УНИВЕРЗИТЕТ У НОВОМ САДУ

ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Игор Кунчак, PR1/2018

2D Визуализација шеме електродистрибутивне мреже са фокусом на оптимизацију BFS алгоритма

ПРОЈЕКАТ

- Примењено софтверско инжењерство (ОАС) -

Нови Сад, 25.06.2023.

Sadržaj

OPIS REŠAVANOG PROBLEMA

Tema projekta je 2D vizualizacija šeme elektrodistributivne mreže sa fokusom na optimizaciju BFS algoritma[1]. Elektrodistributivna mreža je kompleksna mreža povezanih trafostanica, čvorova, prekidača. Cilj projekta je unaprediti Predmetni Zadatak 1 izgradnjom grafa elektrodistributivne mreže, optimizacijom i prikazivanjem rezultata BFS algoritma.

Osnovni zadatak je vizualizacija elektrodistributivne mreže. Zadatak podrazumeva kreiranje grafičkog prikaza mreže na osnovu geografskih podataka koji su dostavljeni u XML formatu[2]. Kako bismo aproksimirali grafički prikaz mreže, koristićemo ortogonalnu reprezentaciju koja deli prostor za crtanje na "imaginarnim" ćelijama. Što je broj ćelija za prikaz veći(dimenzije matrice veće) to je i prikaz preciziji.

Koordinate entiteta mreže će biti učitane iz XML fajla i aproksimirane do najbliže ćelije na prostoru za crtanje pre nego što budu iscrtane. Svaki entitet (trafostanica, čvor, prekidač) će biti predstavljen grafičkim elementom (kvadratom) obojenim specifično za svoj tip. Za svaki grafički element će biti prikazan iskačući prozor (tooltip) sa detaljima o entitetu koji se nalazi na tom mestu.

Vodovi koji povezuju entitete će biti prikazani kao ravne linije, pri čemu će se promena pravca vršiti samo pod pravim uglom. U obzir će biti uzeti samo početni i krajnji čvorovi vodova, bez razmatranja međutemena(vertices). Biće iscrtani samo vodovi koje povezuju entitete iz kolekcija entiteta, bez prikaza drugih entiteta. Takođe, biće izbegnuto ponovno iscrtavanje vodova između istih entiteta.

Optimizacija BFS algoritma je glavni cilj ovog projekta u odnosu na Predmetni Zadatak 1. Algoritam će prvo odrediti najkraću putanju koja izbegava presecanje već iscrtanih vodova. U drugom prolazu će biti iscrtani vodovi, obeležavajući tačke preseka za one slučajeve gde nije bilo moguće pronaći ne-presečnu putanju u prvom prolazu.

Pored optimizacije BFS algoritma i izrade prikaza elektrodistributivne mreže, ovaj projekat obuhvata dodatne funkcionalnosti i opcije za korisnika. Na vrhu prozora će biti prikazane opcije koje korisnici mogu odabrati, promenu veličine platna kao i dopuna prikazane mreže sa oblicima i/ili tekstom.

OPIS KORIŠĆENIH TEHNOLOGIJA I ALATA

Visual Studio:

- Razlog izbora: Visual Studio[3] je jedan od najpopularnijih integrisanih razvojnih okruženja (IDE) za programiranje u C# jeziku[4].
- Svojstva i prednosti:
  - Napredno razvojno okruženje sa bogatim skupom alata i funkcionalnosti.
  - Pruža olakšano kreiranje, testiranje i održavanje aplikacija.
  - Podržava integraciju sa raznim platformama i tehnologijama.

C#:

- Razlog izbora: C# je moderni objektno-orijentisani programski jezik koji pruža snagu i fleksibilnost za razvoj desktop aplikacija.
- Svojstva i prednosti:
  - Jezik sa čistom sintaksom i jednostavnim učenjem.
  - Pruža podršku za objektno-orijentisano programiranje, nasleđivanje, polimorfizam i druge koncepte.
  - Integracija sa .NET Framework-om omogućava pristup bogatim bibliotekama i funkcionalnostima.

Windows Presentation Foundation (WPF):

- Razlog izbora: WPF[5] je moderni okvir za razvoj grafičkih korisničkih interfejsa (GUI) u aplikacijama za Windows platformu.
- Svojstva i prednosti:
  - Pruža bogat set kontrola i stilova za kreiranje atraktivnih GUI-ja.
  - Podržava razne mogućnosti vezivanja podataka i animacija.
  - Omogućava fleksibilan raspored elemenata i skalabilnost na različitim ekranima.

OPIS REŠENJA PROBLEMA

Rešenje problema se sastoji iz više delova koji se izvršavaju kako bi se postigao konačan rezultat, odnosno željeni prikaz elemenata na Canvasu. Na narednim stranicama se nalazi implementacija kao i njeno objašnjenje.

Segment aplikacije koji se odnosi na rad sa podacima o elektroenergetskoj mreži, vodovima, entitetima i vodovima implementira klasa GridData.

```
public class GridData:IGridData
{
    private static readonly Lazy<GridData> instance = new Lazy<GridData>(
        () => new GridData());
    private readonly int size = 500;
    public ConcurrentDictionary<long, NewEntity> entities;
    public ConcurrentDictionary<long, List<Polyline>> lines;
    public ConcurrentDictionary<long,Vod> vodoviClick;
    public List<long>[,] linesIds;
    public bool[,] positions;
    public bool[,] linesObjects;
    2 references | qnchuck, 21 days ago | 1 author, 1 change
    public static GridData Instance => instance.Value;

    1 reference | qnchuck, 21 days ago | 1 author, 1 change
    private GridData()
    {
        vodoviClick = new ConcurrentDictionary<long, Vod>();
        entities = new ConcurrentDictionary<long,NewEntity>();
        linesIds = new List<long>[size, size];
        lines = new ConcurrentDictionary<long, List<Polyline>>();
        positions = new bool[size, size];
        linesObjects = new bool[size, size];
        for (int i = 0; i < size; i++)
        {
            for (int j = 0; j < size; j++)
            {
                linesIds[i, j] = new List<long>();
            }
        }

    }
}
```

Slika 1 - Klasa GridData

Kreiranje pojedinačnih node, switch i substation entiteta:

Prvi deo rešenja podrazumeva kreiranje pojedinačnih entiteta kao što su čvorovi (node), prekidači (switch) i trafostanice (substation). Za implementaciju kreiranja navedenih entiteta koristimo projektni obrazac "Template method" koji omogućava generičko kreiranje entiteta na osnovu pruženih podataka. Podaci se učitavaju iz zadate "Geographic.xml" datoteke. Ovi entiteti se vizuelno predstavljaju pravougaonicima i dodaju se u odgovarajuću listu entiteta.

Klasa "EntityCreation" je apstraktna klasa koja definiše osnovnu strukturu za kreiranje entiteta. Ona sadrži nekoliko zaštićenih podataka kao što su "powerEntity" koji predstavlja entitet, "rectangle" koji je pravougaonik koji se koristi za vizuelni prikaz entiteta, "gridDataEntities" koji predstavlja podatke o mreži entiteta.

```
public abstract class EntityCreation
{
    protected PowerEntity powerEntity = new PowerEntity();
    protected Rectangle rectangle;
    protected GridData gridDataEntities = GridData.Instance;
    protected double convertedX, convertedY;
    protected ToolTip toolTip = new ToolTip();

    3 references | qnchuck, 2 days ago | 1 author, 2 changes
    public Rectangle CreateEntityRectangleFromXmlNode(
        XmlNode node,
        Approximation nodeApproximation,
        int size,
        int dimensions
    )
    {
        this.InitPowerEntity(node);
        this.CreateRectangle(dimensions, size);
        this.SetRectangleColor();
        this.AddEntityToEntityList(size,dimensions,nodeApproximation);
        this.CreateRectangleToolTip();
        this.SetToolTipContent();

        return rectangle;
    }
```

Slika 2 - klasa EntityCreation

Metod "CreateEntityRectangleFromXmlNode" definiše korake koji su neophodni za kreiranje pravougaonika entiteta, ali ne pruža konkretne implementacije svih koraka. Umesto toga, ova klasa prepušta potklasama da implementiraju konkretne detalje određenih koraka, tako da se algoritam može prilagoditi za različite vrste entiteta.

Na primer, metode kao što su "InitPowerEntity", "CreateRectangle", "CreateRectangleToolTip", "AddEntityToEntityList" su metode koje se pozivaju unutar template metoda. Takođe, imamo i apstraktne metode "SetToolTipContent" i "SetRectangleColor". Ove metode su obeležene kao apstraktne jer se konkretne implementacije tih metoda nalaze u potklasama.

"InitPowerEntity" vrši učitavanje podataka iz prosleđenog XML čvora kao i konvertovanje učitanih koordinata u odgovarajuće, koristeći metodu *ToLatLon*.

"CreateRectangle" i "CreateRectangleToolTip" kao što i samo ime govori kreiraju kvadrat i njemu dodeljeni tooltip.

```
protected void InitPowerEntity(XmlNode node)
{
    powerEntity.Id = long.Parse(node.SelectSingleNode("Id").InnerText);
    powerEntity.Name = node.SelectSingleNode("Name").InnerText;
    powerEntity.X = double.Parse(node.SelectSingleNode("X").InnerText);
    powerEntity.Y = double.Parse(node.SelectSingleNode("Y").InnerText);

    Approximation.ToLatLon(powerEntity.X, powerEntity.Y, 34, out convertedY, out convertedX);

}
1 reference | qnchuck, 21 days ago | 1 author, 1 change
protected void CreateRectangle(int dimensions, int size)
{
    rectangle = new Rectangle();
    rectangle.Width = 1.0 * dimensions / size;
    rectangle.Height = 1.0 * dimensions / size;
}
1 reference | qnchuck, 2 days ago | 1 author, 2 changes
protected void CreateRectangleToolTip()
{
    toolTip = new ToolTip();
    toolTip.Background = Brushes.Black;
    toolTip.Foreground = Brushes.White;
    toolTip.BorderBrush = Brushes.Black;
    rectangle.ToolTip = toolTip;
}
1 reference | qnchuck, 21 days ago | 1 author, 1 change
protected void AddEntityToEntityList(int size, int dimensions, Approximation nodeApproximation)
{
    double newX = Approximation.GetX(size, convertedX, nodeApproximation.YMinimum, nodeApproximation.YMaximum),
           newY = Approximation.GetY(size, convertedY, nodeApproximation.XMinimum, nodeApproximation.XMaximum);

    Coordinate coordinate = Approximation.FindPosition(
        gridDataEntities.positions, newX, newY
        );
    gridDataEntities.positions[coordinate.X, coordinate.Y] = true;

    double x = Approximation.GetCanvasX(dimensions, coordinate.X, size);
    double y = Approximation.GetCanvasY(dimensions, coordinate.Y, size);
    rectangle.Margin = new Thickness(x, y, 0, 0);
    NewEntity newEntity = new NewEntity(powerEntity.Id, coordinate.X, coordinate.Y);

    gridDataEntities.entities.TryAdd(newEntity.Id, newEntity);
}
```

Slika 3 Konkretne metode iz apstraktne klase EntityCreation

"AddEntityToEntityList" pomoću metoda *GetX* i *GetY* izračunava relativnu poziciju elementa na osnovu minimalne i maksimalne vrednosti koordinata x i y svih učitanih čvorova. Nakon toga, koristeći metodu *FindPosition* određuje poziciju unutar matrice bool-ova *gridDataEntities.positions*.

Pozicija se određuje tako što se od relativne pozicije uzima celobrojna vrednost i pokušava pozicioniranje na dobijene koordinate, ukoliko to nije moguće, pozicija se traži u ćelijama koje okružuju dobijenu. U slučaju da u prvom „krugu“ oko ćelije ne bude mesta, proveravaju se one koje okružuju taj „krug“ i tako dok se ne nađe slobodno mesto. Na osnovu dobijenih vrednosti koordinata se pronalazi odgovarajuća pozicija pomoću metoda *GetCanvasX* i *GetCanvasY* unutar Canvasa.

Sledeći deo rešenja uključuje učitavanje i sortiranje vod entiteta. Vodovi predstavljaju veze između entiteta i mogu biti grafički prikazani kao linije. Ovi entiteti se čitaju iz odgovarajućeg izvora (XML datoteka) i sortiraju se kako bi se obezbedio pravilan redosled prilikom vizuelnog prikaza.

U nastavku su navedene konkretne klase za kreiranje node, switch i substation entiteta. Svakom od entiteta se dodeljuje vrednost koja će biti prikazana u tooltip-u kao i boja samog grafičkog elementa.

```
public class ConcreteNodeCreation:EntityCreation
{
    2 references | qnchuck, 21 days ago | 1 author, 1 change
    protected override void SetToolTipContent()
    {
        toolTip.Content = "Node \n ID: " + powerEntity.Id + "  Name: " + powerEntity.Name;
    }

    2 references | qnchuck, 21 days ago | 1 author, 1 change
    protected override void SetRectangleColor()
    {
        rectangle.Fill = Brushes.Blue;
    }
}
```

Slika 4 - klasa ConcreteNodeCreation

```
public class ConcreteSwitchCreation : EntityCreation
{
    2 references | qnchuck, 21 days ago | 1 author, 1 change
    protected override void SetToolTipContent()
    {
        toolTip.Content = "Switch \n ID: " + powerEntity.Id + "  Name: " + powerEntity.Name;
    }

    2 references | qnchuck, 21 days ago | 1 author, 1 change
    protected override void SetRectangleColor()
    {
        rectangle.Fill = Brushes.Green;
    }
}
```

Slika 5 - Klasa ConcreteSwitchCreation

```
public class ConcreteSubstationCreation : EntityCreation
{
    2 references | qnchuck, 21 days ago | 1 author, 1 change
    protected override void SetToolTipContent()
    {
        toolTip.Content = "Substation \n ID: " + powerEntity.Id + "  Name: " + powerEntity.Name;
    }

    2 references | qnchuck, 21 days ago | 1 author, 1 change
    protected override void SetRectangleColor()
    {
        rectangle.Fill = Brushes.Brown;
    }
}
```

Slika 6 - Klasa ConcreteSubstationCreation

Na slici ispod prikazano je iteriranje kroz listu učitanih Switch čvorova gde se, koristeći gore pomenutu "CreateEntityRectangleFromXmlNode" metodu, kreiraju grafički elementi nakon čega se dodaju na Canvas. Analogno tome se dodaju i Substation i Node entiteti u Canvas.

```
SwitchEntity switchobj = new SwitchEntity();

nodeList = xmlDoc.DocumentElement.SelectNodes("/NetworkModel/Switches/SwitchEntity");

Approximation switchApproximation = Approximation.GetXYMinMax(nodeList);

ConcreteSwitchCreation concreteSwitch = new ConcreteSwitchCreation();

foreach (XmlNode node in nodeList)
{
    Rectangle rectangle = concreteSwitch.CreateEntityRectangleFromXmlNode(
        node,nodeApproximation,size,dimensions
        );
    rectangle.Uid = node.SelectSingleNode("Id").InnerText;
    elementi.Children.Add(rectangle);
}
```

Slika 7 - Dodavanje elemenata u Canvas

Učitavanje vodova iz zadate XML datoteke

Učitavanje se vrši tako što se iz zadate XML datoteke učitaju čvorovi koji predstavljaju vodove. Nakon toga se foreach petljom prolazi kroz njihovu listu i iz svakog čvora se izvlače potrebni podaci kao što su Id, FirstEnd, SecondEnd, R, ConductorMaterial, LineType, ThermalConstant i Name.

Proverom da li je u listu učitanih entiteta već dodat vod koji spaja FirstEnd i SecondEnd entitete će kasnije biti izbegnuto ponovno iscrtavanje vodova između istog para entiteta.

BFS Algoritam

BFS algoritam se koristi za sistematsko pretraživanje ili pronalaženje najkraćeg puta između dve tačke u mreži. Može se zamisliti da svaki element dvodimenzionalnog niza predstavlja čvor u grafu, pri čemu su susedni elementi povezani granama.

BFS algoritam se primenjuje tako što se počinje od početne tačke i obilazi se svaki susedni element u mreži. Svaki sused koji nije posećen se označava kao posećen i dodaje se u red za čekanje. Zatim se iz reda uzima prvi element i obilaze se njegovi neposećeni susedi. Ovaj proces se ponavlja dok god ima elemenata u redu.

```
2 references | qnchuck, 2 days ago | 1 author, 4 changes
public static Node DoBFS(bool[,] mat, Coordinate src,
                         Coordinate dst, ref Node destination, int size)
{
    bool[,] visited = new bool[size,size];
    visited[src.X, src.Y] = true;

    Queue<Node> nq = new Queue<Node>();
    Node node = new Models.Entities.Node(src.X, src.Y) { Parent = null };
    nq.Enqueue(node);
    while (nq.Count != 0)
    {
        Node curr = nq.Peek();

        for (int i = 0; i < 4; i++)
        {
            int row = curr.X + rowNum[i];
            int col = curr.Y + colNum[i];

            if (isValid(visited,row, col,size) && (!mat[row, col] || (row == dst.X && col == dst.Y)) &&
            !visited[row, col])
            {
                visited[row, col] = true;

                Node parent = curr;
                Node nodeTemp = new Node(row, col, ref parent)
                {
                    dst = parent.dst + 1
                };
                if (dst.X == row && dst.Y == col)
                {
                    destination.dst = nodeTemp.dst;
                    destination.Parent = parent;
                    return destination;
                }
                nq.Enqueue(nodeTemp);
            }
        }
        nq.Dequeue();
    }
    return null;
}
```

Slika 8- optimizovani BFS algoritam

Kreiranje vodova bez preseka, koristeći BFS algoritam

Nakon što su učitani entiteti, vrši se prvi prolazak kroz BFS (Breadth-First Search) algoritam. Ovaj algoritam se koristi za pronalaženje najkraćeg puta između dva entiteta, izbegavajući preklapanje već nacrtanih vodova.

Proverom vrednosti bool promenljive u matrici *GridDataEntities.linesObjects* na svakoj od koordinata kojima prolazi BFS algoritam zaključuje se da li je ona zauzeta, odnosno da li je moguće tuda sprovesti vod. Prvi prolazak identifikuje najkraći put između entiteta i obeležava vodove koji će biti prikazani kao crne linije. Svaka linija je u stvari polyline koji se sastoji iz čvorova povratne vrednosti BFS-a.

Optimizacija BFS algoritma

Dodatna informacija o roditeljskom čvoru za svaki posećeni čvor omogućava efikasno praćenje putanje od ciljnog čvora do početnog čvora. Na taj način, nakon što se pronađe ciljni čvor, putanja se može lako rekonstruisati unazad koristeći informaciju o roditeljskom čvoru za svaki čvor u putanji.

Ova optimizacija smanjuje vreme izvršavanja i kompleksnost algoritma, jer eliminiše potrebu za dodatnom obradom ili pretragom putanje nakon završetka pretrage. Umesto toga, putanja se može direktno izgraditi koristeći informaciju o roditeljskim čvorovima, što dovodi do efikasnijeg pronalaženja putanje u dvodimenzionalnom prostoru.

```
public class Node
{
    29 references | qnchuck, 22 days ago | 1 author, 1 change
    public int X { get; set; }
    29 references | qnchuck, 22 days ago | 1 author, 1 change
    public int Y { get; set; }
    18 references | qnchuck, 22 days ago | 1 author, 1 change
    public Node Parent { get; set; }
    public int dst;

    1 reference | qnchuck, 22 days ago | 1 author, 1 change
    public Node(int x, int y, ref Node parent)
    {
        dst = 0;
        Parent = parent;
        X = x;
        Y = y;
    }
    3 references | qnchuck, 22 days ago | 1 author, 1 change
    public Node(int x, int y)
    {
        dst = 0;
        Parent = null;
        X = x;
        Y = y;
    }
}
```

Slika 9 - Klasa Node

Za pamćenje podataka o čvorovima korišćena je klasa Node.

Kreiranje vodova sa presekom, koristeći BFS algoritam

U ovom delu koda, iteracija kroz listu vodova koji nisu mogli biti iscrtani bez preseka podrazumeva ponovno izvršavanje BFS algoritma, ali nad drugačijom matricom. Ova nova matrica, koja se koristi za BFS algoritam, ima, označene kao zauzete, samo ćelije na kojima se nalaze entiteti, dok ostale ćelije nisu označene kao zauzete (tj. tretiraju se kao slobodne).

Ovo se radi kako bi se povezali entiteti koje nije bilo moguće povezati u prvom delu, gde nije bilo moguće preseći drugi vod.

Na osnovu rezultata BFS pretrage, iscratava se vod, kao i tačke preseka i preklapanja vodova, koje se zatim na odgovarajući način označavaju kako bi se obezbedilo ispravno prikazivanje na Canvasu.

U klasi GridData postoji dvodimenzionalna matrica koja za svaku ćeliju ima listu id-eva vodova koji se nalaze na tim koordinatama.

```
public List<long>[,] linesIds;
```

Provere da li je u određenoj tački bio presek zasnivaju se na vrednostima preseka listi na trenutnoj i na prethodnoj poziciji voda, i ukoliko nije detektovan presek na toj poziciji, dodatno se proverava i da li je presek listi prazan na trenutnoj i narednoj poziciji. Ukoliko se u nekoj od te dve situacije detektuje presek, on se adekvatnom elipsom i označava.

```
idslines = GridDataEntities.linesIds[previous.X, previous.Y].
    Intersect(GridDataEntities.linesIds[destination.X, destination.Y]).ToList();
```

Slika 10 - Presek listi vodova na dve uzastopne ćelije

Provera za preklapanja vodova takođe se određuje koristeći *linesIds* strukturu. U situaciji kada se na dve uzastopna čvora, iz povratne vrednosti BFS algoritma, ne nalazi više različitih vodova, linija se iscrtava, u suprotnom ona se ne iscrtava.

Svakoj liniji se dodaje odgovarajući tooltip sa id-em tog voda kao i delegat da se prilikom desnog klika na vod označe entiteti koje taj vod povezuje.

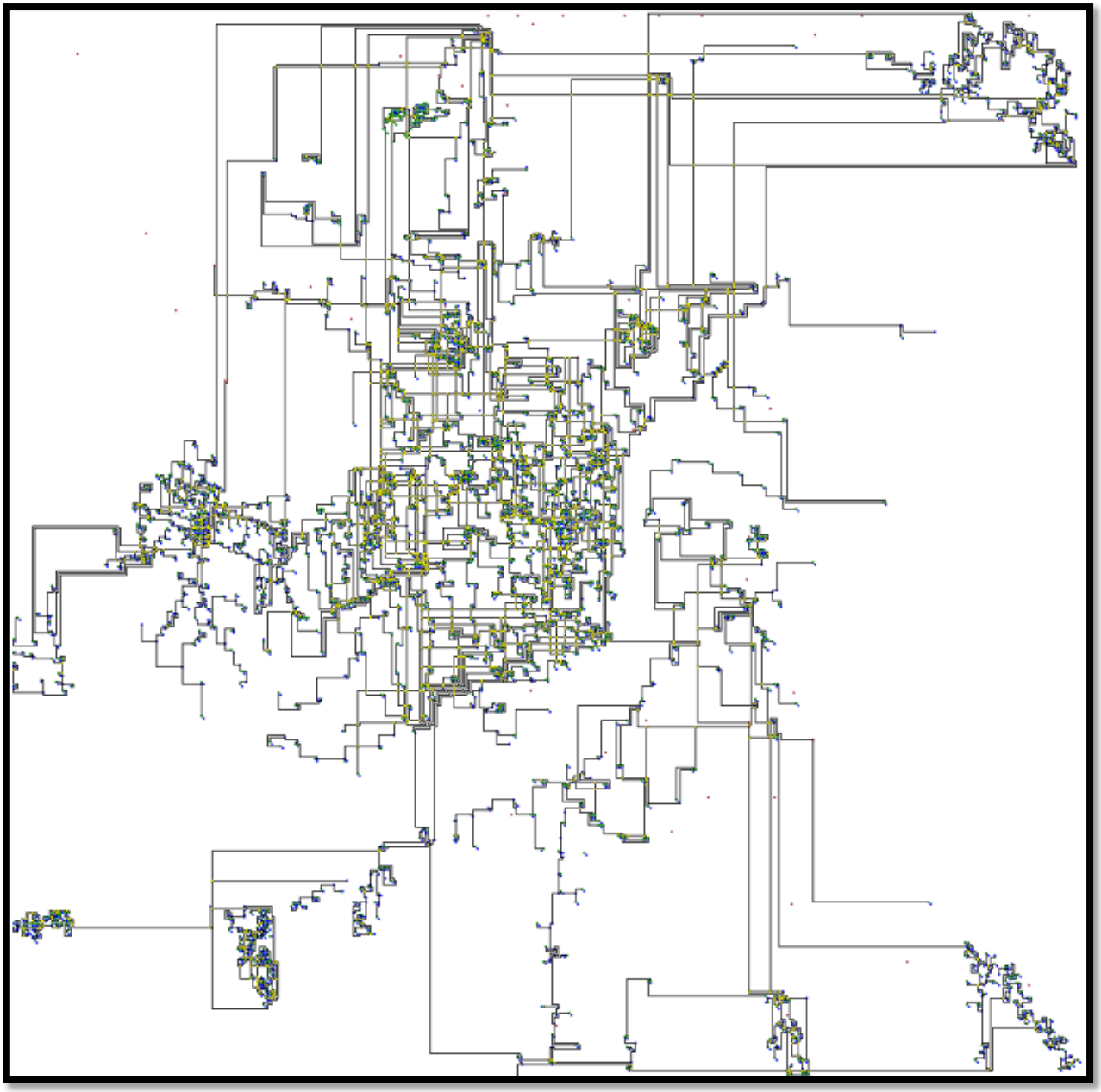
Tabelarni prikaz vremena potrebnih da se mreža iscrta

	Svi entiteti	Bez node entiteta
200x200	4.1605783 s	1.1436534 s
750x750	4.6510254 s	1.2469996 s
1500x1500	5.1399959 s	1.4348533 s

Tabela 1 Vremena iscrtavanja sa i bez node entiteta

Iz analize tabele može se zaključiti da vreme iscrtavanja platna značajno raste sa povećanjem dimenzija platna, što je očekivano jer se povećava broj piksela koji treba obraditi. Takođe, primećuje se da se vreme iscrtavanja značajno smanjuje kada se isključe Node entiteti, što ukazuje na njihovu značajnu ulogu u ukupnom vremenu izvršavanja algoritma. Ovo može biti korisno saznanje prilikom optimizacije algoritma i odabira koje entitete treba uključiti ili isključiti u zavisnosti od specifičnih performansi i zahteva aplikacije.

Izgled Canvasa nakon iscrtavanja svih entiteta



Slika 11 - Izgled Canvasa nakon učitavanja svih entiteta i vodova

Dopuna grafa elektroenergetske mreže oblicima i tekstom

Nakon što korisnik klikom na određeno dugme (DrawPolygon, DrawEllipse, AddText) odabere šta želi da doda na Canvas, on će desnim klikom odabrati tačku/tačke koje će odrediti poziciju na kojoj će se oblik/tekst nalaziti.

U slučaju da je odabrao poligon[6], od korisnika se traži da označi desnim klikom temena tog poligona, nakon čega je potrebno da levim klikom inicira otvaranje prozora za unos atributa poligona kao i opcionog dodavanja teksta na

površinu poligona. Opciono može odabrati i da se boji poligona da efekat providnosti.

```
Color C = cp.SelectedColor.Value;
polygon.Stroke = new SolidColorBrush(C);
if (checkBoxTransparent.IsChecked == true)
{
    byte alpha = 150;
    SolidColorBrush brush = new SolidColorBrush(Color.FromArgb(alpha, C.R, C.G, C.B));
    polygon.Stroke = brush;
}
```

Slika 12 - Davanje boji poligona efekat providnosti

Ukoliko korisnik odabere crtanje elipse[7], sve što se od njega traži jeste da desnim klikom inicira otvaranje prozora za crtanje elipse kojoj će definisati debljinu konturne linije, dužine poluprečnika i boju, kojoj se takođe može dati efekat providnosti.
Opciono se takođe može dodati tekst što se dešava čekiranjem checkBox-a u popunjavanjem zahtevanih atributa teksta.

```
if (checkBoxText.IsChecked == true)
{
    if (cp2.SelectedColor.HasValue)
    {
        textBlock.Foreground = new SolidColorBrush(cp2.SelectedColor.Value);
    }
    else
    {
        textBlock.Foreground = System.Windows.Media.Brushes.Black;
    }

    Canvas.SetLeft(textBlock, xx );
    Canvas.SetTop(textBlock, yy + elipsa.Height / 2 );
```

Slika 13 - dodavanje teksta na površinu elipse

Što se tiče dodavanja samog teksta na površinu Canvasa, sve što će se od korisnika zahtevati jeste da unese sâm tekst kao i boju i veličinu fonta.

Klikom na undo će poslednje dodati oblik/tekst biti uklonjen, dok će klikom na redo biti vraćen. Odabirom opcije clear će svi iscrtani oblici/tekst biti uklonjeni sa Canvasa. Dok će undo nakon clear vratiti sve prethodno uklonjene oblike/tekst.

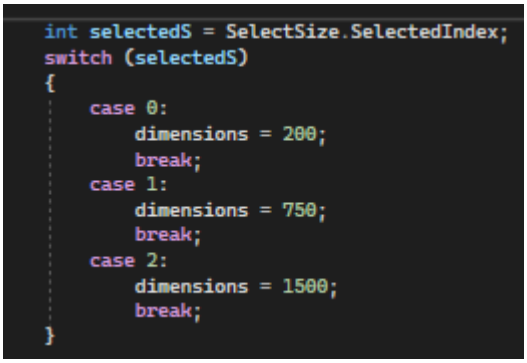
```
private void ClearButton_Click(object sender, RoutedEventArgs e)
{
    buttons.Add(button.clear);
    undoEl = false;
    for (int i = numberOfEntities; i < elementi.Children.Count; i++)
    {
        children.Add(elementi.Children[i]);
    }
    elementi.Children.RemoveRange(numberOfEntities, elementi.Children.Count-numberOfEntities);

}
```

Slika 14 - Opcija clear koja uklanja sve iscrtane oblike/tekst

Kao što se na Slici 14 vidi, prilikom klika na opciju Clear, već postojeći entiteti samog grafa elektrodistributivne mreže će ostati netaknuti, tačnije samo će biti uklonjeni oni elementi koji su dodati nakon njih.

Nakon što korisnik učita graf elektrodistributivne mreže, njega može sačuvati kao sliku, a ime fajla će sadržati vremenski trenutak kada je slika sačuvana. Ovo omogućava korisniku da zadrži trajan zapis svog vizuelnog prikaza i da ga deli ili koristi u kasnijem vremenu.

Korisnik ima mogućnost da promeni dimenzije platna na vrednosti 200x200, 750x750 i 1500x1500. Ova fleksibilnost omogućava prilagođavanje prikaza elektrodistributivne mreže različitim potrebama i zahtevima, omogućavajući prikaz detaljne ili opširne mreže u zavisnosti od dimenzija platna.



```
int selectedS = SelectSize.SelectedIndex;
switch (selectedS)
{
    case 0:
        dimensions = 200;
        break;
    case 1:
        dimensions = 750;
        break;
    case 2:
        dimensions = 1500;
        break;
}
```

Slika 15 - promena dimenzija platna na kom će se iscrtavati graf

PREDLOZI ZA DALJA USAVRŠAVANJA

U ovoj aplikaciji je iskorišćen projektni obrazac Template metoda za olakšavanje kreiranja entiteta elektrodistributivne mreže. Ova prilagodljivost omogućava efikasno kreiranje različitih vrsta entiteta unutar mreže, pružajući fleksibilnost i proširivost aplikacije.

Pored optimizacije BFS algoritma i iskorišćenja projektnog obrasca Template metoda, aplikacija može dalje evoluirati i unaprediti se kroz sledeće predloge:

1. Proširenje na 3D vizualizaciju[8]: Jedan od mogućih pravaca daljeg usavršavanja aplikacije je prelazak sa 2D vizualizacije na 3D vizualizaciju elektrodistributivne mreže. Ovo bi omogućilo korisnicima da dobiju još realističniji prikaz mreže sa dodatnom dubinom i perspektivom. Uz tu promenu, korisnici bi mogli istraživati i analizirati mrežu iz različitih uglova.

2. Interaktivni prikaz podataka: Dodavanje mogućnosti interaktivnog prikaza podataka na grafu mreže bi moglo biti korisno. Na primer, omogućavanje korisniku da pregleda informacije o svakom vodu u mreži ili da prikaže statistike o opterećenju ili kvalitetu napajanja na određenim tačkama mreže. Ovo bi korisnicima pružilo detaljniji uvid u stanje elektrodistributivne mreže i olakšalo donošenje odluka.

3. Korisnički interfejs i personalizacija: Dalje usavršavanje aplikacije može uključivati poboljšanje korisničkog interfejsa i mogućnosti personalizacije. Na primer, omogućavanje korisniku da prilagodi izgled i raspored elemenata na ekranu ili da izabere temu koja odgovara njihovim preferencijama. Ovo bi poboljšalo korisničko iskustvo i omogućilo korisnicima da prilagode aplikaciju prema svojim potrebama.

4. Adaptacija sistema za korišćenje dodatnih XML datoteka koje opisuju elektrodistributivne mreže za različite regione, ne samo za Novi Sad. Ovo bi omogućilo korisnicima da vizualizuju i analiziraju elektrodistributivne mreže iz različitih delova sveta, pružajući im širu sliku o različitim konfiguracijama mreža i njihovim karakteristikama. Dodavanje podrške za različite XML datoteke bi povećalo fleksibilnost i korisnost aplikacije, čineći je primenljivom na globalnom nivou.

LITERATURA

[1] *https://www.ams.org/journals/bull/1973-79-03/S0002-9904-1973-13173-8/*

[2] *https://www.w3schools.com/xml/dom_nodetype.asp*

[3] *https://learn.microsoft.com/en-us/visualstudio/windows/?view=vs-2022*

[4] *https://learn.microsoft.com/en-us/dotnet/csharp/*

[5] *https://learn.microsoft.com/en-us/dotnet/desktop/wpf/?view=netdesktop-7.0*

[6] *http://www.blackwasp.co.uk/WPFPolygon.aspxAddison-Wesley*

[7] *http://www.blackwasp.co.uk/WPFEllipse.aspx*

[8] https://learn.microsoft.com/en-us/dotnet/desktop/wpf/graphics-multimedia/3-d-graphics-overview?view=netframeworkdesktop-4.8