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

Отчет к лабораторной работе № 6

Дисциплина: «Системное программирование»

Михаил Чуворкин
25.11.2021

Лабораторная работа 6

Основы GDI+

Подготовка к лабораторной работе

Для выполнения лабораторной работы необходимо ознакомиться (по литературе или в сети) и сделать краткое описание следующих функций и параметров, а также используемых событий.

1). Дескриптор контекста устройства и функции для его получения:

- **e.Graphics**.

  Класс Graphics предоставляет методы рисования на устройстве отображения. Объект Graphics используется для рисования. Объект класса Graphics предоставляет методы для рисования объектов на устройстве отображения.

- **this.CreateGraphics()**;

  Создает объект Graphics для элемента управления.

- **Graphics.FromHwnd()**.

  Создает новый объект Graphics из указанного дескриптора окна.

2). Выбор и создание кисти:

- **System.Drawing.Brushes**

  Кисти для каждого из стандартных цветов.

- **SolidBrush;**

  Определяет кисть одного цвета. Кисти используются для заливки графических фигур, таких как прямоугольники, эллипсы, круги, многоугольники и пути.

- **System. Drawing. Drawing2D. HatchBrush;**

  Определяет прямоугольную кисть с стиль штриховки, цвет фона и цвет переднего плана.

- **System.Drawing.TextureBrush;**

  Каждое свойство класса TextureBrush является объектом Brush, использующим изображение для заливки внутренней части формы

- **LinearGradientBrush;**

  Инкапсулирует объект Brush с линейным градиентом.

3). Выбор и создание карандаша.

Стандартные карандаши: **System.Drawing.Pens**

Определяет объект, используемый для рисования прямых линий и кривых.

Создание карандаша. Например: p = **new Pen**(Color.Aquamarine, 3);

Конструкторы:

- Pen(Brush) – с параметрами переданной кисти
- Pen(Brush, Single) – с указанием размера
- Pen(Color) – с указанием цвета
- Pen(Color, Single) – с указанием цвета и размера

Ширина: p.**Width** = 5;

```
public float Width { get; set; }
```

Возвращает или устанавливает ширину пера Pen, в единицах измерения объекта Graphics, используемого для рисования.

Цвет: p.**Color** = Color.Coral;

```
public System.Drawing.Color Color { get; set; }
```

Возвращает или задает цвет объекта Pen.

Стиль: p.**DashStyle** = System.Drawing.Drawing2D.DashStyle.DashDot;

```
public System.Drawing.Drawing2D.DashStyle DashStyle { get; set; }
```

Возвращает или задает стиль, используемый для пунктирных линий, нарисованных при помощи объекта Pen.

4). Функции рисования:

DrawArc

Рисует дугу, которая является частью эллипса, заданного парой координат, шириной и высотой.

`DrawArc(Pen, Rectangle, Single, Single)`	Рисует дугу, которая является частью эллипса, заданного структурой Rectangle.
`DrawArc(Pen, RectangleF, Single, Single)`	Рисует дугу, которая является частью эллипса, заданного структурой RectangleF.
`DrawArc(Pen, Int32, Int32, Int32, Int32, Int32, Int32)`	Рисует дугу, которая является частью эллипса, заданного парой координат, шириной и высотой.
`DrawArc(Pen, Single, Single, Single, Single, Single, Single)`	Рисует дугу, которая является частью эллипса, заданного парой координат, шириной и высотой.

DrawLine

Рисует линию, соединяющую две точки, задаваемые парами координат.

`DrawLine(Pen, PointF, PointF)`	Проводит линию, соединяющую две структуры PointF.
`DrawLine(Pen, Int32, Int32, Int32, Int32)`	Проводит линию, соединяющую две точки, задаваемые парами координат.

DrawLine(Pen, Single, Single, Single, Single)	Проводит линию, соединяющую две точки, задаваемые парами координат.
DrawLine(Pen, Point, Point)	Проводит линию, соединяющую две структуры Point.

DrawEllipse

Рисует эллипс, определяемый ограничивающим прямоугольником, заданным с помощью пары координат, ширины и высоты.

DrawEllipse(Pen, Rectangle)	Рисует эллипс, определяемый ограничивающей структурой Rectangle.
DrawEllipse(Pen, RectangleF)	Рисует эллипс, определяемый ограничивающей структурой RectangleF.
DrawEllipse(Pen, Int32, Int32, Int32, Int32)	Рисует эллипс, определяемый ограничивающим прямоугольником, заданным с помощью координат верхнего левого угла прямоугольника, высоты и ширины.
DrawEllipse(Pen, Single, Single, Single, Single)	Рисует эллипс, определяемый ограничивающим прямоугольником, заданным с помощью пары координат, ширины и высоты.

DrawRectangle

Рисует прямоугольник, определяемый парой координат, шириной и высотой.

DrawRectangle(Pen, Rectangle)	Рисует прямоугольник, определяемый структурой Rectangle.
DrawRectangle(Pen, Int32, Int32, Int32, Int32)	Рисует прямоугольник, определяемый парой координат, шириной и высотой.
DrawRectangle(Pen, Single, Single, Single, Single)	Рисует прямоугольник, определяемый парой координат, шириной и высотой.

DrawPolygon

DrawPolygon(Pen, PointF[])	Рисует многоугольник, определяемый массивом структур PointF.
DrawPolygon(Pen, Point[])	Рисует многоугольник, определяемый массивом структур Point.

DrawImage

Рисует указанный объект Image или его часть в заданном месте, используя исходный размер или заданный размер и форму (в зависимости от передаваемых аргументов). Имеет множество перегрузок, см. здесь.

DrawString

Создает заданную текстовую строку в указанном месте с помощью заданных объектов Brush и Font.

DrawString(String, Font, Brush, Single, Single, StringFormat)	Создает заданную текстовую строку в указанном месте с помощью заданных объектов Brush и Font, используя атрибуты форматирования заданного формата StringFormat.
DrawString(String, Font, Brush, RectangleF, StringFormat)	Создает заданную текстовую строку в указанном прямоугольнике с помощью заданных объектов

	Brush и Font, используя атрибуты форматирования заданного формата StringFormat.
DrawString(String, Font, Brush, Single, Single)	Создает заданную текстовую строку в указанном месте с помощью заданных объектов Brush и Font.
DrawString(String, Font, Brush, RectangleF)	Создает заданную текстовую строку в указанном прямоугольнике с помощью заданных объектов Brush и Font.
DrawString(String, Font, Brush, PointF)	Создает заданную текстовую строку в указанном месте с помощью заданных объектов Brush и Font.
DrawString(String, Font, Brush, PointF, StringFormat)	Создает заданную текстовую строку в указанном месте с помощью заданных объектов Brush и Font, используя атрибуты форматирования заданного формата StringFormat.

FillEllipse

Заполняет внутреннюю часть эллипса, определяемого ограничивающим прямоугольником, заданным с помощью пары координат, ширины и высоты.

FillEllipse(Brush, Int32, Int32, Int32, Int32)	Заполняет внутреннюю часть эллипса, определяемого ограничивающим прямоугольником, заданным с помощью пары координат, ширины и высоты.
FillEllipse(Brush, Rectangle)	Заполняет внутреннюю часть эллипса, определяемого ограничивающим прямоугольником, который задан структурой Rectangle.
FillEllipse(Brush, RectangleF)	Заполняет внутреннюю часть эллипса, определяемого ограничивающим прямоугольником, который задан структурой RectangleF.
FillEllipse(Brush, Single, Single, Single, Single)	Заполняет внутреннюю часть эллипса, определяемого ограничивающим прямоугольником, заданным с помощью пары координат, ширины и высоты.

FillRectangle

Заполняет внутреннюю часть прямоугольника, который задается парой координат, шириной и высотой.

FillRectangle(Brush, Rectangle)	Заполняет внутреннюю часть прямоугольника, определяемого структурой Rectangle.
FillRectangle(Brush, RectangleF)	Заполняет внутреннюю часть прямоугольника, определяемого структурой RectangleF.
FillRectangle(Brush, Int32, Int32, Int32, Int32)	Заполняет внутреннюю часть прямоугольника, который задается парой координат, шириной и высотой.
FillRectangle(Brush, Single, Single, Single, Single)	Заполняет внутреннюю часть прямоугольника, который задается парой координат, шириной и высотой.

FillPolygon

Заполняет внутреннюю часть многоугольника, определяемого массивом точек, заданных структурами Point.

FillPolygon(Brush, Point[])	Заполняет внутреннюю часть многоугольника, определяемого массивом точек, заданных структурами Point.
FillPolygon(Brush, PointF[])	Заполняет внутреннюю часть многоугольника, определяемого массивом точек, заданных структурами PointF.
FillPolygon(Brush, Point[], FillMode)	Заполняет внутреннюю часть многоугольника, который определяется массивом точек, заданных

	структурами Point, используя указанный режим заливки.
FillPolygon(Brush, PointF[], FillMode)	Заполняет внутреннюю часть многоугольника, который определяется массивом точек, заданных структурами PointF, используя указанный режим заливки.

FillRegion

Заполняет внутреннюю часть объекта Region с использованием кисти Brush.

```
public void FillRegion (System.Drawing.Brush brush,
System.Drawing.Region region);
```

5). Прямоугольники, регионы, область рисования.

Rectangle r = new Rectangle(100,100,50,150);

Содержит набор из четырех целых чисел, определяющих расположение и размер прямоугольника.

Rectangle(Int32, Int32, Int32, Int32)	Инициализирует новый экземпляр класса Rectangle заданным расположением и размером.
Rectangle(Point, Size)	Инициализирует новый экземпляр класса Rectangle заданным расположением и размером. Point – левый верхний угол, Size – размер прямоугольника

Region reg = new Region(r);

Описывает внутреннюю часть графической формы, состоящей из прямоугольников и контуров.

Region(Rectangle)	Инициализирует новую область Region из указанной структуры Rectangle.
Region(RectangleF)	Инициализирует новую область Region из указанной структуры RectangleF.

Свойство **Clip** контекста устройства

```
public System.Drawing.Region Clip { get; set; }
```

Возвращает или задает объект Region, ограничивающий область рисования данного объекта Graphics.

6). Область отсечения и обновление окна: методы Button.**Refresh**() и **Invalidate();**

`Control.Refresh()`

Принудительно создает условия, при которых элемент управления делает недоступной свою клиентскую область и немедленно перерисовывает себя и все дочерние элементы. Button наследует этот метод от Control.

`Control.Invalidate()`

Делает недействительной указанную область элемента управления (добавляет ее к области обновления элемента, которая будет перерисована при следующей операции рисования) и вызывает отправку сообщения рисования элементу управления. При необходимости объявляет недействительными назначенные

элементу управления дочерние элементы. Если вызывается без аргументов, то перерисовывается вся поверхность элемента Control.

7). Цвет (три и четыре параметра)

```
Color col = Color.FromArgb(255,255,100);
Color col = Color.FromArgb(100,255,255,100);
```

FromArgb(Int32, Int32, Int32, Int32)	Создает структуру Color из четырех значений компонентов ARGB (альфа, красный, зеленый и синий). Хотя и этот метод позволяет передать 32-разрядное значение для каждого компонента, значение каждого из них ограничено 8 разрядами.
FromArgb(Int32, Int32, Int32)	Создает структуру Color из указанных 8-разрядных значений цветов (красный, зеленый, синий). Значение альфа неявно определено как 255 (полностью непрозрачно). Хотя и этот метод позволяет передать 32-разрядное значение для каждого компонента цвета, значение каждого из них ограничено 8 разрядами.

- Метод Clear - обновление области рисования и заливка заданным цветом

```
Graphics.Clear(Color)
```

Задание

Для выполнения работы создать приложение на языке C#.

1. Придумать маленькую эмблему своей программы и нарисовать ее на всех кнопках. Картинка не должна исчезать после сворачивания или перекрытия окна.

В качестве эмблемы будем использовать картинку: 

Для нее создадим объект: `static Image emblem;`

Флаг: `bool emblemFlag;`

В конструкторе формы загрузим ее: `emblem = Image.FromFile("playbtn.png");`

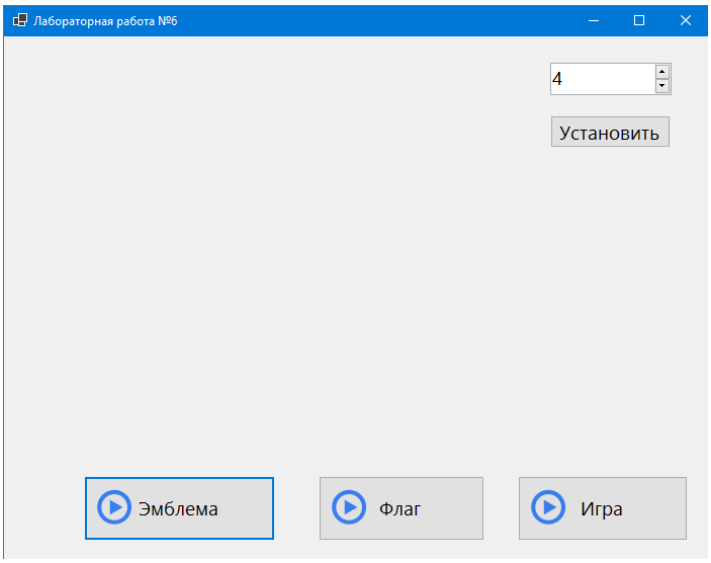
Создадим кнопку на форме и напишем обработчик нажатия на нее:

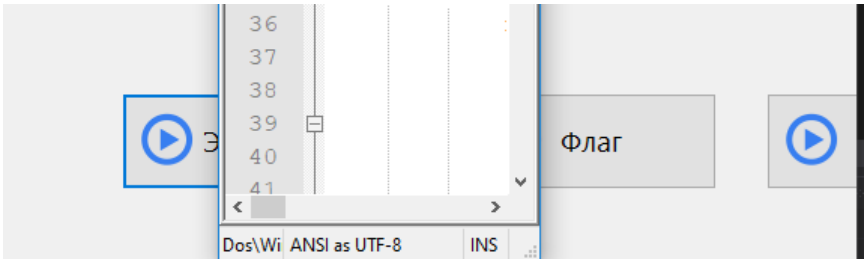
```
private void btnEmblem_Click(object sender, EventArgs e)
{
    emblemFlag = !emblemFlag;
    olympicFlag = gameFlag = false;
    Refresh(); // чтобы вызвалсь Paint кнопок
}
```

В обработчиках Paint кнопок напишем рисование картинки:

```
private void btnEmblem_Paint(object sender, PaintEventArgs e)
```

```
{
    if (emblemFlag) e.Graphics.DrawImage(emblem, 10, 10, 40, 40);
}

private void btnOlympic_Paint(object sender, PaintEventArgs e)
{
    if (emblemFlag) e.Graphics.DrawImage(emblem, 10, 10, 40, 40);
}

private void btnGame_Paint(object sender, PaintEventArgs e)
{
    if (emblemFlag) e.Graphics.DrawImage(emblem, 10, 10, 40, 40);

}
```





2. Нарисовать флаг олимпийских игр.

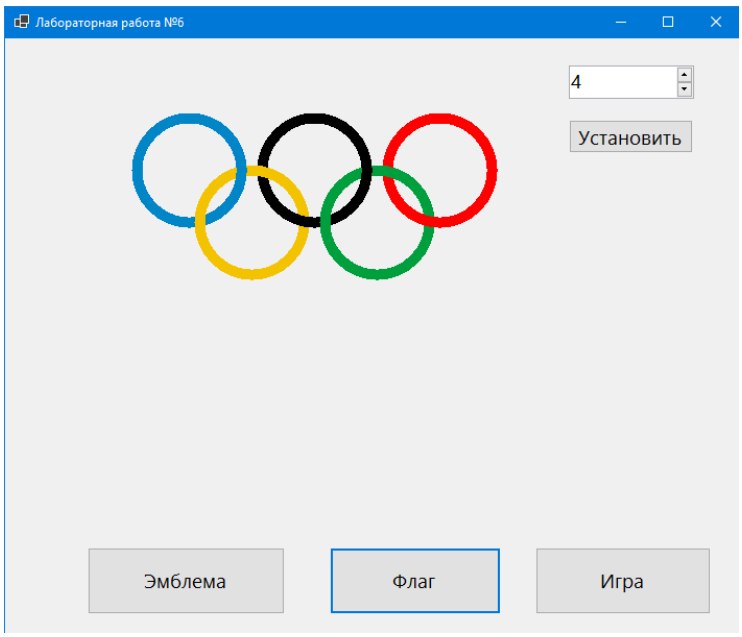
Аналогично заданию 1 создадим кнопку, флаг и обработчик:

```
private void btnOlympic_Click(object sender, EventArgs e)
{
    olympicFlag = !olympicFlag;
    emblemFlag = gameFlag = false;
    Refresh(); // чтобы нарисовать в picturebox и убрать эмблему с кнопок

 }
```

Добавим обработчик Paint и напишем в нем рисование колец с учетом их наложения:

```
private void pictureBox1_Paint(object sender, PaintEventArgs e)
{
```

```
        if (olympicFlag)
        {

            int radius = 50;
            int shift = 10;
            int penSize = 10;

            Pen pen = new Pen(Color.FromArgb(0, 133, 199), penSize);
            Rectangle rect = new Rectangle(radius, radius, radius * 2, radius * 2);
            e.Graphics.DrawEllipse(pen, rect);

            pen.Color = Color.FromArgb(244, 195, 0);
            rect.X += radius + shift;
            rect.Y = radius * (1 + 3 / 2);
            e.Graphics.DrawEllipse(pen, rect);

            pen.Color = Color.Black;
            rect.X += radius + shift;
            rect.Y = radius;
            e.Graphics.DrawEllipse(pen, rect);

            pen.Color = Color.FromArgb(0, 159, 61);
            rect.X += radius + shift;
            rect.Y = radius * (1 + 3 / 2);
            e.Graphics.DrawEllipse(pen, rect);

            pen.Color = Color.Red;
            rect.X += radius + shift;
            rect.Y = radius;
            e.Graphics.DrawEllipse(pen, rect);

            // Зарисовка наложений колец
            pen.Color = Color.FromArgb(0, 133, 199);
            rect.X = radius;
            rect.Y = radius;
            e.Graphics.DrawArc(pen, rect, 340, 40);

            pen.Color = Color.FromArgb(244, 195, 0);
            rect.X += radius + shift;
            rect.Y = radius * (1 + 3 / 2);
            e.Graphics.DrawArc(pen, rect, 270, 40);

            pen.Color = Color.Black;
            rect.X += radius + shift;
            rect.Y = radius;
            e.Graphics.DrawArc(pen, rect, 340, 40);

            pen.Color = Color.FromArgb(0, 159, 61);
            rect.X += radius + shift;
            rect.Y = radius * (1 + 3 / 2);
            e.Graphics.DrawArc(pen, rect, 270, 40);

        }
    }
```



3. Создать программу «Игра», похожую на крестики-нолики. Игровое поле – клетки-прямоугольники. При щелчке на клетке визуализируется ход игрока номер 1, следующий щелчок совершает игрок номер 2. Выигрывает тот, кто выбрал 4 клетки подряд. При создании интерфейса для игры использовать текстурные и градиентные кисти, а также режим прозрачности.

Для игры создадим класс Game. При нажатии на кнопку Игра будем устанавливать соответствующий флаг и создавать экземпляр класса Game. Также он будет создаваться при клике на кнопку Установить, которая задает размер игрового поля (и перезапускает игру). Размер поля будет вводится через элемент NumericUpDown.

Обработчик нажатия на кнопку Игра:

```
 private void btnGame_Click(object sender, EventArgs e)
 {
     gameFlag = !gameFlag;
     emblemFlag = olympicFlag = false;
     int bSize = (int)boardSizeUD.Value;
     if (bSize < 4)
     {
         bSize = 4;
         boardSizeUD.Value = 4;
     }
     else if (bSize > 10)
     {
         bSize = 10;
         boardSizeUD.Value = 10;
     }
     Refresh(); // очищаем от предыдущих действий
     if (gameFlag) {
         game = new Game(bSize, pictureBox1);
         game.main = this; // для пересылки информации о состоянии игры
     }
}
```

```
private void btnSetBoardSize_Click(object sender, EventArgs e)
{
    if (gameFlag)
    {
        int bSize = (int)boardSizeUD.Value;
        if (bSize < 4)
        {
            bSize = 4;
            boardSizeUD.Value = 4;
        }
        else if (bSize > 10) {
            bSize = 10;
            boardSizeUD.Value = 10;
        }

        Refresh();
        game = new Game(bSize, pictureBox1);
        game.main = this; // для пересылки информации о состоянии игры
    }
}
```

По клику мышкой по полю, будем вызывать метод BoardClicked и передавать в него координаты мыши относительно PictureBox:

```
private void pictureBox1_MouseClick(object sender, MouseEventArgs e)
{
    if (gameFlag) game.BoardClicked(e.X, e.Y);
}
```

Метод для отображения сообщений игры:

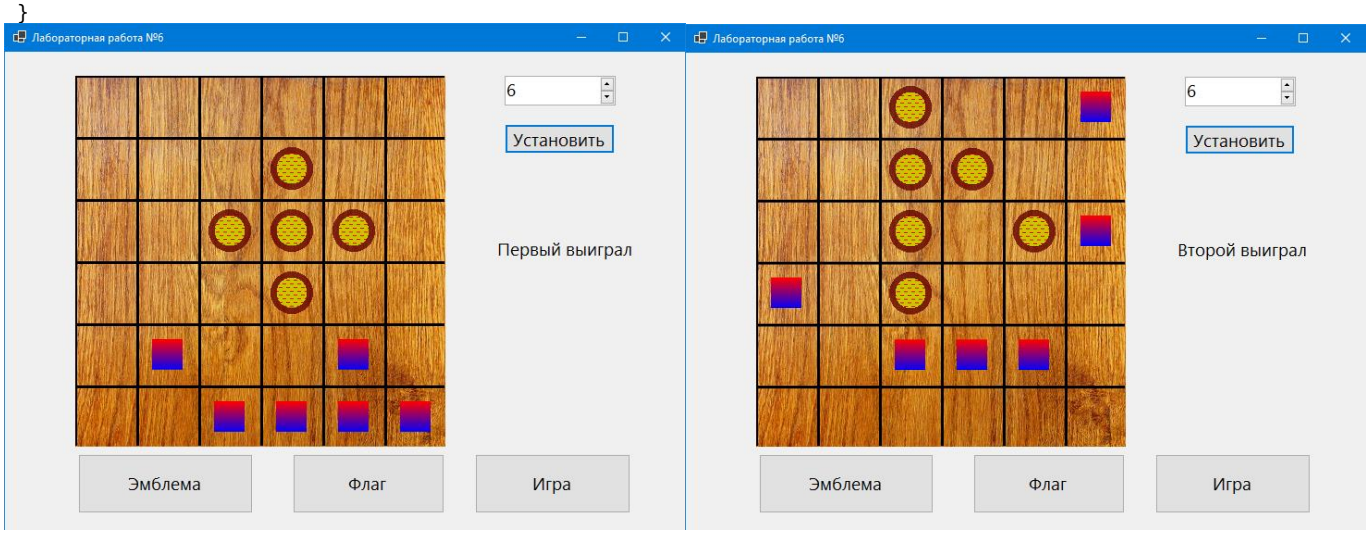
```
public void labelShow(string message) {
    labelGame.Text = message;
}
```

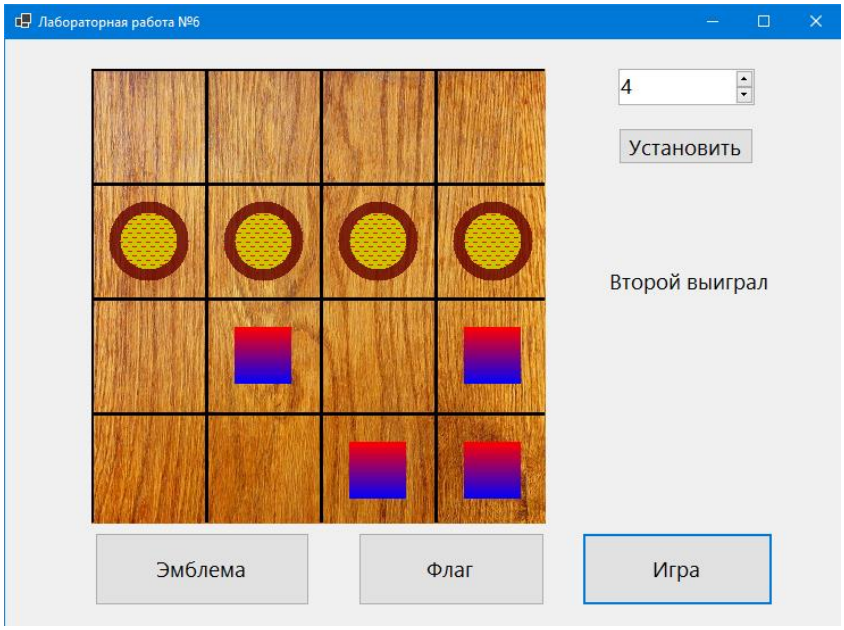
Класс Game:

```
public class Game
    {
        // данные игры

        // состояния - Игра продолжается, Первый игрок выиграл, Второй игрок выиграл
        enum State { Playing, Win1, Win2}
        State _currentState; // текущее состояние игры
        public MainForm main;
        int _currentPlayer;
        int[,] _board; // матрица с ходами
        int _boardSize; // размер поля (кол-во клеток)
        Rectangle _boardRect; // прямоугольник поля
        int _xSize, _ySize; // размеры клетки поля
        Graphics _g; // для рисования на поверхности поля

        public Game(int boardSize, PictureBox pbox) {
            _g = Graphics.FromHwnd(pbox.Handle);
            _boardSize = boardSize;
            _boardRect = pbox.ClientRectangle;
            _xSize = _boardRect.Width / _boardSize;
            _ySize = _boardRect.Height / _boardSize;
            _currentPlayer = 1;
            _board = new int[_boardSize, _boardSize];
            Array.Clear(_board, 0, _board.Length);
            _currentState = State.Playing;
            PaintBg();
```

```
        }

        // рисование фона игрового поля и сетки
        private void PaintBg()
        {
            Bitmap image1 = (Bitmap)Image.FromFile("wood.jpg", true);
            TextureBrush texture = new TextureBrush(image1);
            _g.FillRectangle(texture, _boardRect); // фон - текстурная кисть
            Pen pen = new Pen(Color.Black, 3);
            for (int i = 0; i < _boardSize; i++)
            {
                _g.DrawLine(pen, _xSize * i + i, 0, _xSize * i + i, _boardRect.Width - 1);
                _g.DrawLine(pen, 0, _ySize * i + i, _boardRect.Height - 1, _ySize * i + i);
            }
        }

        // определение победителя: кто первый соберет 4 клетки в столбце или строке
        private int CheckWinner(string key) {
            for (int j = 0; j < _board.GetLength(key == "row" ? 1 : 0); ++j) {
                int pred = key == "row" ? _board[0, j] : _board[j, 0]; // выбор начальной клетки
                int curPlayer = pred;
                int counter = 0;
                for (int i = 1; i < _board.GetLength(key == "col" ? 1 : 0); ++i) {
                    int cur = key == "row" ? _board[i, j] : _board[j, i];
                    if (pred == cur && pred != 0) // ряд/столбец продолжается
                    {
                        counter++;
                    }
                    else // ряд/столбец прерван
                    {
                        counter = 0;
                        curPlayer = cur;
                    }
                    pred = cur;
                    if (counter == 3) return curPlayer;
                }
            }
            return 0;
        }

        // кликнули по полю в точку {x, y}
        public void BoardClicked(int x, int y) {

            // определяем клетку, в которой кликнули
            int xCell = x / _xSize;
            int yCell = y / _ySize;

            // рисование фигуры, соответствующей игроку и изменение матрицы совершенных ходов
            Rectangle markerRect = new Rectangle(xCell * (_xSize + 1) + _xSize / 4, yCell * (_ySize + 1) + _ySize / 4,
_xSize / 2, _ySize / 2);
            if (_currentPlayer == 1) { // ход совершает первый игрок
                if (_board[xCell, yCell] == 0 && _currentState == State.Playing) {
                    LinearGradientBrush linGrBrush = new LinearGradientBrush(
                       markerRect,
                       Color.Red,
                       Color.Blue,
                       90);
                    _g.FillRectangle(linGrBrush, markerRect);
                    _board[xCell, yCell] = _currentPlayer; // пишем в ячейку матрицы номер игрока
                    _currentPlayer = 2; // переход ко второму игроку
                    main.labelShow("Ходит второй");
                }
                else main.labelShow("Занято");
            }
            else if (_currentPlayer == 2) { // ход совершает второй игрок
                if (_board[xCell, yCell] == 0 && _currentState == State.Playing) {
                    int penSize = _xSize / 5;
                    Pen pen = new Pen(Color.FromArgb(200, 100, 0, 0), penSize);
                    HatchBrush brush = new HatchBrush(
                        HatchStyle.DashedHorizontal,
                        Color.Red,
                        Color.FromArgb(255, 200, 200, 0));
                    _g.DrawEllipse(pen, markerRect);
                    _g.FillEllipse(brush, markerRect);
                    _board[xCell, yCell] = _currentPlayer; // пишем в ячейку матрицы номер игрока
                    _currentPlayer = 1; // переход к первому игроку
                    main.labelShow("Ходит первый");
```

```
            }
            else main.labelShow("Занято");
        }

        // определение победителя
        int winnerRows = CheckWinner("row");
        int winnerColls = CheckWinner("col");
        int winner = winnerRows == 0 ? winnerColls : winnerRows;
        switch (winner) {
            case 1:
                {
                    main.labelShow("Первый выиграл");
                    _currentState = State.Win1;
                    break;
                }
            case 2:
                {
                    main.labelShow("Второй выиграл");
                    _currentState = State.Win2;
                    break;
                }
        }

    }
}
```





Полный код программы доступен здесь: https://pastebin.com/mnhK5aKY