УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Јован Гверо, ПР95-2017

3Д ПРИКАЗ ЕЛЕКТРОЕНЕРГЕТСКЕ МРЕЖЕ

ПРОЈЕКАТ
- Примењено софтверско инжењерство (ОАС) -

Нови Сад, 03.07.2022

САДРЖАЈ

ОПИС РЕШАВАНОГ ПРОБЛЕМА

Циљ пројекта је приказ електроенергетске мреже у 3Д простору.

Елементи мреже се исцртавају на 2Д мапи која се налази на дну 3Д сцене на основу њихових координата из фајла „Geographic.xml“.

Доњи леви угао мапе има географска ширину 45,2325 и дужину 19,793909.

Горњи десни угао мапе има географску ширину 45,277031 и дужину 19,894459.

Елементи мреже који се налазе изван простора обухваћеног овим координатама се занемарују и неће бити приказани на мапи.

Ентитети се исцртавају као коцке тако да су сви делови коцке видљиви из свих углова односно да коцке немају делове који се не приказују.

Ентитети чије се позиције преклапају ће бити исцртани један изнад другог.

Водови који спајају ентитете се исцртавају на основу њиховог својства „Vertices“ које се налази у поменутом фајлу и које представља низ географских локација кроз које вод пролази на путу између елемената које спаја.

Водови се исцртавају тако да је њихов попречни пресек квадрат.

Боја којом се вод исцртава зависи од материја од којег је направљен.

Мапа треба да пружи одређене функционалности за остваривање интеракције са корисником као што су: померање мапе, зумирање и ротирање око центра.

Померање мапе се врши померањем миша док је притиснут његов леви тастера.

Зумирање мапе се врши помоћу средњег тастера миша односно „скрол“ точкића.

Ротирање мапе се врши померањем миша док је притиснут „скрол“ точкић.

Током интеракције са мапом сви елементи треба да остану на својим позицијама на мапи независно од позиције мапе, нивоа зумирања или угла ротације.

Поред ових функционалности треба омогућити да се кликом на неки ентитет испишу његове информације (идентификатор, име, тип) а у случају да је у питању вод потребно је променити боју ентитета које тај вод повезује.

Интерфејс такође треба да омогући приказивање/сакривање:

1) Свих ентитета осим водова у зависности од броја конекција:
   a) Од 0 до 3.
   b) Од 3 до 5.
   c) Преко 5.
2) Водова у зависности од отпорности:
   a) Од 0 до 1.
   b) Од 1 до 2.
   c) Преко 2.
3) Водова који излазе из елемената чији је статус „отворен“ односно „Open”.

ОПИС КОРИШЋЕНИХ ТЕХНОЛОГИЈА И АЛАТА

Microsoft Visual Studio је програмско окружење креирано од компаније Мајкрософт 1997. године.

Користи софтверске развојне платформе „Windows API“, „Windows Forms“, „Windows Presentation Foundation (WPF)“ и разне друге за развој апликација (рачунарских, мобилних, веб), веб-сајтова и веб-сервиса. [1]

Windows Presentation Foundation (WPF) је графички подсистем за рендеровање корисничког интерфејса у апликацијама заснованим на „Windows“ оперативним системима.

Предност WPF-а је јасно раздвајање корисничког интерфејса и пословне логике.

Интерфејс користи „XAML“, изведен из „XML“, да дефинише и повеже различите елементе интерфејса. Као декларативан језик, „XAML“ омогућава описивање понашања компоненти интерфејса (анимације, трансформације, ефекти...) без ослањања на програмски код пословне логике. [2]

.Net Framework је софтверска платформа која се може инсталирати на уређаје које покреће „Microsoft Winows“ оперативни систем. Он укључује велики број готових библиотека кодова за уобичајене проблеме у програмирању. [3]

C# је објектно-оријентисан програмски језик развијен од компаније Мајкрософт 2000. године као део развојног окружења „ .Net Framework 1.0“. Јако сличан програмским језицима „C“ и „C++“. [4]

За приказ и рад са 3Д сценом коришћени су и:

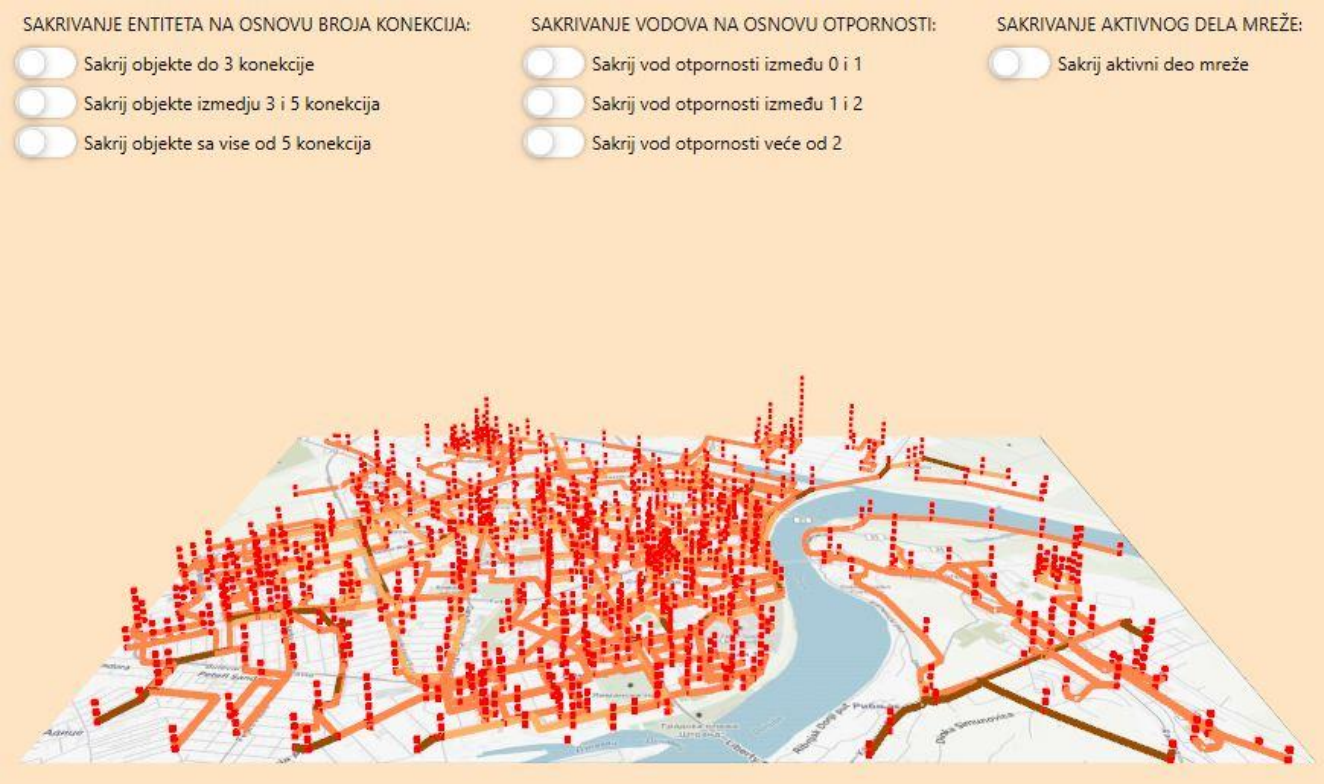
- **HelixViewport3D** – Елемент који омогућава приказивање 3Д сцене. Предност употребе овог елемента у односу на стандардни „Viewport3D“ ће бити описана у наставку текста. [5]
- **PerspectiveCamera –** Камера која омогућава видљивост сцене.
  Предност ове камере у односу на камеру са ортогоналном пројекцијом је у томе што се објекти приказују различитим величинама у зависности од удаљености од камере што доприноси реалнијем приказу сцене. [6]
- **AmbientLight –** Светло које омогућава видљивост сцене.
  Амбијентално светло обасјава целу сцену истим интензитетом из свих углова па нема сенки.
  Предност употребе овог светла у односу на остала је у једноставној имплементацији јер нема особине попут позиције и смера обасјавања већ је само потребно дефинисати боју. [7]

ОПИС РЕШЕЊА ПРОБЛЕМА

Покретањем апликације добија се главни прозор као на слици 1.

Прозор је сачињен из два дела. Први део чине „toggle“ дугмад намењена за интеракцију са корисником, односно за сакривање/приказивање ентитета и водова мреже. Дугмад су помоћу „stackpanel“ -а груписана у колоне тако да свака колона одговара једном типу сакривања/приказивања.

Други део прозора чини 3Д сцена са мапом на којој се исцртавају ентитети и водови мреже.



Слика 1 – Главни прозор

На слици примећујемо да су водови обојени различитим бојама у зависности од материјала од којег су направљени:

- *Steel* – „Coral“ боја.
- *Copper* – „Rajah“ боја.
- *ACSR* – „Saddle brown“ боја.
- *Other* – „Navy blue“ боја.

Апликација је структурално сачињена из 4 дела:

1. Читање података.
2. Мапирање елемената на мапу и њихово цртање.
3. Цртање водова који повезују елементе.
4. Интеракције са апликацијом:
   4.1. Хит тестирање.
   4.2. Сакривање/приказивање делова мреже.
   4.3. Трансформације сцене.

Читање података

Подаци са којима наша апликација ради се чувају унутар „Geographic.xml“ фајла па их је потребно прочитати. Како је структура овог фајла „xml“ подаци се чувају унутар тагова. Слика 2. приказује изглед једног елемента унутар фајла док слика 3. приказује класу која опсује елемент у апликацији.

```
<SubstationEntity>
  <Id>38094</Id>
  <Name>TSH 20</Name>
  <X>413321.68355993</X>
  <Y>5009020.27018182</Y>
</SubstationEntity>
```

Слика 2 – Подаци елемента у фајлу

```
public class PowerEntity
{
    private long id;
    private string name;
    private double x;
    private double y;
    private double boardx;
    private double boardz;
}
```

Слика 3 – Класа која описује елемент у апликацији

Рад са „xml“ фајлом се своди на коришћење постојећих класа:

- *XmlDocument* – Учитава цео фајл.
- *XmlNodeList* – Чита све инстанце неког тага из фајла.

На почетку апликације се променљивој типа „XmlDocument“ преко уграђене методе „Load“ као њен параметар проследи путања до фајла. Након овога наша променљива има све податке из фајла.

По учитавању података из фајла потребно је прочитати све елементе мреже (трафостанице, чворове и прекидаче). Читање неког елемента мреже се врши помоћу променљиве типа „XmlNodeList“ а као параметар уграђеној методи „SelectNodes“ се прослеђује име тага који одговара жељеном елементу мреже.

Након овога, у нашој променљивој типа „XmlNodeList“ ће остати само елементи жељеног типа. Проласком кроз ову листу се креирају жељени ентитети мреже и уколико задовољавају услов мапе и чувају за даље коришћење.

Елементи задовољавају услов уколико су њихове координате обухваћене мапом. Како су координате мапе задате географском ширином и дужином њених углова а елементи у фајлу описани УТМ координатама оне се морају конвертовати у географску ширину и дужину. Ово конвертовање се врши постојећом методом „ToLatLon“ па о њој овде неће бити речи.

Слика 4. приказује функцију задужену за учитавање једног елемента мреже. Остале функције су идентичне овој само се мења елемент који се чита.

```
public void LoadSubstations()
{
    xmlNodeList = xmlDocument.DocumentElement.SelectNodes("/NetworkModel/Substations/SubstationEntity");
    foreach (XmlNode xmlNode in xmlNodeList)
    {
        SubstationEntity sub = new SubstationEntity();
        sub.Id = long.Parse(xmlNode.SelectSingleNode("Id").InnerText);
        sub.Name = xmlNode.SelectSingleNode("Name").InnerText;
        sub.X = double.Parse(xmlNode.SelectSingleNode("X").InnerText);
        sub.Y = double.Parse(xmlNode.SelectSingleNode("Y").InnerText);
        double convertedx, convertedy;
        ToLatLon(sub.X, sub.Y, 34, out convertedx, out convertedy);
        if (convertedx >= 45.2325 && convertedx <= 45.277031)
        {
            if (convertedy >= 19.793909 && convertedy <= 19.894459)
            {
                elements.Add(sub);
                allelements.Add(sub);
            }
        }
    }
}
```

Слика 4 – Функција која учитава трафостанице

Сви елементи се чувају у 2 листе:

- *elements* – Елементи мреже. Користи се приликом провере валидности вода.
- *allelements* – Елементи и водови. Користи се приликом хит тестирања.

Учитавање водова је по структури готово идентично учитавању елемената са тим што провера валидности вода подразумева проверу ентитета које он спаја. Уколико се оба краја вода налазе у листи „elements“ вод се сматра валидним.

Да би се вод могао додати у листу „allelements“ и тиме олакшати хит тестирање класа која га описује такође наслеђује „PowerEntity“ класу.

```
<LineEntity>
 <Id>35126</Id>
 <Name>SEC_137438958224</Name>
 <IsUnderground>false</IsUnderground>
 <R>0.866</R>
 <ConductorMaterial>Acsr</ConductorMaterial>
 <LineType>Overhead</LineType>
 <ThermalConstantHeat>2400</ThermalConstantHeat>
 <FirstEnd>37898</FirstEnd>
 <SecondEnd>37899</SecondEnd>
 <Vertices>
   <Point>
     <X>411460.535231187</X>
     <Y>5017665.8507269</Y>
   </Point>
 </Vertices>
</LineEntity>
```

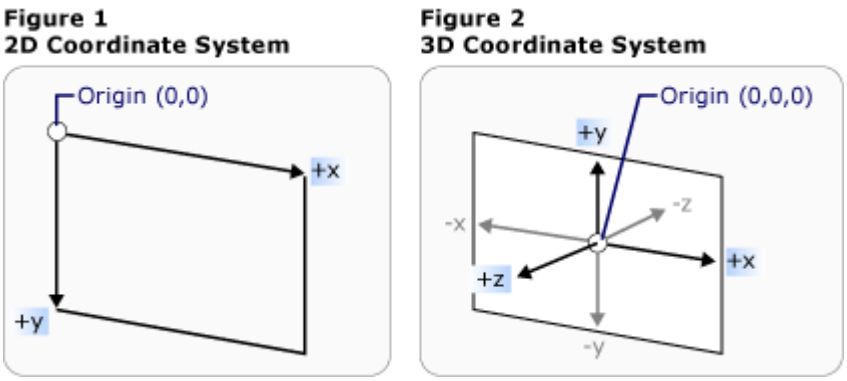
Слика 5 – Подаци вода у фајлу

```
public class LineEntity : PowerEntity
{
    private bool isUnderground;
    private float r;
    private string conductorMaterial;
    private string lineType;
    private long thermalConstantHeat;
    private long firstEnd;
    private long secondEnd;
    private List<Point> vertices;
}
```

Слика 6 – Класа која описује вод

Мапирање и цртање елемената

Да би разумели рад са 3Д сценом потребно је разумети 3Д координатни систем. На слици 7. примећујемо да се почетак 3Д координатног система налази у његовом центру. На слици се такође примећује координата „Z“ која представља „дубину“ сцене. [8]



Слика 7 – 2Д и 3Д координатни систем

Када смо учитали све елементе у листу потребно их је мапирати на мапу а потом и исцртати. Рачунање координата елемената се врши на основу њихових географских ширина и дужина и позиције мапе унутар 3Д сцене. Како је мапа 2Д и сви њени ћошкови имају „Y“ координату 0, потребно је израчунати „X“ и „Z“ координате елемената док ће „Y“ координата бити 0.

Мапирање се врши итерацијом кроз све елементе (листа „elements“) и рачунањем њихових 3Д координата. Када се израчунају 3Д координате неког елемента оне се додају у нову листу након чега се елемент црта на мапу.

```
public List<Point> boardpoints = new List<Point>();
```

Слика 8 – Листа 3Д координата свих елемената

За цртање елемента на мапу је задужена функција „DrawCube“ која као параметар прима редни број елемента који се црта. Како редослед елемената у листи „elements“ одговара редоследу у листи „boardpoints“ коришћењем овог параметра се могу добити 3Д координате из листе „boardpoints“ након чега се пролази кроз њу и проверава колико елемената се већ налази на тој позицији. На крају итерације добијена позиција представља центар основе коцке па се у односу на њу рачунају ћошкови коцке. Када израчунамо координате ћошкова, коцка се може нацртати.

Цртање коцке се своди на конструисање модела коцке, који ће се додати у листу свих модела а затим и на сцену. Разлог додавања модела, било да је елемент или вод, у листу модела, је имплементација хит тестирања. О овоме ће бити речи касније.

Цртање водова

Цртање водова се своди на итерацију кроз листу водова и спајање тачака. Први крај вода се спаја са првом позицијом унутар „Vertices“ листе, она са другом, и све до краја листе. Последња позиција из ове листе се спаја са другим крајем вода. Дефиниција функције задужене за спајање позиција је приказана сликом испод:

```
public void DrawLine(Point p1, Point p2, string material, string pozicijaelementa, LineEntity mle)
```

Слика 9 - Дефиниција функције за спајање позиција

Осим позиција које спаја и материјала од којег је вод сачињен, функција прима и два додатна параметра. Четврти параметар функције указује на којој се од ове две позиције налази елемент а на којој „Vertices“ позиција. Разликовање ова два типа позиција је круцијално да не би дошло до преклапања вода и елемента, односно да се не би стварале рупе у воду.

Како се вод састоји из делова, приликом цртања сваког дела се прослеђују подаци о воду којем тај део припада. Овим се омогућује исписивање истих информација приликом клика на било који део вода.

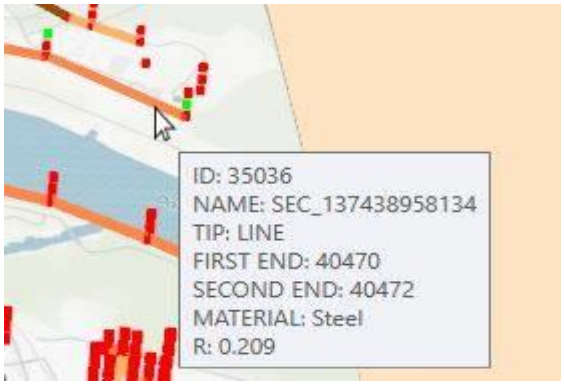
Хит тестирање

Хит тестирање је функционалност којом корисник кликом десног тастера миша на неки ентитет добија његове основне информације. Хит тестирање се имплементира тако што проверимо да ли је корисник кликнуо на било који постојећи модел.

Уколико је корисник кликнуо на модел, итерацијом кроз листу свих модела утврдимо позицију у листи на којој се налази модел на који је корисник кликнуо.

Како ова листа модела одговара листи свих елемената „allelements“(која прима и водове), преко позиције у листи модела приступимо листи „allelements“ и добавимо његове основне информације. Информације ентитета се исписују у близини курсора миша унутар „tooltip“-а.

У случају да корисник кликне на вод, осим исписивања његових информација, мења се боја ентитета које тај вод спаја.



Слика 10 – Хит тестирање

Приказивање и сакривање делова мреже

Корисник помоћу „toggle“ дугмади има могућност сакривања и поновног приказивања одређених делова мреже. У зависности од дела мреже, могуће је сакрити ентитете на основу броја конекција, водове на основу отпорности и водове који спајају прекидаче чији је статус „open“.

Како је у питању „toggle“ дугме, дефинишу се функције које се извршавају приликом сваког наредног активирања дугмета и приликом сваког наредног деактивирања.

Иницијално, сва дугмад су деактивирана па ће се сакривање вршити њиховим активирањем, односно функцијом додељеној „Checked“ атрибуту.

Поновно приказивање ће се вршити деактивирањем дугмета, односно функцијом додељеној „Unchecked“ атрибуту.

Приликом активирања дугмета које сакрива ентитете са мање од 3 конекције позива се функција „Od0do3_Checked“. Функција пролази кроз све елементе и рачуна њихов број конекција. Уколико је број конекција ентитета мањи од 3 на основу позиције ентитета у листи се приступа листи модела у сцени и одговарајућем моделу мења атрибут „Visibility“ на „Visibility.Hidden“. Овим се модел сакрива са сцене.

Деактивирањем овог дугмета се позива метода „Od0do3_Unchecked“. Она је структурално идентична методи која сакрива ентитет али се овде елемент поново приказује подешавањем „Visibility“ атрибута на „Visibility.Visible“.

Остале методе за сакривање/приказивање ентитет у зависности од броја конекција су идентичне само се мења параметар.

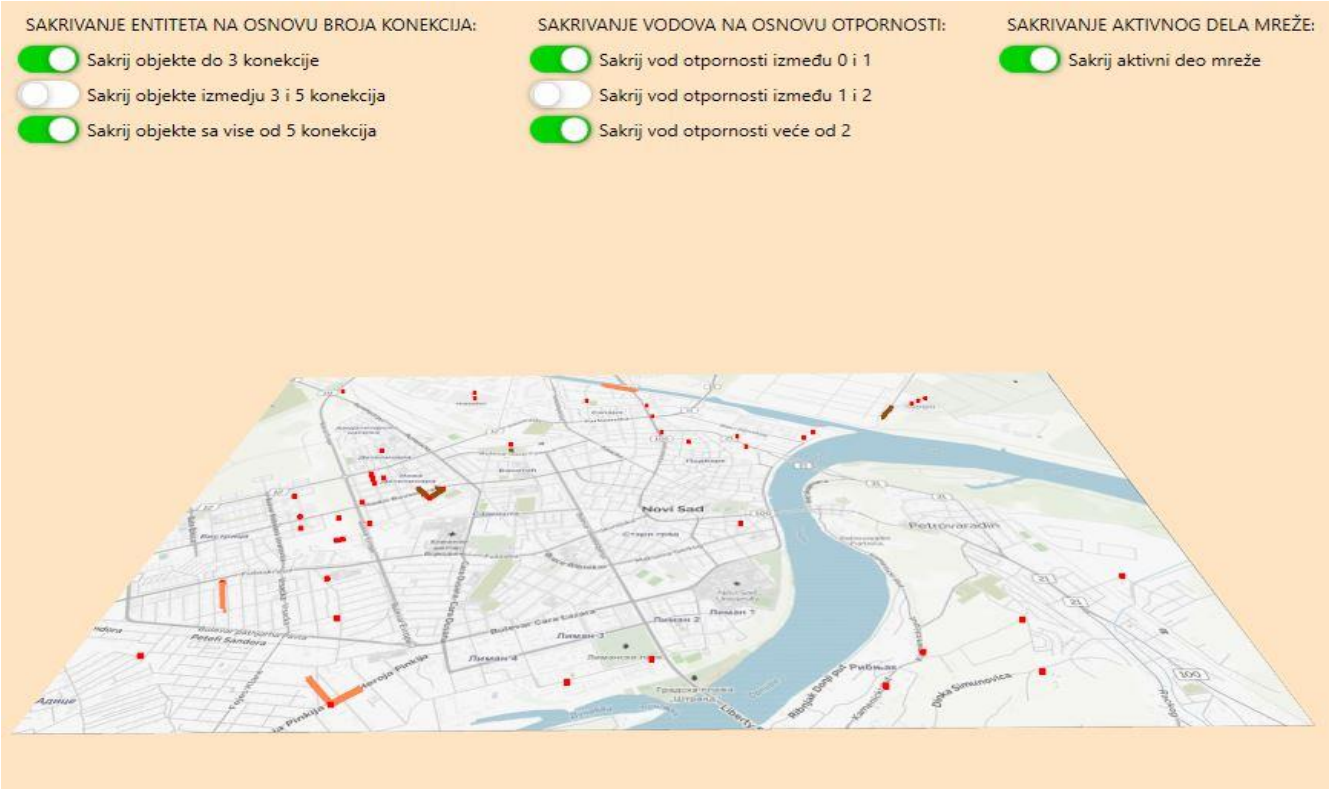
```
private void Od0Do3_Checked(object sender, RoutedEventArgs e)
{
    for(int i=0;i<elements.Count;i++)
    {
        int br_veza = CalculateNumberOfConnections(elements[i].Id);
        if(br_veza<3)
        {
            ModelUIElement3D model = viewport1.Children[i+3] as ModelUIElement3D;
            model.Visibility = Visibility.Hidden;
        }
    }
}
```

Слика 11 – Функција за сакривање ентитета са мање од 3 конекције

Разлог због којег се на позицију елемента додаје 3, приликом приступања моделима сцене је постојање неких модела пре модела првог ентитета. Ти модели су светло, камера и мапа па се повећањем за 3 они прескачу.

Сакривање и приказивање водова је готово идентично сакривању и приказивању ентитета али се као параметар проверава његова отпорност која се чува као атрибут „R“ у класи која описује вод.

Сакривање вода који спаја прекидач чије је статус „open“ подразумева пролазак кроз све водове. За сваки вод се пролази кроз све прекидаче и проверава да ли је тај прекидач један од крајева вода и да ли је отворен. Уколико су оба услова задовољена на већ објашњен начин се вод сакрива.



Слика 12 – Главни прозор након насумичне комбинације сакривања делова мреже

Трансформације сцене

Пре је било речи о „HelixViewport3D“ али је изостао разлог његовог коришћења. Предност коришћења ове сцене у односу на стандардну „Viewport3D“ се огледа у једноставној имплементацији метода за њено трансформисање. Ова сцена је намењена за рад са постојећим моделима па у себи има уграђена својства која то омогућују и олакшавају. Овим својствима се додаје гест миша који је за њих задужен.

```
viewport1.RotateGesture = new MouseGesture(MouseAction.MiddleClick);
viewport1.FixedRotationPoint = new Point3D(0, 0, 0);
viewport1.FixedRotationPointEnabled = true;
viewport1.PanGesture = new MouseGesture(MouseAction.LeftClick);
viewport1.PanGesture2 = null;
```

Слика 13 – Подешавање трансформација сцене

Ротирање мапе се подешава „RotateGesture“ својством. За ово својство биће одговоран гест миша када је притиснут „скрол“ точкић. Ротирање модела око одређене тачке у „HelixViewport3D“-у је могуће активирањем својства „FixedRotationPointEnabled“ и постављањем жељене 3Д тачке у својсвто „FixedRotationPoint“. Како је мапа постављена тако да јој је центар у координатном почетку ова тачка ће имати све три координате 0.

Померање мапе се врши подешавањем „PanGesture“ својства. Ово својство је остављено од стране креатора „HelixViewport3D“ класе да би програмери могли да дефинишу своја правила за померање модела. Сама класа има уграђен начин померања који је аутоматски постављен на клик „скрол“ точкића и такво померање је реализовано кроз „PanGesture2“ својство па се оно мора искључити.

Зумирање је уграђено у „HelixViewport3D“ и одговара ротирању „скрол“ точкића па га није потребно додатно дефинисати.

ПРЕДЛОЗИ ЗА ДАЉА УСАВРШАВАЊА

Апликација је прилично интуитивна за коришћење али има делова који се могу унапредити, не само са аспекта корисничког искуства већ и са аспекта перформанси.

Реализација сакривања и приказивања делова мреже помоћу „toggle“ дугмета омогућује висок ниво интуиције и приликом првог коришћења апликације.

Померање, ротирање и зумирање мапе се извршава на уобичајене начине па кориснику није потребно додатно време да се навикне на рад са апликацијом.

Предлог за побољшање корисничког искуства је цртање ентитета у једној равни. У тренутној верзији апликације ентитети који имају исту позицију се цртају једни изнад других па се губи на прегледности. Овај проблем је могуће решити тако што ће се ентитети на истим позицијама цртати једни до других.

Мана интерфејса је и то што недостаје објашњење различитих боја водова односно недостатак легенде која говори који материјал одговара којој боји.

Апликација се може надоградити и са аспекта перформанси, коришћењем бржих структура података од листи, на пример коришћењем речника.

ЛИТЕРАТУРА

[1] Microsoft Visual Studio - Wiki, Visual Studio

[2] Windows Presentation Foundation - Wiki, WPF

[3] .NET Framework - Wiki, .NET

[4] A Tour Of C# - Microsoft Docs, C#

[5] Helix Toolkit - hithub.io, HelixViewport3D

[6] Perspective Camera - Microsoft Docs, Perspective Camera

[7] Ambient Light - Microsoft Docs, Ambient Light

[8] 3D Graphics Overview, WPF 3D Graphics