Sveučilište u Zagrebu

Prirodoslovno- matematički fakultet

Matematički odjel

RAČUNARSKI PRAKTIKUM 3

Projektni zadatak: PUZZLE

Studenti: Nikola Jelić, Luka Lončar, Gabrijela Pirija

Sadržaj

1. UVOD 1
1.1. Tekst zadatka 1
1.2. Opis rješenja 1
2. IMPLEMENTACIJA APLIKACIJE 3
2.1. Izgled 3
2.2. Filtriranje slika 5
2.3. Generiranje puzzle i pomicanje dijelova 6
2.4. Provjera rješenja 9
3. ZAKLJUČAK 10

1. UVOD

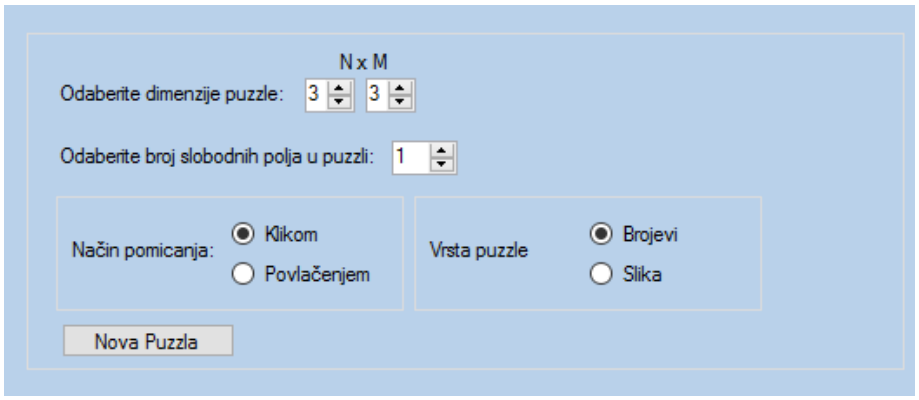
Ova dokumentacija opisuje rješenje projektnog zadatka Puzzle iz kolegija Računarski praktikum 3. Kod je pisan u programskom jeziku C# te smo koristili Visual Studio za izradu projekta.

1.1. Tekst zadatka

Napravite aplikaciju u kojoj se sastavlja puzzla, s tim da se može birati puzzla s različitim brojem dijelova, n × n ili n × m. Također, može se i birati broj slobodnih polja u puzzli. Cilj je napraviti sliku u kojoj su u svakom redu dijelovi iste boje ili su dijelovi označeni brojevima pa ih treba sastaviti po redu. Može se sastavljati i neka konkretna slika. Omogućite igraču da sam određuje postavke igre. Dijelovi u puzzli mogu se pomicati klikom na susjedno polje ili povlačenjem (dozvolite odabir te opcije). Sa strane postoji slika neke stvari, osobe i sl. koja je više puta zamućena nekim filterom (dozvolite odabir teme koja će se pogađati te napravite nekoliko različitih filtera koji će se izmjenjivati). Svaki put kad se sastavi puzzla, slika se vraća prema normalnoj za jedan stupanj. Igrač može u tom trenutku pogađati. Ako ne pogodi, mora sastavljati novu puzzlu. Ukoliko nije u stanju sastaviti puzzlu, može odustati, dobije novu puzzlu i slika se zamuti za jedan stupanj.

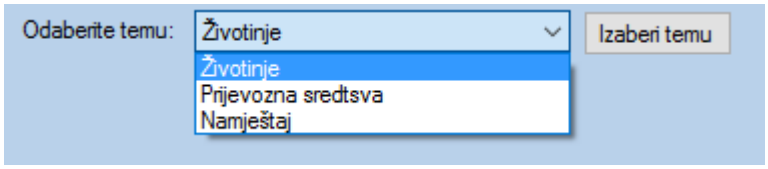
1.2. Opis rješenja

Izabrali smo slijedeće postavke: Igrač može birati dimenzije puzzle koju želi slagati, gdje dozvoljavamo i pravokutnu puzzlu, ne nužno kvadratnu, kako je i traženo u zadatku. Osim dimenzija, igrač bira i broj slobodnih polja s time da ukoliko je igrač odabrao način pomicanja klikom slobodno polje može biti samo jedno. Nema smisla da postoji više slobodnih polja u tom slučaju jer bismo tada morali pitati igrača na koje točno slobodno mjesto želi pomaknuti dio puzzle koji je kliknuo, što bespotrebno komplicira i usporava igru. Ukoliko je izabrao način pomicanja povlačenjem tada može imati više slobodnih mjesta jer točno znamo mjesto na koje želi pomaknuti dio puzzle. Izabrali smo opciju slaganja brojeva po redu, a ne boja i dodali smo opciju slaganja slike. Razlog za odabir brojeva umjesto boja je taj što za brojeve igrač odmah zna kako ih treba poredati. Kada bi morao slagati boje onda ne bi unaprijed znao s kojom bojom mora početi, već bismo mu morali dati predložak kako puzzla treba izgledati, što je onda zapravo kao da slaže sliku, a tu smo opciju ionako implementirali.



Slika 1: Odabir postavki igre

Igrač na samom početku igre bira jednu od ponuđenih tema, te mu program, uz mogućnost biranja postavki igre, nasumično zadaje sliku koju mora pogađati. Slika je već na samom početku obrađena filterima koje smo preuzeli iz biblioteke ImageProcessor (https://imageprocessor.org/). Svaki put kada igrač točno složi puzzlu, jedan filter sa slike se ukloni te igrač ima priliku za pogoditi rješenje. Ukoliko odgovori točno igrač je pobijedio, ako pogriješi mora slagati novu puzzlu, a slika koju pogađa ostaje ista. Igrač ima mogućnost i odustati od puzzle koju slaže, u tom mu se slučaju na sliku dodaje još jedan filter te dobiva novu puzzlu. U svakom trenu igrač može kliknuti na gumb „Nova igra“ te ispočetka birati temu.

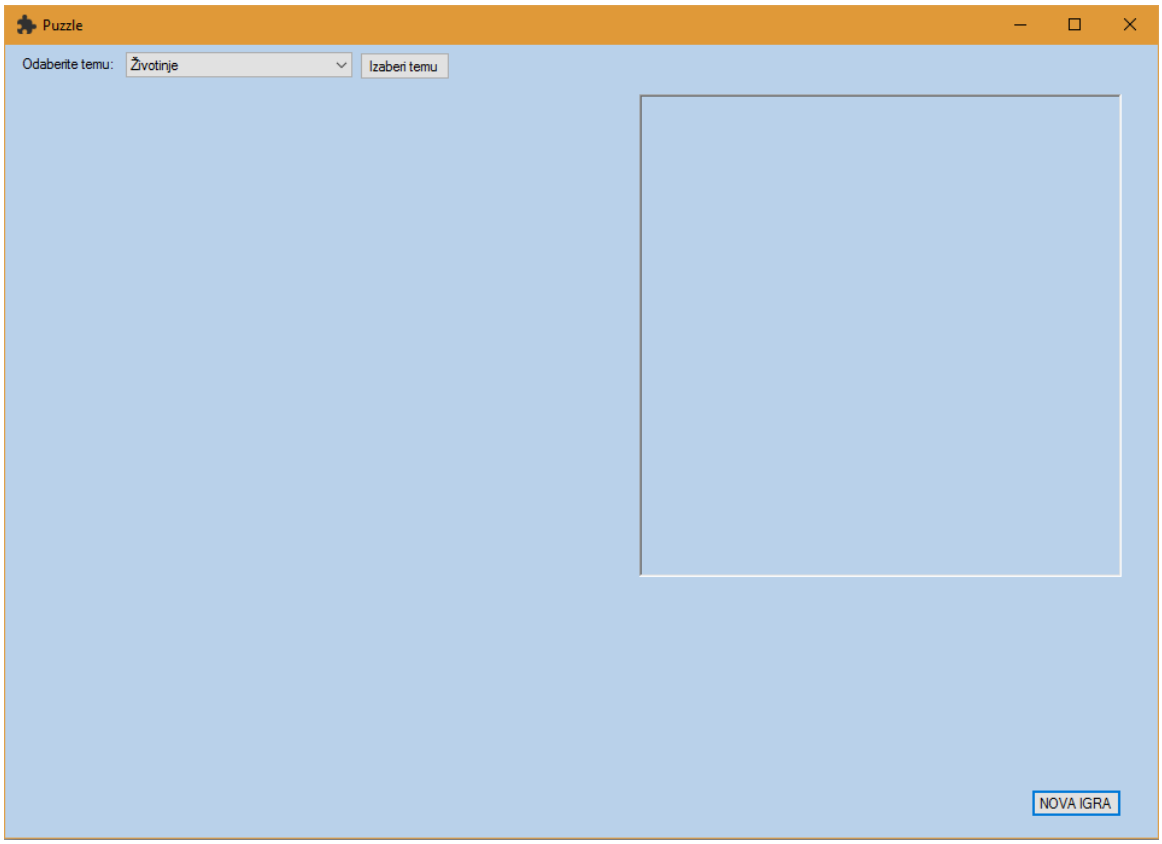


Slika 2: Odabir teme

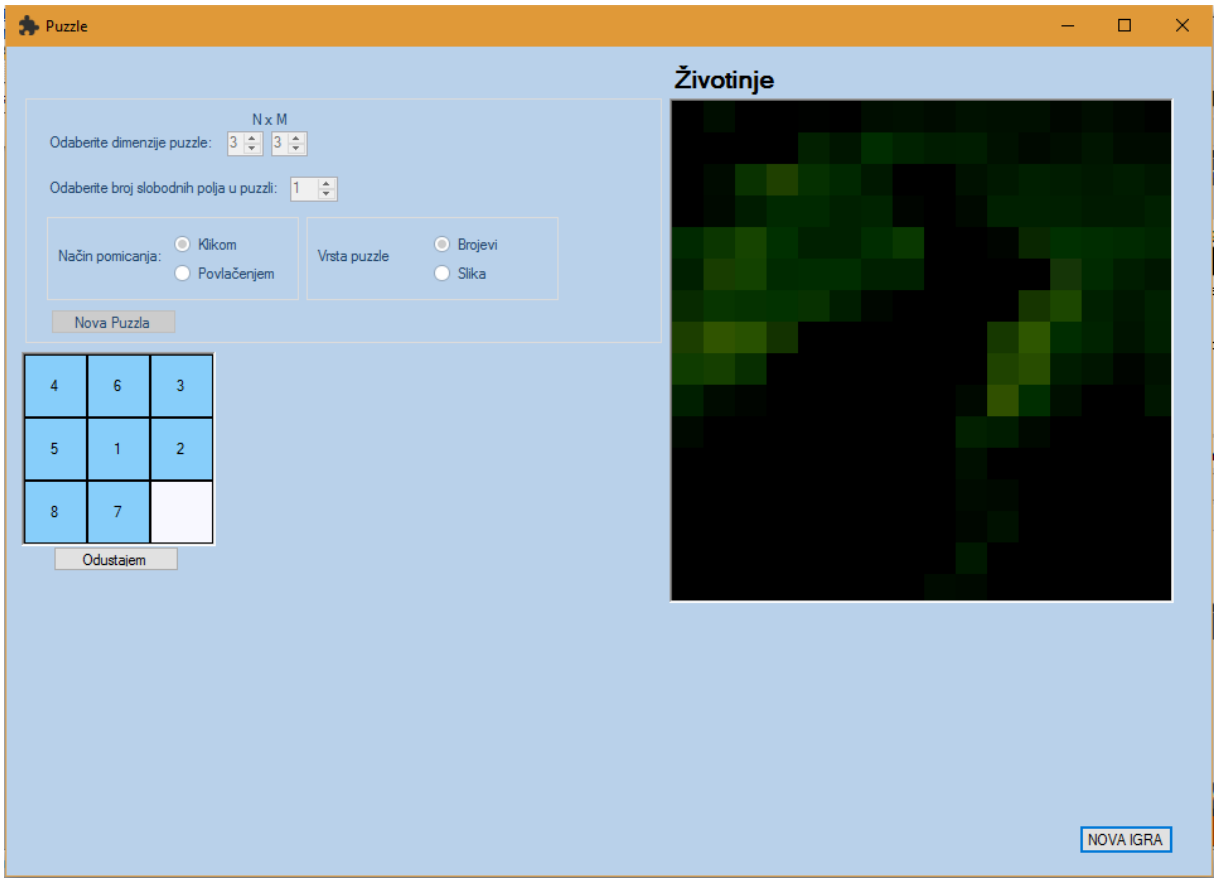
2. IMPLEMENTACIJA APLIKACIJE

2.1. Izgled

Aplikacija se prikazuje u obliku Windows forme koju smo nazvali Pocetna. U početku igrač u prozoru aplikacije vidi samo mogućnost odabira teme kao što vidimo na slici 3. Igraču nudimo odabir jedne od sveukupno tri teme, a svaka tema sadrži po četiri slike. Nakon odabira teme ta se mogućnost uklanja sve dok traje igra. Igra završava pobjedom igrača, klikom na gumb „Nova igra“ ili zatvaranjem aplikacije. Zatim se pokazuje ostatak postavki koje igrač bira, nasumično odabrana obrađena slika koju će igrač pogađati i naslov izabrane teme iznad te slike. Na slici 4 vidi se kako izgleda prozor aplikacije nakon što igrač odabere temu i postavke te klikne na „Nova Puzzla“. Pojavljuje se gumb „Odustajem“ na fiksnom mjestu točno ispod puzzle. Dok slaže puzzlu onemogućeno mu je mijenjanje svih postavki. To smo implementirali tako da smo smjestili sve postavke u jedan GroupBox i onemogućili ga tijekom slaganja puzzle (slika 5).



Slika 3: Prvi zaslon



Slika 4:Zaslon za vrijeme slaganja puzzle

```
//pritiskom na button nova puzzla poziva se ova metoda koja slaže novu igru tako da nasumično rasporedi
//brojeve buttona
private void btnNovaPuzzla_Click(object sender, EventArgs e)
{
    grbSve.Enabled = false;

    n = (int)numN.Value;
    m = (int)numM.Value;
    prazno = (int)numPrazno.Value;
    int brPolja = n * m - prazno;

    flpPuzzle.Width = n * 50 + n + 1;
    flpPuzzle.Height = m * 50 + m + 1;

    btnOdustajem.Visible = true;
    btnOdustajem.Location = new Point(flpPuzzle.Width / 2 - 40, flpPuzzle.Location.Y + flpPuzzle.Height);
```

Slika 5: Onemogućen GroupBox s postavkama

2.2. Filtriranje slika

S interneta smo preuzeli besplatnu biblioteku ImageProcessor koja nam omogućuje filtriranje slika. Odabrali smo tri različita filtra: GaussianBlur, Pixelate i Brightness. Igra kreće tako da je slika obrađena 6 puta, sa svakim odabranim filterom po dva puta. Filtere na sliku primjenjujemo redom: GaussianBlur, Pixelate, Brightness, GaussianBlur, Pixelate, Brightness, GaussianBlur itd., ovisno o broju filtera. Također mijenjamo veličinu slike kako bi odgovarala veličini panela u kojem ju prikazujemo. Navedeno se vidi na slici 6. Svaki put kada igrač točno složi puzzlu slika se obrađuje s jednim filterom manje čime smo simulirali otklanjanje filtera sa slike. Primjer razlike možemo vidjeti na slici 7.

```
public void imgFactoryFiltri()
{
    imgf.Load(resizeImage((Bitmap)Properties.Resources.ResourceManager.GetObject(slika),new Size(400,400)));
    for (int i = 0; i < filtri; ++i){
        if (i % 3 == 0)
            imgf.GaussianBlur(25);
        else if (i % 3 == 1)
            imgf.Pixelate(25);
        else
            imgf.Brightness(-25);
    }
    pnlSlika.BackgroundImage = imgf.Image;
}
```

Slika 6: Obrađivanje slike

Slika 7: Smanjenje filtera

Funkciju imgFactoryFiltri() koju vidimo na slici 6 zovemo u nekoliko navrata. Prvo u trenutku kada se na početku igre odabere tema jer u tom trenu aplikacija nasumično bira sliku iz zadane teme i obrađuje ju. Zatim u funkciji rjesenje() koja nakon svakog poteza igrača ispituje jesu li dijelovi puzzle točno složeni. Ukoliko jesu broj filtera se smanjuje za jedan i zove se funkcija imgFactoryFiltri(). Treća situacija u kojoj ju zovemo je ona kada igrač klikne na „Odustani“. U toj funkciji povećavamo broj filtera za jedan i zovemo funkciju koja vrši obradu slika. Zadnja situacija u kojoj ju zovemo je kada igrač pogodi rješenje pa smanjimo broj filtera na 0 i pozovemo funkciju kako bi igrač vidio „čistu“,

neobrađenu sliku koju je pogađao. Moramo napomenuti da ta funkcija znatno usporava aplikaciju jer svaka obrada slike zahtijeva čekanje od nekoliko sekundi.

2.3. Generiranje puzzle i pomicanje dijelova

Kada igrač klikne na gumb „Nova puzzla“ prvo čitamo vrijednosti n, m i prazno koje je igrač izabrao. Varijabla prazno označava broj praznih polja koju je igrač sam izabrao. Postavljamo veličinu same puzzle, pozicije gumbi u puzzli te gumba „Odustajem“, te ovisno o odabranom načinu micanja dijelova puzzle određujemo koje ćemo funkcije zvati: swapLabel ili btnMouseDown (slika 8).

```
private void btnNovaPuzzla_Click(object sender, EventArgs e)
{
    grbSve.Enabled = false;

    n = (int)numN.Value;
    m = (int)numM.Value;
    prazno = (int)numPrazno.Value;
    int brPolja = n * m - prazno;

    flpPuzzle.Width = n * 50 + n + 1;
    flpPuzzle.Height = m * 50 + m + 1;

    btnOdustajem.Visible = true;
    btnOdustajem.Location = new Point(flpPuzzle.Width / 2 - 40, flpPuzzle.Location.Y + flpPuzzle.Height);

    Random rand = new Random();

    for (int i = 0; i < n * m; i++)
    {
        PuzzleButton pbtn = new PuzzleButton();
        pbtn.pos = i + 1;
        if (nacin)
            pbtn.Click += new EventHandler(swapLabel);
        else
        {
            pbtn.MouseDown += new MouseEventHandler(btnMouseDown);
        }
        flpPuzzle.Controls.Add(pbtn);
    }
```

Slika 8: Postavljanje početnih postavki

Ukoliko igrač odabere slaganje brojeva tada koristimo listu int-ova listaBrojeva pomoću koje svakom dijelu puzzle dodjeljujemo tekst s odgovarajućim brojem i njegovu vrijednost pomoću koje provjeravamo je li slagalica ispravno složena, što ćemo vidjeti kasnije. Lista sadrži n*m-prazno elemenata jer su nam sva prazna polja bez teksta i njih ne mičemo (slika 9).

```
List<int> listaBrojeva = new List<int>();
for (int i = 1; i <= brPolja; i++)
{
    listaBrojeva.Add(i);
}
//provjeravamo je li vrsta puzzle brojevi ili slika, ako su brojevi dodijeli nasumicno brojeve
//ako se želi implementirati slika, treba sliku razrezati na n*m dijelova te nasumicno te dijelove podijeliti
//pazeći da postavljamo za odgovarajući dio odgovarajući value svakom buttonu
if (vrsta)
{
    foreach (PuzzleButton btn in flpPuzzle.Controls)
    {

        if (listaBrojeva.Count != 0)
        {
            int r = rand.Next(listaBrojeva.Count);
            btn.Text = listaBrojeva[r] + "";
            btn.value = listaBrojeva[r];
            btn.BackColor = Color.LightSkyBlue;
            listaBrojeva.RemoveAt(r);
        }
        else
        {
            btn.Text = "";
            btn.value = 0;
            btn.BackColor = Color.GhostWhite;
        }
    }
}
```

Slika 9: Puzzla s brojevima

Ukoliko je igrač izabrao slaganje slike tada prvo tu sliku moramo podijeliti na n*m komadića što radimo u kodu na slici 10. Prvo joj promijenimo veličinu da ju možemo smjestiti u okvire puzzle. Zatim u polje smještamo odgovarajuće manje dijelove slike redom.

```
Bitmap img = Properties.Resources.slagalica;
Bitmap newimg = resizeImage(img, new Size(n * 50, m * 50));
var Pixel_format = newimg.PixelFormat;
var imgarray = new Image[n * m];
//int k = 0;
for (int i = 0; i < m; i++)
{
    for (int j = 0; j < n; j++)
    {
        var index = i * n + j;
        RectangleF Tile = new RectangleF();
        Tile.Width = 50;
        Tile.Height = 50;
        Tile.X = j * 50;
        Tile.Y = i * 50;
        imgarray[index] = newimg.Clone(Tile, Pixel_format);
    }
}
```

Slika 10: Dijeljenje slike na dijelove

Korištenjem iste liste brojeva kao i gore, odabiremo koji dio slike ćemo dati svakom dijelu puzzle redom i postavljamo im odgovarajuće vrijednosti (slika 11).

```
for (int i = 0; i < (n * m - prazno); i++)
{
    listaBrojeva[i] = i;
}

foreach (PuzzleButton btn in flpPuzzle.Controls)
{
    if (listaBrojeva.Count != 0)
    {
        int r = rand.Next(listaBrojeva.Count);
        btn.Image = imgarray[listaBrojeva[r]];
        btn.value = listaBrojeva[r] + 1;
        listaBrojeva.RemoveAt(r);
    }
    else
    {
        btn.Image = null;
        btn.value = 0;
    }
}
```

Slika 11: Puzzla sa slikom

Ukoliko je igrač odabrao način pomicanja povlačenjem miša tada zamjenu dopuštamo ako je dio puzzle na koji ispustimo miš gumb s praznim tekstom ili gumb bez slike. Tada vršimo zamjenu teksta i vrijednosti, odnosno slike i vrijednosti kao što je prikazano u kodu na slici 12.

```
private void flpPuzzle_DragDrop(object sender, DragEventArgs e)
{
    Point p = this.flpPuzzle.PointToClient(new Point(e.X, e.Y));

    PuzzleButton btn = (PuzzleButton)flpPuzzle.GetChildAtPoint(p);
    if (btn != null)
    {
        if (vrsta)
        {
            if (btn.Text == "")
            {
                btn.BackColor = Color.LightSkyBlue;
                moveBtn.BackColor = Color.GhostWhite;
                btn.Text = moveBtn.Text;
                btn.value = int.Parse(moveBtn.Text);
                moveBtn.value = 0;
                moveBtn.Text = "";
                rjesenje();
                flpPuzzle.Focus();
            }
        }

        else if (btn.Image == null)
        {
            btn.Image = moveBtn.Image;
            moveBtn.Image = null;
            btn.value = moveBtn.value;
            moveBtn.value = 0;
            rjesenje();
            flpPuzzle.Focus();
        }
    }
```

Slika 12: Pomicanje povlačenjem

Ukoliko je odabrana opcija pomicanja klikom na gumb odnosno dio puzzle prvo provjeravamo je li taj gumb bez teksta odnosno bez slike. Ako je, ne radimo nikakvu promjenu. Ako nije onda nalazimo prazni gumb u cijeloj puzzli te dopuštamo zamjenu ako se naš odabrani gumb nalazi lijevo ili desno od praznoga ili ispod ili iznad praznoga (slika 13).

```
PuzzleButton prazniBtn = null;
if (vrsta)
{
    foreach (PuzzleButton bt in flpPuzzle.Controls)
    {
        if (bt.Text == "")
        {
            prazniBtn = bt;
            break;
        }
    }

    if ((btn.pos == (prazniBtn.pos - 1) && prazniBtn.pos % n != 1) || btn.pos == (prazniBtn.pos + n) ||
        btn.pos == (prazniBtn.pos - n) || (btn.pos == (prazniBtn.pos + 1) && prazniBtn.pos % n != 0))
    {
        prazniBtn.BackColor = Color.LightSkyBlue;
        btn.BackColor = Color.GhostWhite;
        prazniBtn.value = btn.value;
        prazniBtn.Text = btn.Text;
        btn.Text = "";
        btn.value = 0;
        flpPuzzle.Focus();
    }
    rjesenje();
}
```

Slika 13: Pomicanje klikom

2.4. Provjera rješenja

Provjera je li puzzla složena točno vrši se tako da za svaki gumb odnosno dio puzzle provjeravamo njegov tekst i vrijednost tj. sliku i vrijednost. Prvi gumb mora imati zapisan tekst odnosno sliku te mora imati vrijednost 1, drugi gumb mora imati zapisan tekst odnosno sliku te mora imati vrijednost 2, itd. Slike odnosno tekstovi će ići dobrim redoslijedom jer smo ih zadavali zajedno s vrijednostima redom i kada pomičemo gumbe vršimo zamjenu teksta/slike i vrijednosti. Ukoliko nađemo gumb koji ne zadovoljava tražene uvjete funkcija ne radi ništa, a ako svi gumbi zadovoljavaju uvjete broj filtera na slici se smanjuje, vrši se obrada slike i nudi se mogućnost pogađanja rješenja. Navedeno vidimo na slici 14.

```
int i = 1;
foreach (PuzzleButton btn in flpPuzzle.Controls)
{
    if (btn.Text != "" && btn.value != i)
    {
        return;
    }
    i++;
}
filtri--;
imgFactoryFiltri();
txtPogadaj.Visible = true;
lblPogadaj.Visible = true;
btnPogodi.Visible = true;
btnOdustajem.Visible = false;
txtPogadaj.Focus();
```

```
int i = 1;
foreach (PuzzleButton btn in flpPuzzle.Controls)
{
    if (btn.Image != null && btn.value != i)
    {
        return;
    }
    i++;
}
filtri--;
imgFactoryFiltri();
txtPogadaj.Visible = true;
lblPogadaj.Visible = true;
btnPogodi.Visible = true;
txtPogadaj.Focus();
```

Slika 14: Provjera rješenja

3. ZAKLJUČAK

Izrada projekta trajala je 4 dana gdje smo određene dijelove radili zasebno, svatko sam, a određene dijelove smo radili zajedno. Mogućnost poboljšanja vidimo u ubrzanju rada aplikacije korištenjem neke druge biblioteke za filtriranje slika ili izrada vlasititih funkcija koje obrađuju sliku.