UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA U NOVOM SADU

Marija Jovanović, PR 155/2018

2D VIZUALIZACIJA ELEKTRODISTRIBUTIVNE MREŽE SA FOKUSOM NA NAČIN VIZUALIZACIJE

Novi Sad,13.01.2023.

SADRŽAJ

1. OPIS REŠAVANOG PROBLEMA

U ovom dokumentu će biti prezentovano jedno rešenje projekta iz računarske grafike. Tema projekta je 2D vizalizacija elektrodistributivne mreže sa fokusom na način vizualizacije i manipulacije učitanim entitetima.

1.1. Konkretni problemi:

- Učitavanje elektroenergetske mreže iz već postojaće baze podataka (xml fajl sa podacima o koordinatama i karakteristikama entiteta) i njeno iscrtavanje na kanvas (pre crtanja se podrazumeva računanje i skaliranje koordinata iz xml fajla na koordinate kanvasa).
- Svaki učitani entitet ima panel o svojim podacima, različitu sliku u odnosu na tip entiteta (node,switch,substitution,line) i ne sme se preklapati sa drugim entitetima.
- Korišćenjem BFS algoritma pronaći najkraću putanju između entiteta koje spajamo linijama. Korisni podaci za BFS algoritam su Start i End Node. Za implementaciju BFS algoritma biće korišćena dva prolaza. U prvom prolazu se crtaju samo one linije koje mogu biti nacrtane bez presecanja, a u drugom prolazu linije za koje u prvom prolazu nije bilo moguće naći put bez presecanja i tada se one iscrtavaju uz označavanje tačaka preseka.

1.2. Dodatni problemi:

- Implementirati bojenje dva entiteta spojena istim vodom radi lakšeg uočavanja na kanvasu.
- Implementirati zumiranje/odzumiranje kanvasa.
- Implementirati crtanje elipse,poligona i teksta.
- Implementirati dodatne opcije: undo,redo,clear.
- Implementirati sakrivanje/prikazivanje neaktivnog dela mreže.
- Implementirati promenu transparentnosti boje svakog entiteta (osim linije) u odnosu na broj konekcija koji poseduje.
- Implementirati bojenje linija različitim bojama u zavisnosti od materijala i otpornosti voda kojeg ta linija predstavlja.
- Implementirati mogućnost korišćenja slika za prikaz entiteta u mreži.
- Implementirati opciju za čuvanje snimka ekrana.

2. OPIS KORIŠĆENIH TEHNOLOGIJA I ALATA

U ovom poglavlju biće dat sažet opis korišćenih tehnologija na projektu.

2.1. Visual Studio [1]

Visual Studio IDE je alat za editovanje, debagovanje, bildovanje koda i pablikovanje aplikacije. U sebi sadrži kompajlere, alate za automatsko završavanje koda, alate za grafičke dizajnere i razne druge dodatke koji pomažu pri izradi projekata.

Ovaj alat je izabran iz razloga jer je besplatan za studente i na veoma lak način se u njega mogu dodati potrebne ekstenzije i biblioteke za rad.

2.2. Programski jezik C# [2]

C# je moderan,objektno-orjentisan i tipno siguran programski jezik. Programerima omogućava pravljenje mnogo tipova sigurnih i robusnih aplikacija koje se pokreću u .NET-u. Njegovo poreklo dolazi iz familije C-olikih programskih jezika kao što su C, C++, Java, JavaScript.

Izabran je zbog manje kompleksnosti u odnosu na njegove prethodnike (C, C++) i zbog lakoće njegovog korišćenja u prethodno izabranom Visual Studio-u kao i u mnoštvu vec postojaćih rešenja na slične teme kojima se bavimo.

2.3. WPF – Windows Presentation Foundation [3]

WPF je okvir za korisnički interfejs koji je nezavisan od rezolucije i koristi vektorski baziranu osnovu za renderovanje slike. Napravljen je da koristi prednosti modernog hardvera za grafički prikaz. Neke od funkcionalnosti koje podržava su: XAML, kontrole, povezivanje podataka, 2D, 3D grafiku, animacije,stilove,templejte...

Izabran je zato što u sebi vec sadrži mnoštvo nama potrebnih funkcionalnosti kao i zbog kompatibilnosti sa C#-om.

3. OPIS REŠENJA PROBLEMA

3.1. Opis struktura podataka

3.1.1. Klase

- Row – klasa koja u sebi sadrži listu svih entiteta u celoj koloni za datu vrstu.

```
10 references
public class Row
{
    14 references
    public int Id { get; set; }
    2 references
    public double Value { get; set; }
    16 references
    public List<Column> Columns { get; set; }

    3 references
    public Row(int id, double value, List<Column> columns)
    {
        Id = id;
        Value = value;
        Columns = columns;
    }
}
```

Slika 1 – Klasa Row

- Column – jedan element iz kolone. Sadrži sledeca polja:

```
public class Column : ICloneable
{
    14 references
    public int Id { get; set; }
    2 references
    public double Value { get; set; }
    10 references
    public bool Taken { get; set; }

    3 references
    public Column(int id, double value)
    {
        Id = id;
        Value = value;
        Taken = false;
    }

    3 references
    public object Clone()
    {
        return this.MemberwiseClone();
    }
}
```

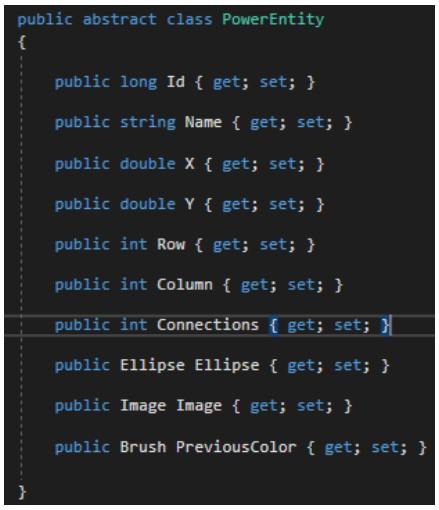
Slika 2 – Klasa Column

- RowColumn – klasa pomoću koje predstavljamo koordinate za crtanje koje smo pronašli pomoću BFS algoritma.

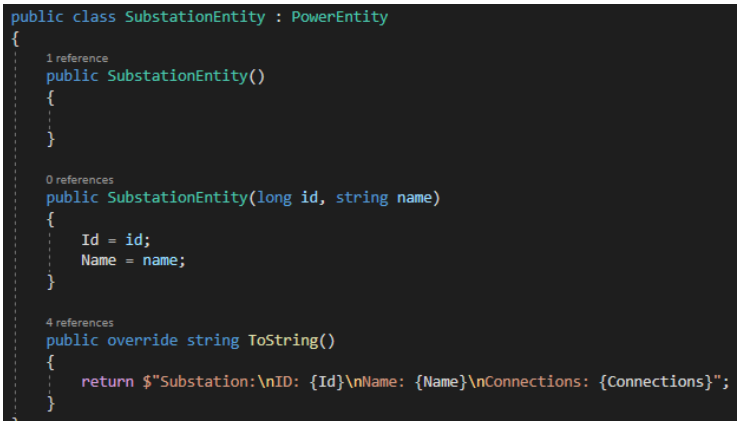
```
public class RowColumn
{
    25 references
    public int Row { get; set; }
    27 references
    public int Column { get; set; }

    7 references
    public RowColumn Parent { get; set; }

    11 references
    public RowColumn(int row, int column, RowColumn parent = null)
    {
        Row = row;
        Column = column;
        Parent = parent;
    }

    1 reference
    public RowColumn()
    {

    }
}
```

Slika 3 – Klasa RowColumn

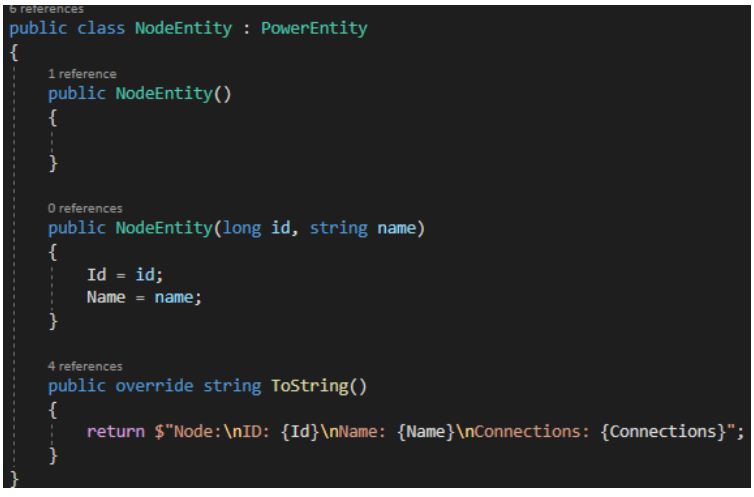
- PowerEntity – roditeljska klasa za podaklase „SubstitutionEntity“, „NodeEntity“ i „SwitchEntity“.



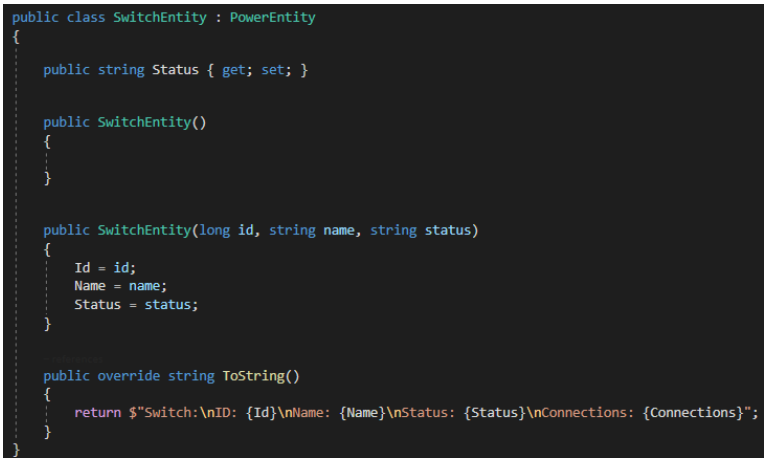
Slika 4 – Klasa PowerEntity



Slika 5 – Klasa SubstitutionEntity



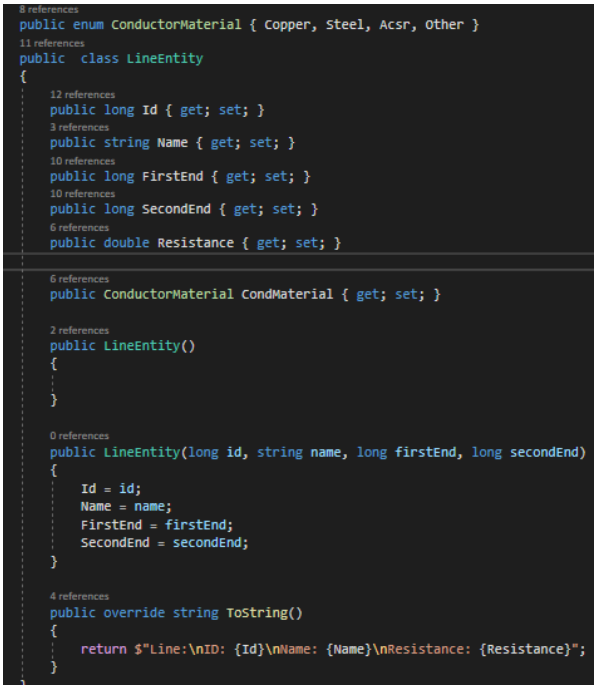
Slika 6 – Klasa NodeEntity



```
public class SwitchEntity : PowerEntity
{
    public string Status { get; set; }

    public SwitchEntity()
    {
    }

    public SwitchEntity(long id, string name, string status)
    {
        Id = id;
        Name = name;
        Status = status;
    }

    public override string ToString()
    {
        return $"Switch:\nID: {Id}\nName: {Name}\nStatus: {Status}\nConnections: {Connections}";
    }
}
```

Slika 7 – Klasa SwitchEntity

- LineEntity – klasa koja predstavlja jedan entitet linije iz ,,Geographic.xml'' fajla.

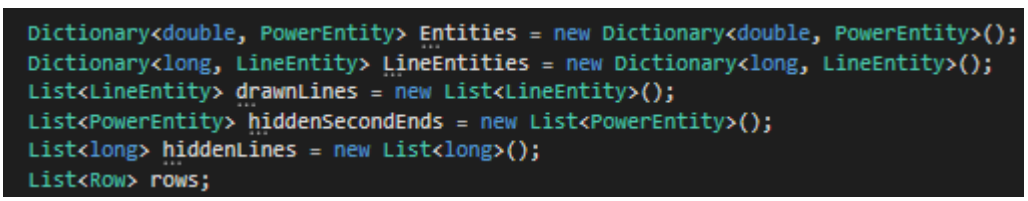


```
8 references
public enum ConductorMaterial { Copper, Steel, Acsr, Other }
11 references
public  class LineEntity
{
    12 references
    public long Id { get; set; }
    3 references
    public string Name { get; set; }
    10 references
    public long FirstEnd { get; set; }
    10 references
    public long SecondEnd { get; set; }
    6 references
    public double Resistance { get; set; }

    6 references
    public ConductorMaterial CondMaterial { get; set; }

    2 references
    public LineEntity()
    {
    }

    0 references
    public LineEntity(long id, string name, long firstEnd, long secondEnd)
    {
        Id = id;
        Name = name;
        FirstEnd = firstEnd;
        SecondEnd = secondEnd;
    }

    4 references
    public override string ToString()
    {
        return $"Line:\nID: {Id}\nName: {Name}\nResistance: {Resistance}";
    }
}
```

Slika 8 – Klasa LineEntity

3.1.2. Globalne promenljive

U projektu su korišćene sledeće strukture podataka:



```
Dictionary<double, PowerEntity> Entities = new Dictionary<double, PowerEntity>();
Dictionary<long, LineEntity> LineEntities = new Dictionary<long, LineEntity>();
List<LineEntity> drawnLines = new List<LineEntity>();
List<PowerEntity> hiddenSecondEnds = new List<PowerEntity>();
List<long> hiddenLines = new List<long>();
List<Row> rows;
```

Slika 9 – Strukture podataka

- Entities – rečnik za čuvanje svih entiteta učitanih iz ,,Geographic.xml'' fajla, a koji su tipa ,,PowerEntity''. Sadržaj rečnika su key-value parovi pri čemu key predstavlja identifikator entiteta, a value je sam entitet.

- LineEntities – rečnik za čuvanje entiteta učitanih iz ,,Geographic.xml'' fajla koji su tipa ,,LineEntity''. Sadržaj rečnika su key-value parovi pri čemu key predstavlja identifikator linije (voda),a value predstavlja sam entitet.
- drawnLines – lista nacrtanih linija na kanvasu. Ova lista nam omogućava da izvršimo proveru da li je neka linija već nacrtana na kanvasu kako ne bi bespotrebno bila crtana više puta.
- hiddenSecondEnds – lista onih entiteta koji bi trebalo da budu skriveni sa kanvasa kada se klikne na dugme ,,Hide inactive'' (zadatak skirvanja neaktivnog dela mreže). Ova lista nam pomaže pri ponovnom prikazivanju istih entiteta kada se klikne na dugme ,,Show inactive''.
- rows – lista koja u sebi sadrži sve entitete sa originalnim koordinatama i podatak da li je nešto već nacrtano na tim specifičnim koordinatama.

Strukture podataka korišćene za neke od dodatnih zadataka:

```
public static List<UIElement> UndoObjects;
public static List<UIElement> listOfSllDrawnObjectsOnCanvas;
public static List<Point> PolygonPoints;
```

Slika 10 - Dodatne strukture podataka

- UndoObjects – lista u kojoj se čuvaju objekti koji su uklonjeni sa kanvasa u slučaju da želimo da ih ponovo prikažemo.
- listOfsllDrawnObjectsOnCanvas – svaki put kada dodamo neki objekat na kanvas (elipsa, poligon, tekst) čuvamo u ovoj listi. Kroz ovu listu prolazimo kada želimo da uklonimo sve objekte sa kanvasa (dugme ,,Clear'').
- PolygonPoints – lista koja pomaže pri crtanju poligona na kanvasu tako što u sebi čuva tacke koje čine ivice poligona kojeg želimo da prikažemo.

3.1.3. Geographic.xml

Ovaj fajl predsatvlja bazu svih entiteta zapisanih u xml formatu. Osnovni podaci koji se nalaze u jednom od xml objekata su: id, ime, koordinate. Primer jednog od xml objekata iz ovog fajla:

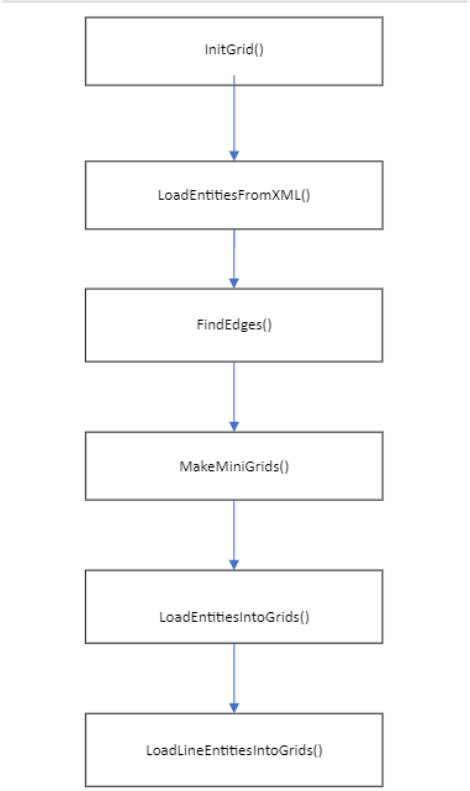
```
<Substations>
  <SubstationEntity>
    <Id>38094</Id>
    <Name>TSH 20</Name>
    <X>413321.68355993</X>
    <Y>5009020.27018182</Y>
  </SubstationEntity>
  <SubstationEntity>
    <Id>38095</Id>
    <Name>TSH 21</Name>
    <X>406884.872804481</X>
    <Y>5009357.24759746</Y>
  </SubstationEntity>
```

Slika 11 – Geographic.xml

3.2. Algoritam učitavanja entiteta

Zbog kompleksnosti samog učitavanja njegova funkcionalnost će biti opisana u nezavisnom poglavlju. Prilikom pritiska na dugme ,,Load entities" dešava se sledeći tok poziva funkcija:

Slika 12 – Dugme ,,Load entities“



Slika 13 – Graf toka poziva funkcije

3.2.1. Funkcija InitGrid()

Funkcija inicijalizuje broj vrsta i kolona u gridu koristeći rezoluciju main-window-a.

3.2.2. Funkcija LoadEntitiesFromXML()

Funkcija služi da se iz ,,Geographic.xml" fajla učitaju svi postojaći entiteti (node, switch, substitution, line) i skladišti te entitete u predviđene strukture podataka.

3.2.3. Funkcija FindEdges()

Funkcija vrši skaliranje postojaćeg u dovoljan koordinatni sistem prolaskom kroz koordinate entiteta (traže se minimalna i maksimalna koordinata po obe ose).

3.2.4. Funkcija MakeMiniGrids()

Funkcija pravi matricu u kojoj će svaki element biti minimalni deo grida (,,gridić") koji u sebi može da sadrži neki od entiteta. Za izračunavanje broja elemenata po vrsti i po koloni se koriste podaci izračunati u prethodnoj funkciji. Dodatno se vrši inicijalizacija matrice.

3.2.5. Funkcija LoadEntitiesIntoGrids()

Funkcija prolazi kroz listu učitanih entiteta (svih osim linija) i u zavisnosti od koordinata pozivaju se već postojaći API-ji za crtanje (Canvas.SetTop(), Canvas.SetLeft(), Canvas.SetZIndex()).

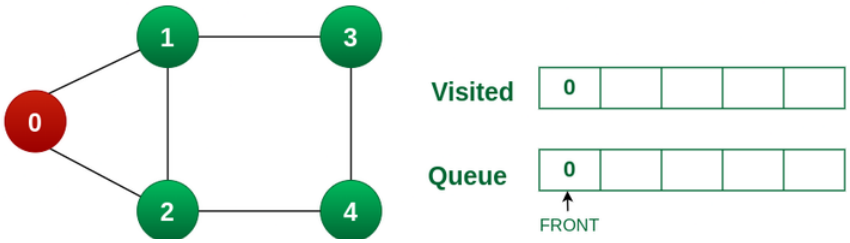
3.2.6. Funkcija LoadLineEntitiesIntoGrids()

Poslednja preostala stvar za iscrtavanje su linije. Zadatak ovog projekta je da se linije crtaju na način zahtevan u poglavlju **1.1.** U nastavku će biti dat kratak opis BFS algoritma i njegova modifikacija za konkretan problem.

3.2.6.1. **BFS** [4]

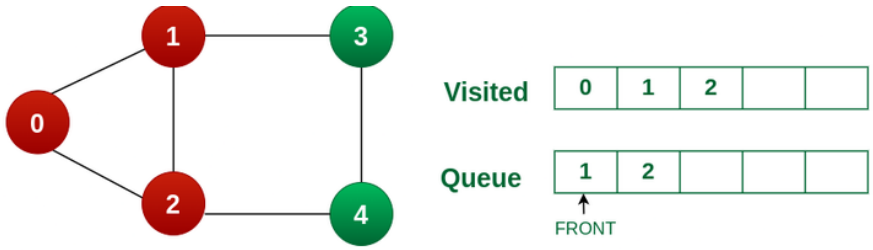
Jedan od najpopularnijih algoritama za pronalaženje najkraće putanje u besteżinskim grafovima.Njegova implementacija i primena na mnoge druge stvari je veoma jednostavna. Koraci algoritma:

- Delimo čvorove na posećene i neposećene.
- Kreće se od nasumično izabranog čvora i stavlja se u red:



Slika 14 – BFS primer 1

- Za prvi čvor u redu, redom posećujemo njegovu nikada posećenu decu i njih stavljamo u red.



Slika 15 – BFS primer 2

- Ukoliko trenutno obrađivani čvor nema nijedno dete koje nije posećeno, izbacujemo ga iz reda i obrađujemo decu narednog čvora.
- Algoritam se ponavlja sve dok ne posetimo sve čvorove u grafu (dok red ne bude potpuno prazan).
- Najkraća putanja se traži tako što primenimo BFS algoritam nad početnim čvorom i stanemo kada dođemo do krajnjeg čvora. Jedina dodatna stvar potrebna za pamćenje putanje je da imamo niz/listu u kojoj će biti prethodnik svakog od posećenih čvorova.

3.2.6.2. Modigikacija BFS algoritma

Naša implementacija BFS algoritma se sastoji iz dva prolaza:

- **Prvi prolaz:** Osnovna ideja BFS algoritma primenjena nad matricom ,,gridića'' koja predstavlja ,,graf''. Kreće se od početnog čvora i pretraživanjem elemenata matrice oko njega traži se završni čvor. Dodatni zahtev prvog prolaza je spajanje samo onih čvorova prilikom čijeg crtanja vodova neće biti presečen neki drugi vod. Za iscrtavanje preostalih linija služi drugi prolaz.
- **Drugi prolaz:** Ponovo se izvršava BFS algoritam,ali ovaj put korišćenjem infomracije da li je u trenutnom ,,gridiću'' iz najkraće putanje

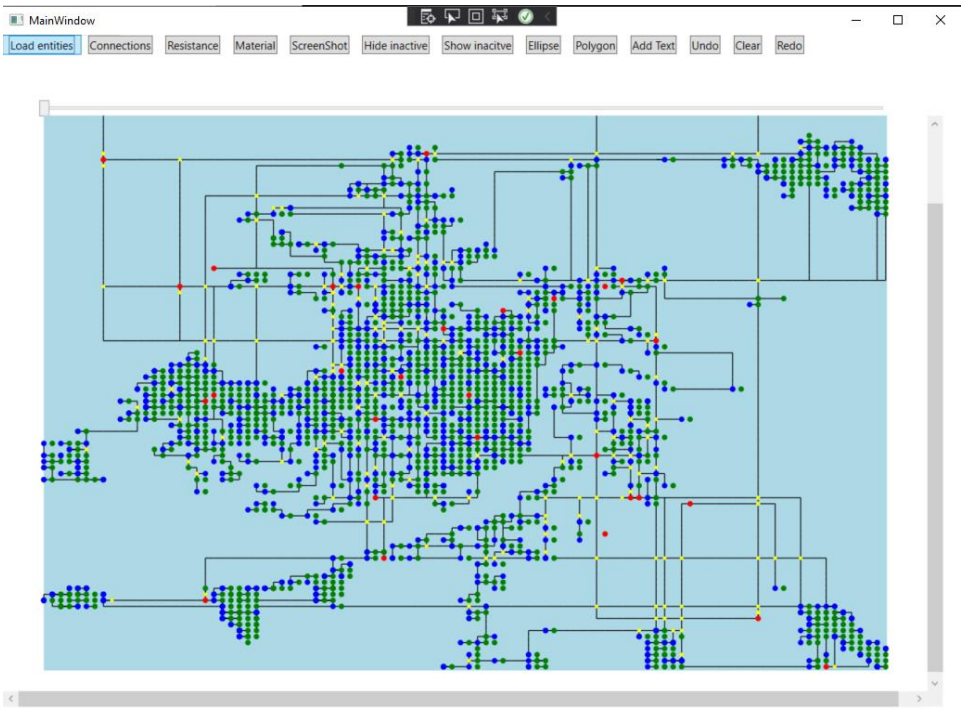
već nacrtana linija koja spaja neka druga dva čvora. Ukoliko je to tačno nastavlja se traženje najkraće putanje uz označavanje linije preseka.

3.3. Primeri upotrebe programa

3.3.1. Glavne funkcionalnosti

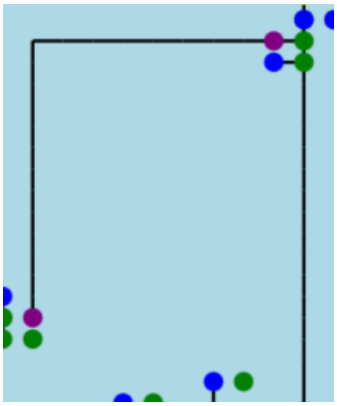
U nastavku će biti prikazane najvažnije funkcionalnosti pri korišćenju aplikacije.

1. Klikom na dugme ,,Load entities" učitaju se svi entiteti na kanvas. Na slici je prikazan kanvas sa svim učitanim entitetima iz ,,Geographic.xml" fajla.



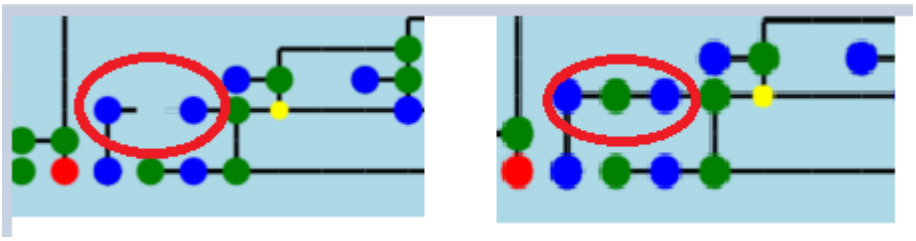
Slika 16 – Kanvas sa učitanim entitetima

2. Desnim klikom na vod možemo videti kako se njegovi čvorovi boje ljubičastom bojom radi lakšeg uočavanja.



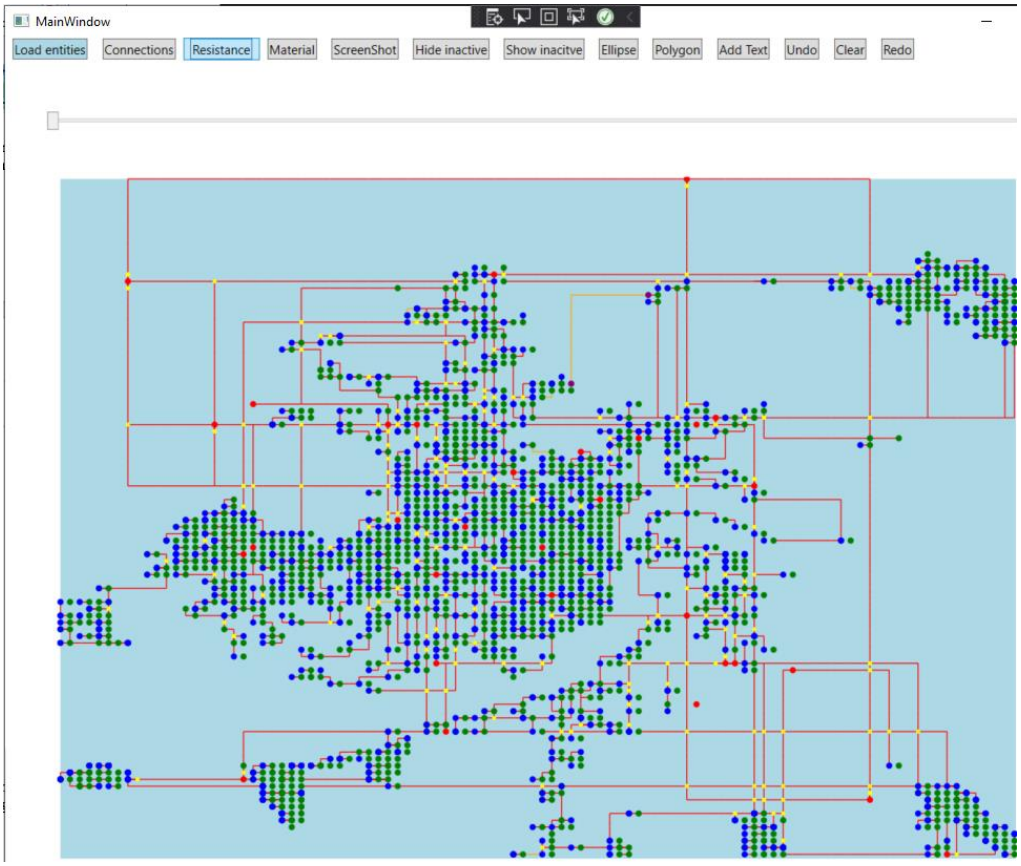
Slika 17 -Simulacija desnog klika na vod

3. Kada kliknemo na dugme ,,Hide inacitve" sa kanvasa nestane neaktivni deo mreže,a prilikom kliktanja na dugme ,,Show inactive" možemo da vratimo prethodni prikaz.



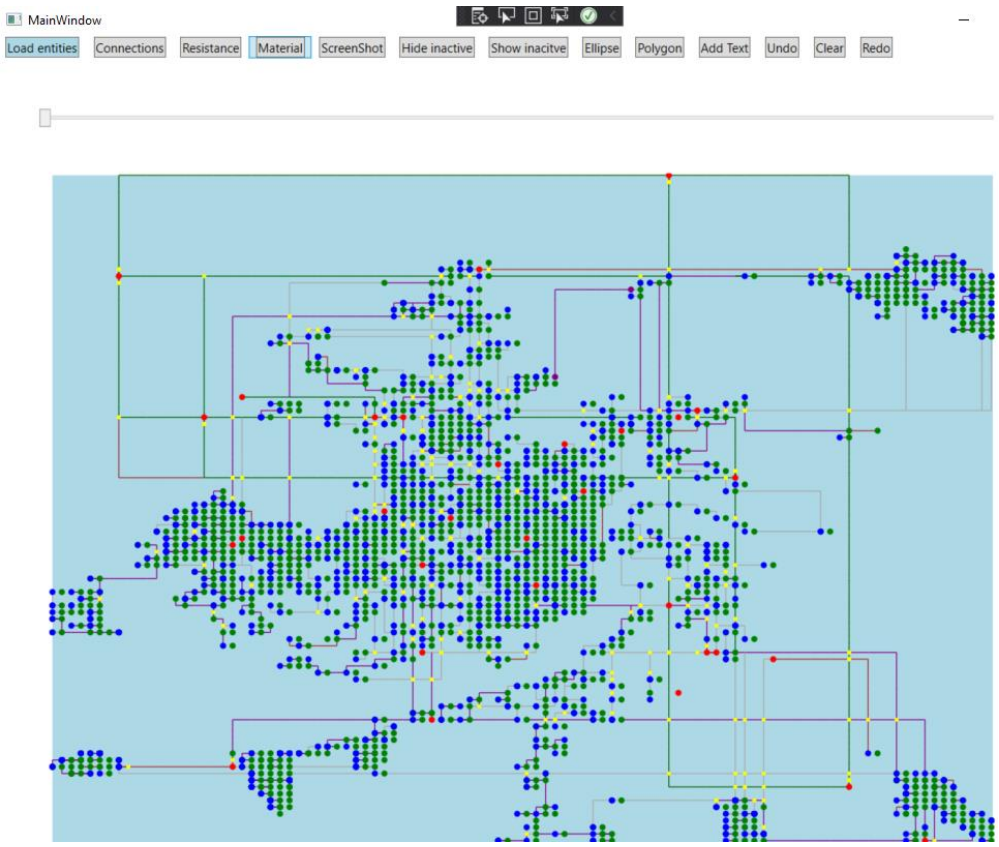
Slika 18 – Neaktivni deo mreže

4. Prilkom klika na dugme ,,Resistance" vodovi se oboje različitim bojama u zavisnosti od njihove otpornosti. Ponovnim klikom na isto dugme možemo vratiti prvobitan izgled kanvasa sa učitanim entitetima (prva slika iz koraka 1).



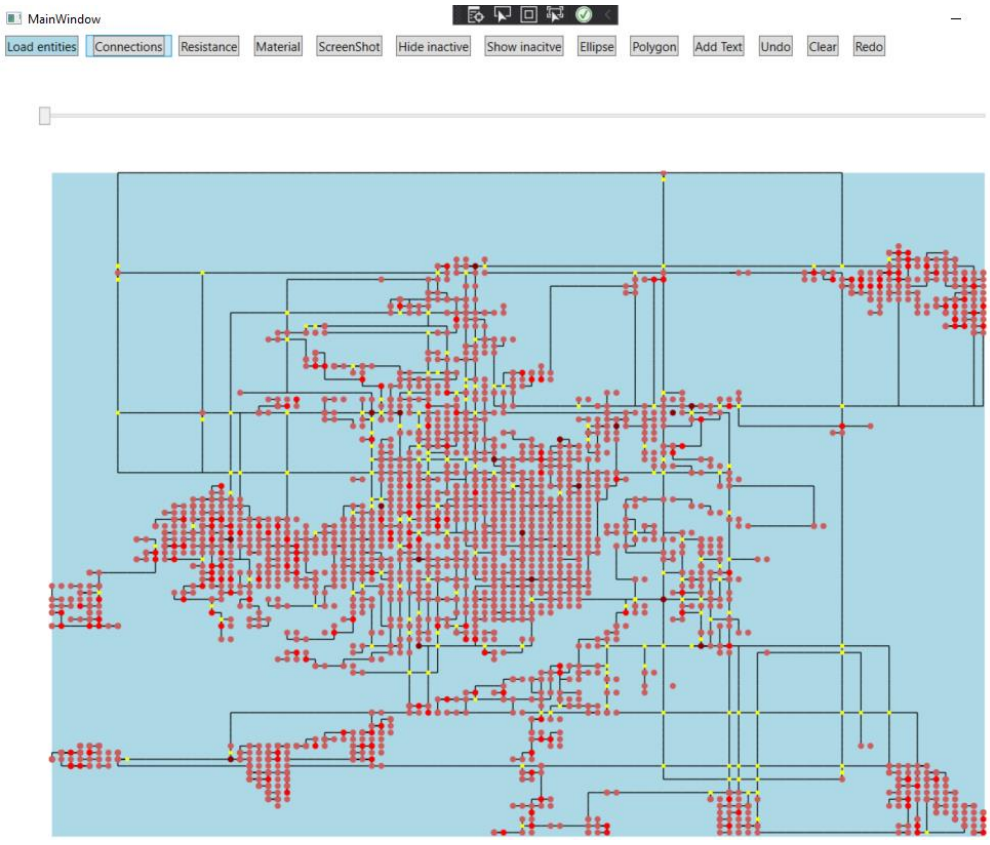
Slika 19 – Boja vodova na osnovu njihove otpornosti

5. Prilikom klika na dugme ,,Material'' vodovi se oboje različitim bojama u zavisnosti od materijala od kojeg su napravljeni. Ponovnim klikom na isto dugme možemo vratiti prvobitan izgled kanvasa sa učitanim entitetima (prva slika iz koraka 1).



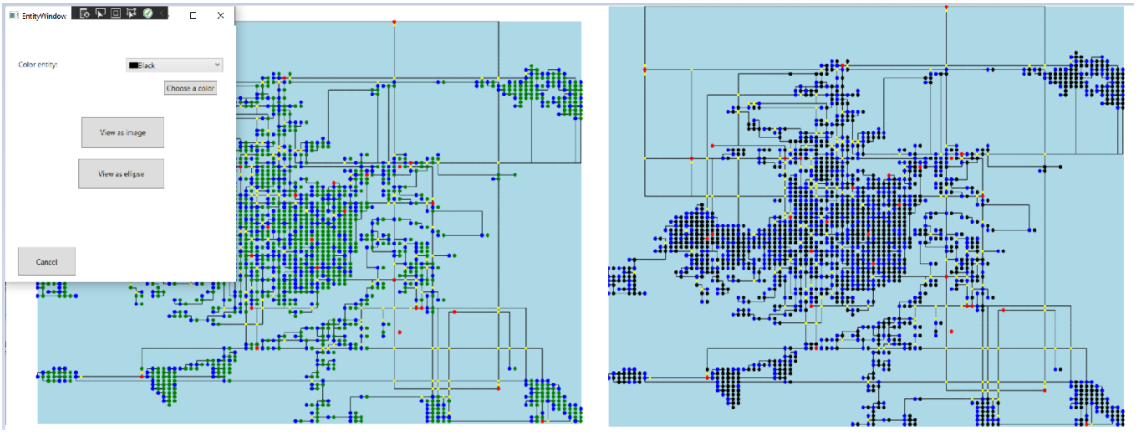
Slika 20 – Boja vodova na osnovu njihovog materijala

6. Klikom na dugme ,,Connections'' čvorovi se boje crvenom bojom različitog intenziteta u zavisnosti od broja konekcija čvora. Ponovnim klikom na isto dugme možemo vratiti prvobitan izgled kanvasa sa učitanim entitetima (prva slika iz koraka 1).



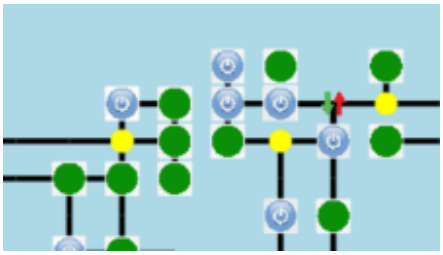
Slika 21 – Boja čvorova na osnovu broja konekcija

7. Levim klikom na bilo koju elipsu (čvor) otvara nam se novi prozor pomoću kojeg možemo da izaberemo boju kojom će se obojiti svi čvorovi koju su istog tipa kao čvor na čiju smo elipsu kliknuli.



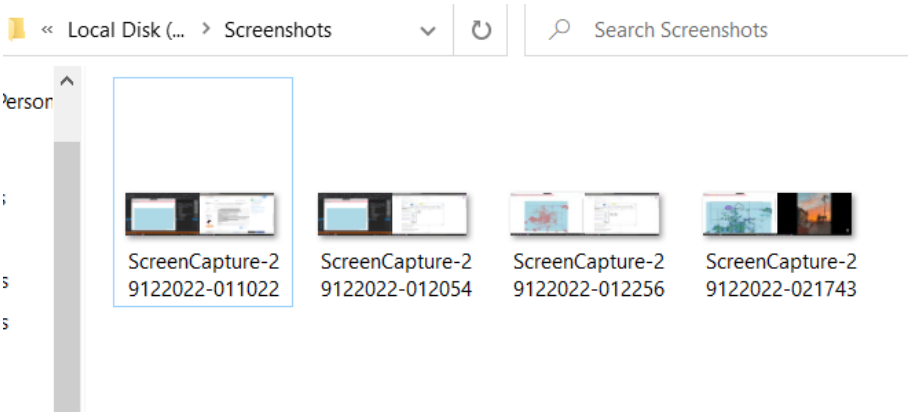
Slika 22 – Bojenje entiteta mreže

8. Prethodno pomenuti prozor nam takođe nudi opciju da čvorove predstavimo pomoću slika umesto elipsi. U skladu sa tim, za čvorove tipa ,,Node'' imamo sliku sa zelenim krugom, za čvorove tipa ,,Switch'' imamo sliku plavog prekidača, dok za čvorove tipa ,,Substitution'' imamo sliku zelene i crvene strelice.



Slika 23 – Predstavljanje entiteta pomoću slika

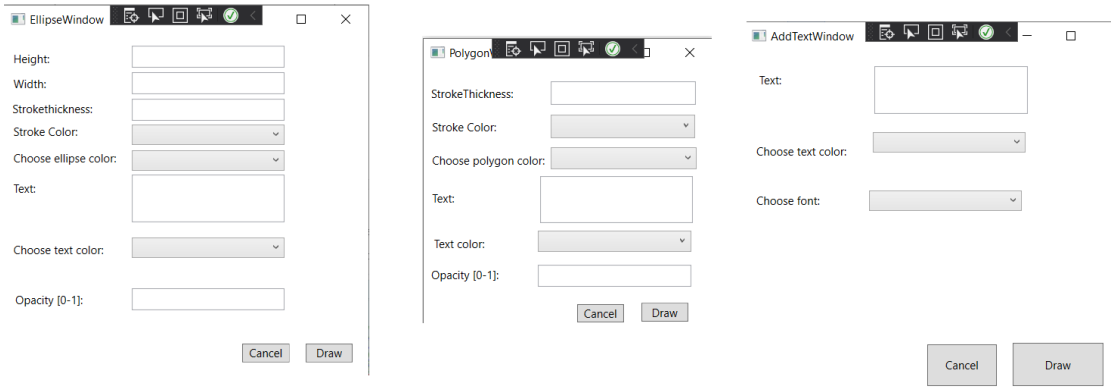
9. Klikom na dugme ,,ScreenShot'' čuva se slika trenutnog prikaza ekrana sa datumom na lokaciji ,, C:\Screenshots''.



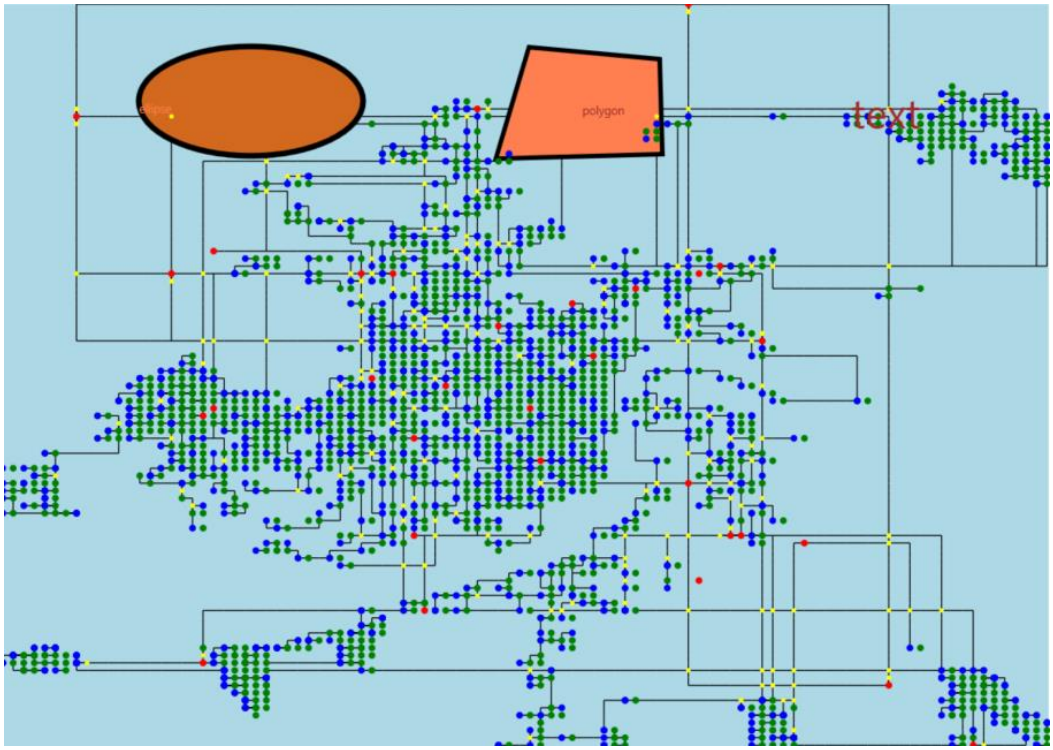
Slika 24 – ScreenShot

3.3.2. Dodatne funkcionalnosti

1. Levim klikom na dugme ,,Ellipse" , ,,Polygon" ili ,,Add Text", a zatim desnim klikom na kanvas otvara se prozor u okviru kojeg zadajemo osobine i parametre naše elipse,poligona ili teksta. Desnim klikom na kanvas se navodi pozicija od koje kreće iscrtavanje objekata (za poligon je potrebno imati bar 3 tacke tj. 3 takva klika pri čemu sledi jos jedan levi klik bilo gde na kanvasu kako bi se otvorio prozor).

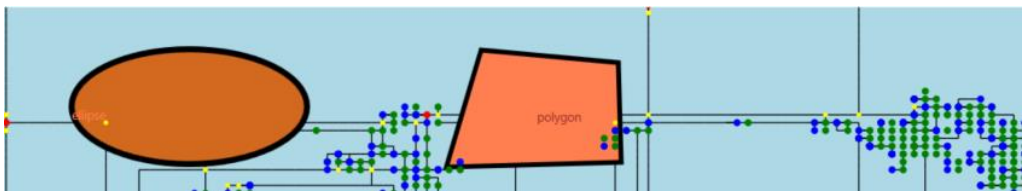


Slika 25 – Prozori za elipsu,poligon i tekst



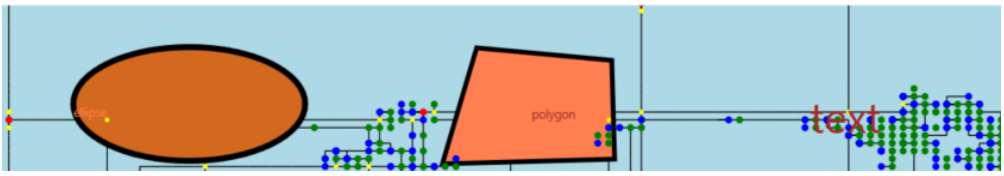
Slika 26 – Prikaz elipse,poligona i teksta na kanvasu

2. Klikom na dugme ,,Undo" briše se poslednji nacrtan objekat. Ukoliko je prethodno kliknuto dugme ,,Clear" onda ova opcija vraća sve prethodno obrisane objekte sa kanvasa.



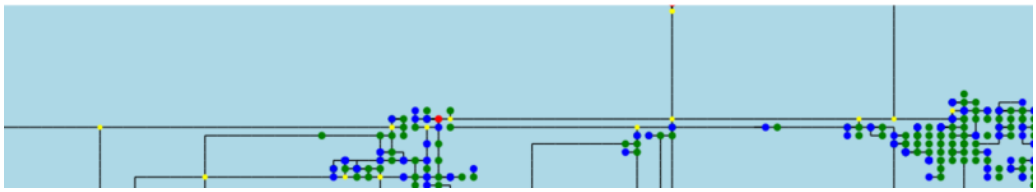
Slika 27 – Undo opcija

3. Klikom na dugme ,,Redo" možemo da vratimo poslednji obrisan objekat.



Slika 28 – Redo opcija

4. Klikom na dugme ,,Clear" uklanjaju se svi nacrtani objekti sa kanvasa.



Slika 29 – Clear opcija

4. PREDLOZI ZA DALJA USAVRŠAVANJA

U prethodnim poglavljima ste mogli da vidite jedno od rešenja za vizualizaciju elektroenergetske distributivne mreže.

Ovaj projekat nije savršen i otvoren je za dalja unapređenja kao što su:

- Testiranje samog projekta za više različitih setova podataka (u projektu je korišćen samo jedan – Geographic.xml). Razlog testiranja sa različitim setovima podataka bi bio da se testiraju krajnje granice softvera, npr: set podataka bez linija, bez vodova, sa entitetima koji imaju ogroman broj konekcija, različite veličine fajlova (od onih sa samo par entiteta do onih sa nekoliko miliona entiteta, prilagoditi algoritam veličini fajla zbog iskorišćenosti hardvera).
- Optimizacija brzine rada nekih delova koda kao što su: BFS, crtanje, brisanje pojedinih elemenata, njihovo bojenje itd.
- Crtanje elektroenergetske mreže na pravu mapu i sinhronizacija sa google mapama.
- Pisanje koda koji je mnogo više generički nego trenutno napisani zbog lakše implementacije novih projekata koji u sebi sadrže drugačije strukture podataka i tipova entiteta.
- Dodavanje konfiguracionih fajlova čijom promenom možemo editvati rezoluciju kanvasa, skaliranje podeljenih poligona, ograničavanje maksimalnog broja entiteta na mreži, davanja prioriteta crtanja entiteta na samu mrežu.

5. LITERATURA

[1] Visual Studio IDE
[2] A tour of the C# language
[3] Desktop Guide (WPF .NET)
[4] Breadth First Search or BFS for a Graph