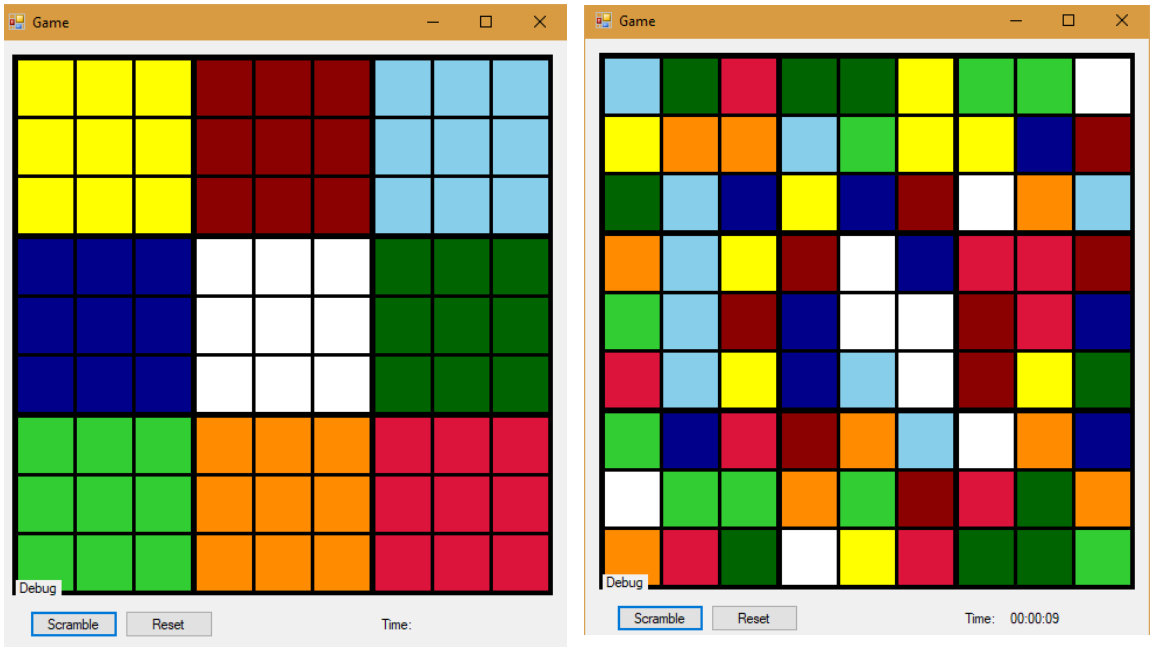
ПРОЕКТНА ЗАДАЧА ПО ПРЕДМЕТОТ ВИЗУЕЛНО ПРОГРАМИРАЊЕ

Ѓорѓи Каламерников 161198

Александра Манзаклиева 161112

Twisty Sudoku

1. Опис на апликацијата
   Апликацијата што ја развиваме е играта Rubik's Cube имплементирана во форма на судоку. Соодветно на класичната Рубикова коцка шесте бои (квадрати) се претставени со шест 3x3 подматрици кои ја составуваат 9x9 матрицата во играта судоку.
2. Упатство за користење
   Со клик на копчето Scrumble се размешуваат полињата . Целта е секој квадрант да биде обоен во една боја (како кај рубиковата коцка). Не е битно кој квадрант која боја ќе биде. Потезите се состојат од кликнување и повлекување на одредена колона или редица за произволен број полиња. Со самото кликнување на Scrumble започнува и мерење на времето на решавање. Доколку сакате да започнете одново тогаш со клик на копчето Reset се ресетира времето и полињата одново се размешуваат.



3. Претставување на проблемот
   Сите податоци и функции се организирани во класи. Во класата `public partial class Canvas : Form, Monitor` дефинирани функции кои се основни за играта но повикуваат

други функции неопходни за правилно функционирање, кои пак се дефинирани во соодветните класи.

Во класата `class Cube : Monitor` се дефинирани функции за справување со движењата на маусот(`public void handleDown(int x, int y)`, `public void handleUp(int x, int y)` , `public void handleMove(int x, int y)`). Исто така имплементирана и е и функцијата `public void draw(Graphics g, int width, int height)` која служи за исцртување и обојување на матрицата.

```
public void draw(Graphics g, int width, int height)
        {
            g.FillRectangle(Brushes.Black, 0, 0, width, height);
            foreach(Sticker[] array in matrix)
            {
                foreach(Sticker sticker in array)
                {
                    sticker.draw(g, width, height);
                }
            }
           }
```

Обојувањето на секое од полињата се врши во foraech циклусот каде се повикува функцијата `public void draw(Graphics g, int width, int height)` од класата `Sticker.`

```
public void draw(Graphics g, int width, int height)
        {
            Brush br = new SolidBrush(color);
            sizeX = (int) Math.Round((width - 38) / 9F);
            sizeY = (int) Math.Round((height - 38) / 9F);
            upRight = new Point((position.X * sizeX) + blankspace(position.X),
(position.Y * sizeY) + blankspace(position.Y));

            int posX = (upRight.X + offsetX);
            if (posX < 0)
            {
                posX += width;
            }
            posX %= width;
            int posY = (upRight.Y + offsetY);
            if (posY < 0)
            {
                posY += height;
            }
            posY %= height;
            g.FillRectangle(br, posX, posY, sizeX, sizeY);

            if (posX + sizeX > width)
            {
                g.FillRectangle(br, 0, posY, (posX + sizeX) - width, sizeY);
            }
            if (posY + sizeY > width)
            {
                g.FillRectangle(br, posX, 0, sizeX, (posY + sizeY) - height);
            }
            br.Dispose();
           }
```

Во класата `Matrices` ваљсе врши иницијално обојување на матрицата кое го дава изгледот на едно од можните решенија. Исто така се врши и проверка дали проблемот е решен. Овие две работи ги вршат следните функции соодветно.

```
public static void initialize(Sticker[][] matrix, Observer o)
        {
            for (int i = 0; i < 9; i++)
            {
                for (int j = 0; j < 9; j++)
                {
                    if (i < 3)
                    {
                        if (j < 3)
                        {
                            matrix[i][j] = new Sticker(Color.Yellow, new Point(i, j), o);
                        }
                        else if (j >= 3 && j < 6)
                        {
                            matrix[i][j] = new Sticker(Color.DarkBlue, new Point(i, j),
o);
                        }
                        else if (j >= 6)
                        {
                            matrix[i][j] = new Sticker(Color.LimeGreen, new Point(i, j),
o);
                        }
                    }
                    else if (i >= 3 && i < 6)
                    {
                        if (j < 3)
                        {
                            matrix[i][j] = new Sticker(Color.DarkRed, new Point(i, j),
o);
                        }
                        else if (j >= 3 && j < 6)
                        {
                            matrix[i][j] = new Sticker(Color.White, new Point(i, j), o);
                        }
                        else if (j >= 6)
                        {
                            matrix[i][j] = new Sticker(Color.DarkOrange, new Point(i, j),
o);
                        }
                    }
                    else if (i >= 6)
                    {
                        if (j < 3)
                        {
                            matrix[i][j] = new Sticker(Color.SkyBlue, new Point(i, j),
o);
                        }
                        else if (j >= 3 && j < 6)
                        {
                            matrix[i][j] = new Sticker(Color.DarkGreen, new Point(i, j),
o);
                        }
                        else if (j >= 6)
```

```
                        {
                            matrix[i][j] = new Sticker(Color.Crimson, new Point(i, j),
o);
                        }
                    }
                }
            }
        }
        public static bool isSolved(Sticker[][] matrix)
        {
            int[] centers = { 1, 4, 7 };
            int[] offsetsX = { 1, 1,  1, 0,  0, -1, -1, -1 };
            int[] offsetsY = { 1, 0, -1, 1, -1,  1,  0, -1 };
            for(int i=0; i<3; i++)
            {
                for(int j=0; j<3; j++)
                {
                    for(int k=0; k<8; k++)
                    {
                        int x = centers[i] + offsetsX[k];
                        int y = centers[j] + offsetsY[k];
                        if(matrix[x][y].color != matrix[centers[i]][centers[j]].color)
                        {
                            return false;
                        }
                    }
                }
            }
            return true;
        }
    }
    }
```

Функционирањето на тајмерот е обезбедено со фунцијата

```
private void timer1_Tick(object sender, EventArgs e)
        {
            i++;
            TimeSpan i_minutes = TimeSpan.FromSeconds(i);
            time.Text = String.Format("{0:D2}:{1:D2}:{2:D2}", i_minutes.Hours,
i_minutes.Minutes, i_minutes.Seconds);
                }
```

каде што променливата i е глобална променлива иницијализирана на 0. Истата се инкрементира и ги мери секундите . Времето на решавање потоа се испишува во соодветен формат часови:минути:секунди.

Доколку се дојде до решение тајмерот се стопира и се испишува порака со времето потребно за решавање. Ова е овозможено во следната функција во класата Canvas.

```
private void panel_MouseUp(object sender, MouseEventArgs e)
        {
            theCube.handleUp(e.X, e.Y);
            if (theCube.isSolved())
            {
                timer1.Stop();
```

```
                TimeSpan i_minutes = TimeSpan.FromSeconds(i);
                MessageBox.Show("Congratulations! Your solving time is: " +
                    String.Format("{0:D2}:{1:D2}:{2:D2}", i_minutes.Hours,
i_minutes.Minutes, i_minutes.Seconds)
                    + "\n Click SCRAMBLE to start again.");
                i = 0;
                time.Text = "";
            }
           }
```