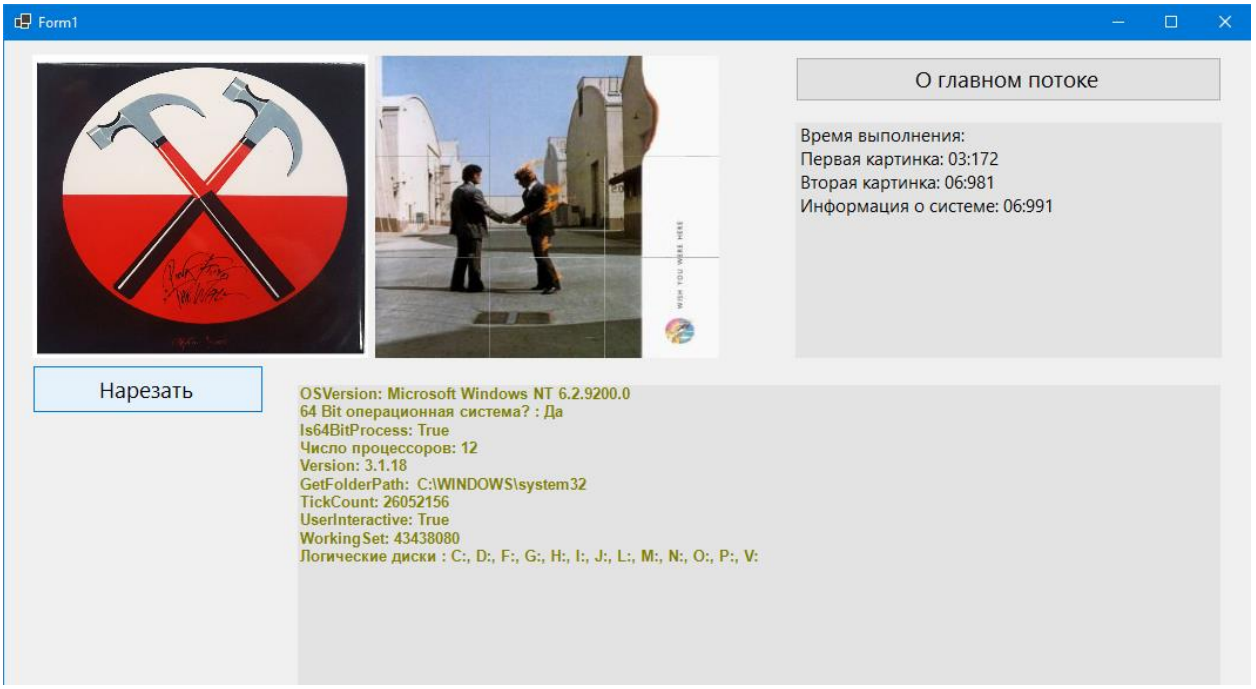
```
            var clipBounds = g.VisibleClipBounds;
            var visibleWidth = clipBounds.Width / 3;
            var visibleHeight = clipBounds.Height / 3;
            foreach (var (el, idx) in WithIndex(permutation)) {
                g.DrawImage(imgList[el], (idx % 3) * visibleWidth, (idx / 3) *
visibleHeight, visibleWidth, visibleHeight);
            }
        }


        private void GoThroughPermutations(Graphics g, List<Bitmap> imgList)
        {
            var trueOrder = Enumerable.Range(0, 9);
            // возвращает список из 9! = 362880 перестановок
            var permutations = GetPermutations(trueOrder, 9);
            permutations = Shuffle(permutations);
            foreach (var (el, idx) in WithIndex(permutations))
            {
                if (idx % 300 == 0) DrawSliced(g, imgList, el.ToList());
                if (el.SequenceEqual(trueOrder))
                {
                    DrawSliced(g, imgList, el.ToList());
                    break;
                }
            }
        }
```

3. Приложение должно выводить на экран эти «разрезанные» картинки на экран (в выбранном в потоке порядке).

Скриншоты работы программы:







4. Третий поток должен собрать информацию о компьютерной системе (не менее 10-ти параметров), подготовить картинку для демонстрации этой информации на экране (**в виде рекламы или презентации**) и ждать события о завершении двух предыдущих потоков (информация о системе должна быть представлена пользователю, когда произойдет завершение двух потоков).

Напишем функцию, на основе которой будет создаваться поток. В ней будем вызывать функцию сбора информации и замерять время. Для того, чтобы информация о системе вывелась только после того, как два потока

завершат работу, добавим вызовы методов Join (заблокируем этот поток):

```
        private void infoDoWork() {
            infoStopWatch.Restart();
            systemInfo = GetSystemInfo();
            hammersThread.Join();
            wywhThread.Join();
            pb3Graphics.DrawString(systemInfo, font, Brushes.Olive,
pb3Graphics.VisibleClipBounds);
            infoTimeSpan = infoStopWatch.Elapsed;
        }
```

Функция сбора информации:

```
private string GetSystemInfo()
        {
            var output = new StringBuilder();
            output.AppendFormat("OSVersion: {0}\n", Environment.OSVersion);
            output.AppendFormat("64 Bit операционная система? : {0}\n",
                Environment.Is64BitOperatingSystem ? "Да" : "Нет");
            output.AppendFormat("Is64BitProcess: {0}\n",
Environment.Is64BitProcess);
            output.AppendFormat("Число процессоров: {0}\n",
Environment.ProcessorCount);
            output.AppendFormat("Version: {0}\n",
Environment.Version.ToString());
            output.AppendFormat("GetFolderPath: {0}\n",
Environment.GetFolderPath(Environment.SpecialFolder.System));
            output.AppendFormat("TickCount: {0}\n", Environment.TickCount);
            output.AppendFormat("UserInteractive: {0}\n",
Environment.UserInteractive);
            output.AppendFormat("WorkingSet: {0}\n", Environment.WorkingSet);
            output.AppendFormat("Логические диски : {0}\n",
                string.Join(", ", Environment.GetLogicalDrives())
                    .Replace("\\", string.Empty));
            return output.ToString();
        }
```

По кнопке будут запускаться все три потока. Так как поток со сбором информации ждет завершения двух других потоков, будем ждать только его.

```
        private void btnGoSlicing_Click(object sender, EventArgs e)
        {
            hammersThread = new Thread(hammersImgDoWork);
            wywhThread = new Thread(wywhImgDoWork);
            infoThread = new Thread(infoDoWork);

            hammersThread.Start();
```

```
            wywhThread.Start();
            infoThread.Start();


            infoThread.Join();

            string hammersTimeInfo = String.Format("Первая картинка:
{0:00}:{1:000}", hammersTimeSpan.Seconds, hammersTimeSpan.Milliseconds);
            string wywhTimeInfo = String.Format("Вторая картинка:
{0:00}:{1:000}", wywhTimeSpan.Seconds, wywhTimeSpan.Milliseconds);
            string infoTimeInfo = String.Format("Информация о системе:
{0:00}:{1:000}", infoTimeSpan.Seconds, infoTimeSpan.Milliseconds);
            labelAboutMainThread.Text = "Время выполнения:" + "\n" +
hammersTimeInfo + "\n" + wywhTimeInfo + "\n" + infoTimeInfo;


        }
```

```
OSVersion: Microsoft Windows NT 6.2.9200.0
64 Bit операционная система? : Да
Is64BitProcess: True
Число процессоров: 12
Version: 3.1.18
GetFolderPath:  C:\WINDOWS\system32
TickCount: 23155656
UserInteractive: True
WorkingSet: 43520000
Логические диски : C:, D:, F:, G:, H:, I:, J:, L:, M:, N:, O:, P:, V:
```

5. Программа должна также предоставить пользователю информацию о времени, в течении которого выполнялись потоки.

   Время выполнения потока сбора информации указана с учетом ожидания двух других потоков.

```
Время выполнения:
Первая картинка: 03:172
Вторая картинка: 06:981
Информация о системе: 06:991
```