|  |  |  |
| --- | --- | --- |
|  | 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, датум

Садржај

[ОПИС РЕШАВАНОГ ПРОБЛЕМА 1](#_Toc103777256)

[ОПИС КОРИШЋЕНИХ ТЕХНОЛОГИЈА И АЛАТА 2](#_Toc103777257)

[ОПИС РЕШЕЊА ПРОБЛЕМА 3](#_Toc103777258)

# 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.

У случају да вод спаја елементе који се налазе изнад других елемената посматра се елемент најближи мапи односно први елемент на тој позицији.

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
   1. Од 0 до 3
   2. Од 3 до 5
   3. Преко 5
2. Водова у зависности од отпорности
   1. Од 0 до 1
   2. Од 1 до 2
   3. Преко 2
3. Водова који излазе из елемената чији је статус „отворен“ односно „Open”.

# ОПИС КОРИШЋЕНИХ ТЕХНОЛОГИЈА И АЛАТА

**Microsoft Visual Studiо** 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.

**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 елементe. Као декларативан језик, „XAML“ омогућава описивање понашања компоненти интерфејса (анимације, трансформације, ефекти...) без ослањања на програмски код пословне логике.

**.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.

**C#** је објектно-оријентисан програмски језик развијен од компаније Мајкрософт 2000. године као део развојног окружења „ .Net Framework 1.0“.

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

* **HelixViewort3D** – Елемент који омогућава учитавање готових 3Д модела. Предност употребе овог елемента у односу на стандардни „ViewPort3D“ ће бити описана у наставку текста.
* **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.

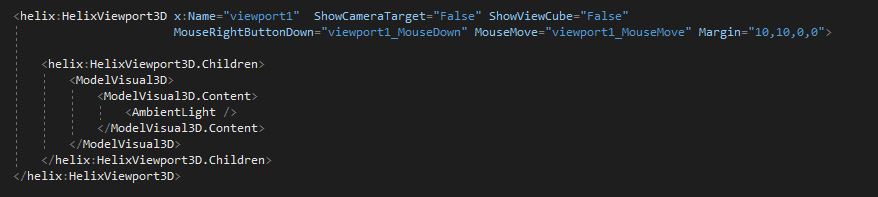
* **AmbientLight –** Светло које омогућава видљивост сцене.

Амбијентално светло обасјава целу сцену истим интензитетом из свих углова па нема сенчења.

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.

# ОПИС РЕШЕЊА ПРОБЛЕМА

Опис решења проблема ћемо започети од интерфејса односно од саме 3Д сцене и XAML језика који је описује.



ЛИСТИНГ 1 - HELIXVIEWPORT3D

За приказ сцене потребно је дефинисати елемент који може да прикаже сцену у 3Д простору. Изабрано је да тај елемент буде класе „HelixViewport3D“ ради лакше реализације интерактивних способности које сцена пружа кориснику (зумирање, померање, ротација).

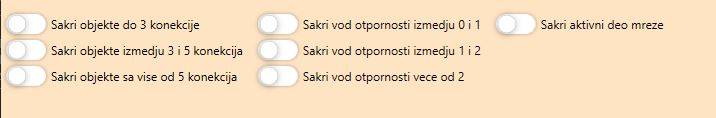
Како је намењен за приказ и манипулацију са већ готовим моделима, HelixViewport3D пружа и могућности као што су:

* *CameraTarget* – мета камере која се појављује на месту где смо се позиционирали мишом током манипулисања сценом.
* *ViewCube* – навигациони алат у виду коцке на дну која кориснику приказује из којег смера гледа сцену и како је та сцена позиционирана у односу на камеру.

Зарад лепшег приказа обе опције су искључене.

Унутар хеликса се дефинишу и елементи који сачињавају сцену односно његова „деца“. Једини елемент који дефинишемо кроз XAML биће глобално светло док ће остала бити додата кроз код.

Поред сцене, интерфејс садржи и дугмад за приказивање и сакривање елемената. Они су распоређени и груписани у 3 колоне.

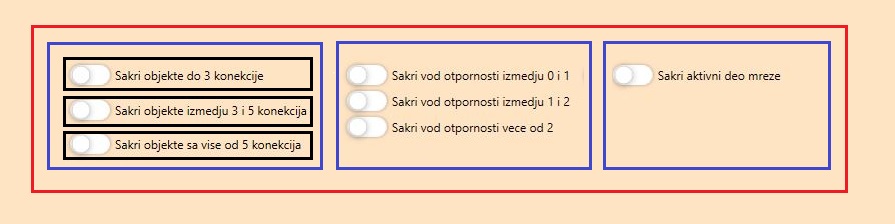


СЛИКА 1 – ДЕО ИНТЕРФЕЈСА ЗАДУЖЕН ЗА САКРИВАЊЕ И ПРИКАЗИВАЊЕ ЕЛЕМЕНАТА МРЕЖЕ

Колоне у које су распоређени дугмићи одговарају врсти сакривања и приказивања за која су намењена, тј. у првој колони се налазе дугмићи за видљивост елемената која зависи од броја конекција, у другој колони су дугмићи за видљивост водова у зависности од отпорности и у 3. колони се ради са приказом отвореног дела мреже.

Груписање дугмића се врши помоћу елемента „StackPanel“.

У интересу што једноставнијег и краћег објашњења структуре StackPanel-a, детаљан опис њихових међусобних положаја ће уместо из XAML кода бити објашњен следећом сликом.



СЛИКА 2 - МЕЂУСОБНИ ПОЛОЖАЈИ „STACKPANEL“-a

Црвеном бојом је означен први панел. Његова оријентација је хоризонтална што значи да ће елементи унутар њега бити поређани са лево на десно.

Његове елементе чине 3 панела означена плавом бојом. Они имају вертикалну оријентацију што значи да ће елементи унутар њих бити поређани од горе ка доле.

Њихове елементе чине панели хоризонталне оријентације означени црном бојом. Ови панели су сачињени од два елемента, „toggle“ дугмета и текста.

Како се елемент електроенергетске мреже може сакрити и приказивати погодно је користити „toggle“ дугме које се може наћи у два стања, чекираном и нечекираном. У нашем случају ће се активирањем, тј. „чекирањем“ дугмета одговарајући елементи сакрити док ће деактивирањем они поново бити видљиви.

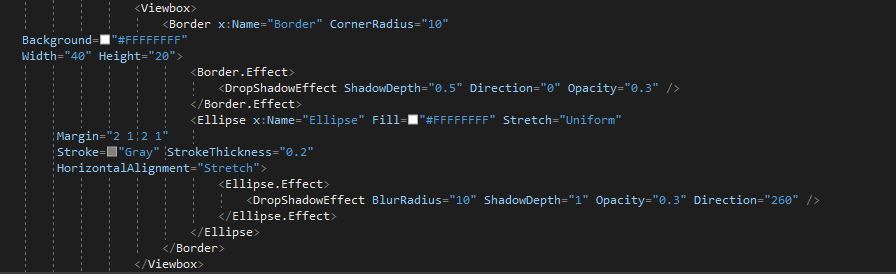
Како WPF нуди само основно „toggle“ дугме, побољшање корисничког искуства је постигнуто дефинисањем новог дугмета. Као и стандардно дугме, поред назива потребно је дефинисати и:

* *Checked* – метода која се позива сваким наредним активирањем дугмета.
* *Unchecked* – метода која се позива сваким наредним деактивирањем дугмета.



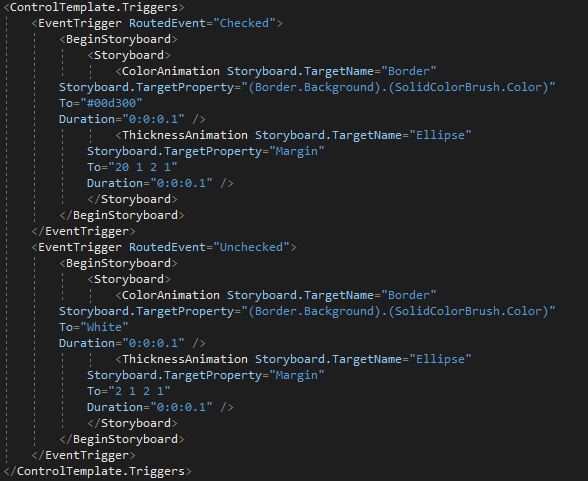
ЛИСТИНГ 2 - ОСНОВНИ Елементи „TOGGLE“ дугмета

Ново дугме ће бити састављено од заобљеног правоугаоника који у себи има круг. И правоугаоник и круг имају белу позадину и повећану провидност ивице. Ивица круга је сиве боје и има сенку чиме се лакше уочава разлика између два објекта.



ЛИСТИНГ 3 - Структура "TOGGLE" Дугмета

WPF омогућава анимацију дугмета директно из XAML кода. Дефинишу се две анимације, једна која се извршава приликом паљења а друга приликом гашења дугмета. Обе анимације трају десетинку секунде. Приликом паљења дугмета мења се боја позадине правоугаоника из беле у зелену а круг се помера ка десној ивици. Приликом гашења дугмета позадина правоугаоника се враћа на бело а круг на почетну позицију.



ЛИСТИНГ 4 - АНИМАЦИЈЕ "TOGGLE" ДУГМЕТА

Програм почиње иницијализацијом компоненти, након чега се подешава сцена. Кад подесимо сцену и додамо мапу, из фајла учитавамо елементе мреже. Након што учитамо све елементе одређујемо њихове позиције на мапи након чега их исцртамо. По завршетку цртања елемената, исцртавају се и водови који их спајају.



ЛИСТИНГ 5 - Главни Прозор

Метода за иницијализацију компоненти се генерише као део WPF пројекта и о њој неће бити речи.

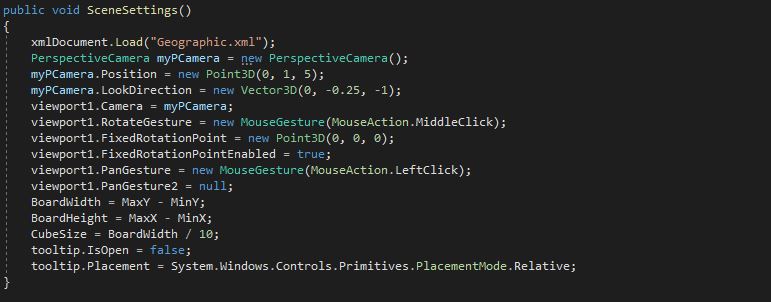
Како се пројекат ослања на податке који се чувају у фајлу са XML структуром погодно је користити већ постојеће технологије за њихово читање.



ЛИСТИНГ 6 - ПРОМЕНЉИВЕ ЗА РАД СА „XML“ ФАЈЛОМ

* *xmlNodeList* – променљива која ће се користити за читање одређеног типа чвора односно тага из XML фајла.
* *xmlDocument* – променљива која ће представљати XML документ.

SceneSettings() – метода која служи за иницијализацију и подешавање сцене.



ЛИСТИНГ 7 - МЕТОДА за ПОДЕШАВАЊе сцене

Метода започиње учитавањем фајла „Geographic.xml“ у променљиву xmlDocument.

Након учитавања фајла, креира се и подешава камера. Својства камере која се подешавају су њена позиција у 3Д сцени и вектор који представља смер гледања. Након подешавања камере она се додаје сцени.

Већ је било речи о HelixViewport3D-a али његова предност у односу на Viewport3D није била објашњена до сада. HelixViewport3D је намењен за готове моделе па има и уграђене методе за њихову манипулацију које се могу реализовати брзо и лако.

Манипулација моделом се реализује кроз класу „MouseGesture“ којој се као параметар задаје акција миша која је одговорна за гест.

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-u 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.

„Ширина“ и „висина“ плоче представљају разлику између екстремних вредности координата па ће се те вредности користити за проналазак позиције елемента на мапи, односно у 3Д координатном систему. О овоме ће бити речи касније.



ЛИСТИНГ 8 - ЕКСТРЕМНЕ КООРДИНАТЕ

Димензија коцке, која представља елемент мреже, је „ширина“ мапе подељено са 10. Овим се обезбеђује да се пропорција елемената у случају промене координата које плоча треба да обухвати не мења.

„toolTip“ је променљива класе „ToolTip“ која служи за исписивање информација о елементима. Његово стандардно стање је затворено а отвара се кликом на неки ентитет. Његова позиција је релативна, тј. зависи од позиције миша.

LoadBoard() *–* Метода која служи за додавање плоче односно мапе на сцену.

За дефинисање модела у 3Д свету је потребно креирати његову геометрију и материјал од којег ће он бити сачињен.

*Material* – Класа која омогућава креирање материјала модела.

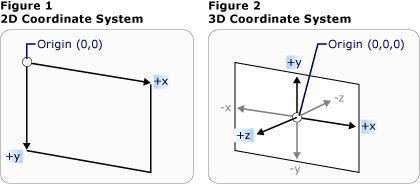
*MеshGeometry3D* – Класа која омогућава креирање геометрије модела.

Подешавање материјала се своди на дефинисање одређеног типа материјала који је обојен одређеном текстуром. У пројекту је коришћен дифузни тип материјал који не емитује светло. Текстура материјала је слика која представља мапу.



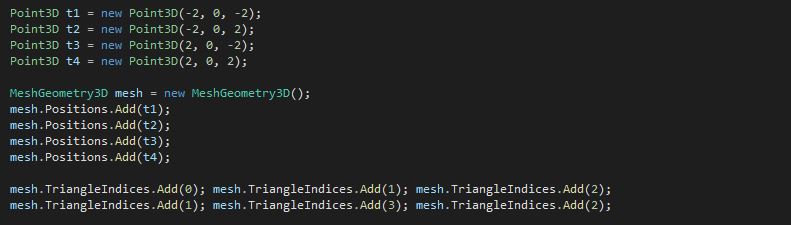
ЛИСТИНГ 9- креирање материјала за израду мапе

Приликом дефинисања геометрије 3Д модела користе се тродимензионалне тачке. Ове тачке су дефинисане са 3 координате. У 2Д координатном систему за почетак се узима горњи леви угао екрана. Од њега X координата расте ка десно а Y координата ка доле. У 3Д систему координатни почетак се налази у центру генерисаног простора. У односу на њега, X координата расте ка десно, Y расте ка горе а координата Z која представља дубину сцене расте ка смеру гледаоца.



СЛИКА 3 - 2Д и 3д координатни систем

Подешавање геометрије се своди на дефинисање тачака модела. Након што се дефинишу тачке њих је потребно повезати у троуглове. Разлог употребе троуглова у 3Д графици је њихова једноставност, они су геометријски објекат са најмање темена који дефинишу неку раван. Како је само једна страна троугла видљива редослед којим се темена додају у троугао је битан и прати правило десне руке. Ако палац одговара редоследу додавања тачака у троугао нормала троугла одговара правцу и смеру савијених прстију. Да би неки троугао био видљив његова нормала мора да иде према камери из које се тај троугао посматра.



ЛИСТИНГ 10 - Темена и троуглови мапе

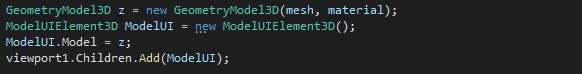
На листингу 10. се може видети креирање геометрије модела плоче. Како је у питању квадрат он се дефинише са 4 тачке које су изабрана тако да се центар плоче нађе у координатном почетку.

Само креирање 3Д тачака се своди на коришћење Point3D класе а као параметри се прослеђују вредности X, Y па Z координате.

Након што се дефинишу 3Д тачке оне се додају у геометрију модела и чувају у листи 3Д тачака „Positions”.

Дефинисање троуглова подразумева дефинисање његових темена преко „TriangleIndices“ својства. Ово својство представља листу темена троуглова. Да би нека 3Д тачка постала теме троугла она се додаје у ову листу а као параметар се прослеђује њена позиција унутар „Positions“ листе. Троуглови се сами генеришу тако да 3 узастопна темена у листи „TriangleIndices“ чине један троугао.

Модел плоче има 4 тродимензионалне тачке од којих је направљено 6 темена. Ових 6 темена генеришу 2 троугла који чине плочу облика квадрата.



ЛИСТИНГ 11 - Креирање и Приказивање Мапе

Након креирања материјала модела и дефинисања троуглова који описују његову геометрију креира се сам модел. Модел који се креира геометријом и материјалом је класе „GeometryModel3D“ па се такав не може додати сцени. Да би креирали модел који се може приказати кориснику креира се модел класе „ModelUIElement3D”. Кроз својство „Model” овај модел може да прими претходно креирани геометријски 3Д модел након чега се додаје сцени и приказује кориснику.

Изглед сцене након учитане мапе је приказан сликом 4:



СЛИКА 4 - Изглед сцене након учитавања мапе

LoadSubstations() - Метода која чита трафостанице из фајла.

Метода започиње читањем свих трафостаница из XML фајла. У ту сврху се користи већ поменута променљива „xmlNodeList“.



ЛИСТИНГ 12 - Читање свих трафостаница из фајла

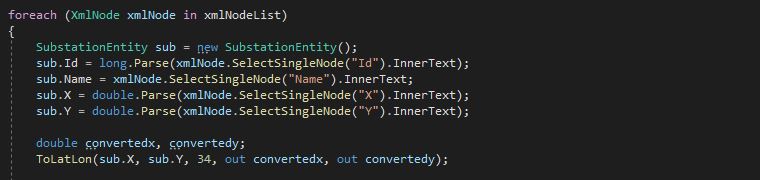
Листинг 12. показује читање свих инстанци неког тага из XML фајла. Вредност која је смештена у xmlNodeList има податке из фајла али структура није исправна.

SubstationEntity наслеђује класу PowerEntity следеће структуре:



ЛИСТИНГ 13 - структура "powerentity" класе

* *id* – Јединствени идентификатор ентитета.
* *name* – Назив ентитета.
* *x , y* – УТМ координате ентитета.
* *boardx , boardz* – Проширење класе пољима која представљају координате елемента унутар 3Д сцене. Како је наша сцена положена водоравно на дно координате које се траже на основу фајла су X и Z док ће координата Y бити јако мала, скоро 0, да би елементи „налегли“ на мапу. Y координате модела плоче су 0.



ЛИСТИНГ 14 - Итерација кроз "xmlNodelist" структуру

Након читања, пролази се кроз xmlNodeList и од сваког тага трафостанице креира објекат типа „SubstationEntity“.

За сваки чвор типа „SubstationEntity“ креира се објекат те класе. Објекат се попуњава информацијама на основу вредности унутар одређених тагова чвора који тренутно читамо. Након читања вредности унутар тага оне се парсирају у одређени тип податка и смештају у објекат.

* *SelectiSingleNode* – Метода којом читамо вредност тага којег смо проследили као параметар.
* *InnerText* – Опција којом добијамо само вредност смештену унутар тага не цео таг.

Када смо попунили објекат информацијама одређујемо његову географску дужину и ширину на основу УТМ координата из фајла. Ово се постиже коришћењем већ постојеће методе „ToLatLon” па о њој овде неће бити речи.

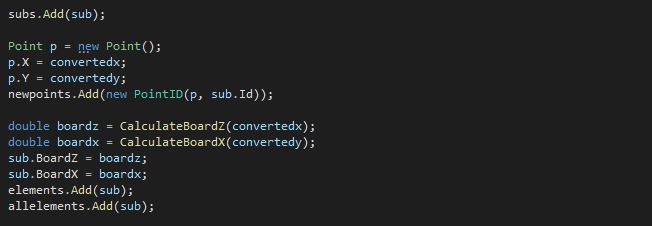
Када израчунамо географску ширину и дужину елемента врши се провера да ли се он налази унутар површи захваћене нашом мапом.



ЛИСТИНГ 15 - провера географске ширине и дужине трафостанице

Добијену географску ширину и дужину проверавамо са екстремним координатама. Уколико се географска ширина („convertedx“) налази унутар жељене површи врши се провера и за географску дужину („convertedy“).

Трафостанице за које провера утврди да се налазе на мапи се обрађују следећим кодом:



ЛИСТИНГ 16 - обрада трафостанице која задовољава критеријуме

* *subs* – Листа свих трафостаница.
* *newpoints* – Листа објеката типа PointID. Ови објекти садрже географске координате елемента и јединствени идентификатор елемента. Улога ове листе ће бити објашњена касније.
* *elements* – Листа у којој се чувају трафостанице, чворови и прекидачи.
* *allelements* – Листа у којој се осим горе поменутих чувају и водови.

Након што успоставимо да трафостаница задовољава координатне услове, додајемо је у листу свих трафостаница. Од њених географских координата и идентификатора креирамо објекат типа „PointID“ и додамо га у листу „newpoints“.

На основу географске ширине се рачуна Z координата методом „CalculateBoardZ“.

На основу географске дужине се рачуна X координата методом „CalculateBoardX“.

Након што израчунамо координате трафостанице у 3Д сцени додајемо их у поља којим смо проширили ову класу.

Разлог због којег додајемо трафостанице, чворове и прекидаче који задовољавају услов у једну листу је лакша провера валидности водова. Како вод може да спаја елементе од којих неки не задовољава услов потребно је проверити оба елемента које вод спаја. Када су сви валидни елементи у једној листи њихов проналазак је лакши и бржи а самим тим и провера валидности вода.

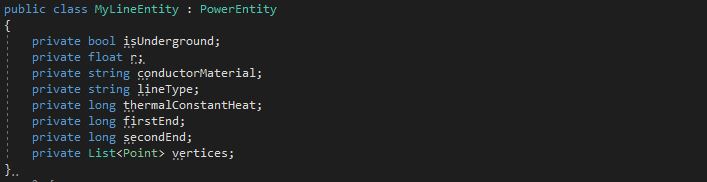
LoadNodes() – Метода која служи за ишчитавање чворова електроенергетске мреже из фајла.

LoadSwitches() – Метода која служи за ишчитавање прекидача електроенергетске мреже из фајла.

Ове методе су својом структуром идентичне методи која чита трафостанице па оне неће бити детаљно објашњене. Једина разлика је у ентитету који се чита односно обрађује. Важно је напоменути да је класа „SwitchEntity“ проширена атрибутом „Status“ који одређује да ли је прекидач отворен или затворен.

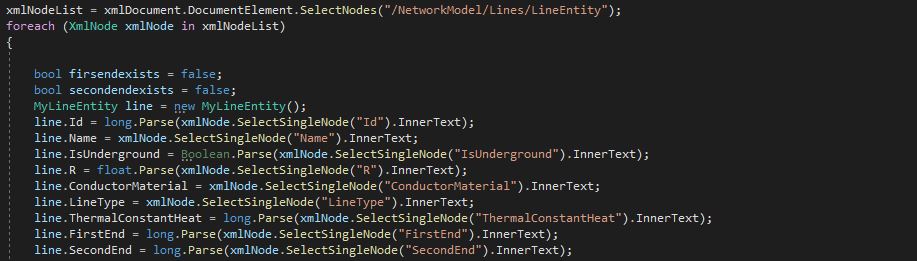
LoadLines() – Метода која је задужена за ишчитавање и обраду водова.

Структура класе која описује вод је приказана на следећем листингу:



ЛИСТИНГ 17 - структура класе која описује вод

На горе приказаном листингу треба приметити да се користи класа „MyLineEntity“ за опис вода. Ова класа је структурално идентична датој класи „LineEntity“ али, за разлику од ње, наслеђује класу „PowerEntity“. Разлог овоме је што се сад сви ентитети мреже могу чувати у једну листу чиме ће знатно бити олакшана имплементација хит тестирања. О овоме ће бити речи касније.

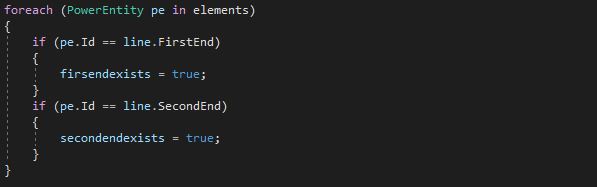


ЛИСТИНГ 18 - Oбрада водова

Као и претходне 3 методе, ова метода започиње читањем свих тагова типа „LineEntity“ из фајла. Креира се нови вод типа „MyLineEntity“ и попуњава прочитаним подацима.

* *R* – Отпорност вода.
* *ConductorMaterial* – Материјал од којег је сачињен вод.
* *firstend* – Идентификатор првог ентитета којег вод спаја.
* *secondend* – Идентификатор другог ентитета којег вод спаја.
* *firstendexists* – Променљива која нам говори да ли је први крај вода задовољио услов, односно да ли се налази у захваћеној површини.
* *secondendexists* – Променљива која нам говори да ли је други крај вода задовољио услов, односно да ли се налази у захваћеној површини.

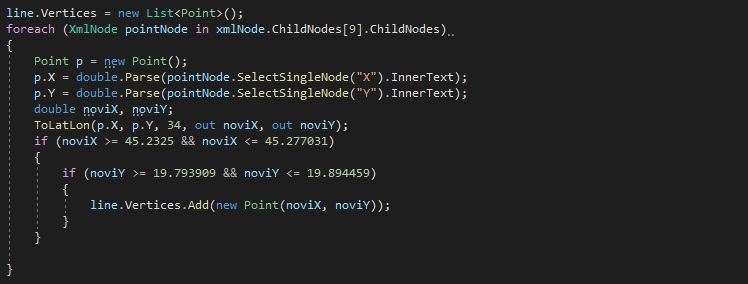
Након што креирамо објекат и попунимо га прочитаним информацијама врши се провера његове валидности. Вод је валидан ако су оба његова краја валидна, односно ако се они налазе у листи ентитета који задовољавају услов.[[1]](#footnote-1)



ЛИСТИНГ 19 - Провера валидности вода

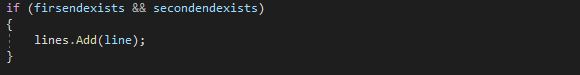
Провера вода подразумева пролазак кроз листу и проналазак првог и другог краја вода. Како се у листи налазе само валидни елементи уколико се пронађу оба краја то значи да је и сам вод валидан.

Како се вод исцртава преко својства „Vertices“ које представља низ географских локација кроз које вод пролази, потребно их је прочитати из фајла и испитати њихову валидност. Следећи листинг показује управо то:



ЛИСТИНГ 20 - ОБрада "Vertices" координата

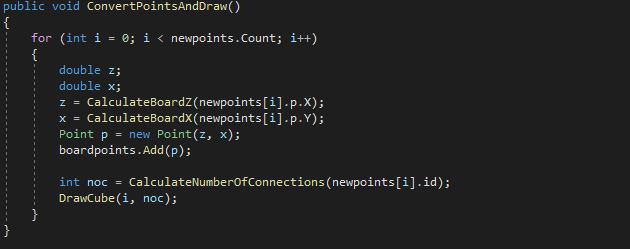
УТМ координате у фајлу се чувају као подтагови унутар деветог тага самог ентитета вода па се њима приступа преко „ChildNodes“ својства. УТМ координате се на већ описан начин претварају у географску ширину и дужину након чега се врши стандардна провера. Само онe локације које задовољавају услов се обрађују док ће остале бити занемарене.



ЛИСТИНГ 21 - додавање вода у листу водова

Кад у вод учитамо све локације кроз које пролази, уколико је исправан додаје се у листу свих водова што је приказано на листингу 21.

ConvertPointsAndDraw() – Метода која конвертује географске координате трафостаница, чворова и прекидача у координате 3Д система и црта елементе на мапу. Њен изглед је приказан следећим листингом:

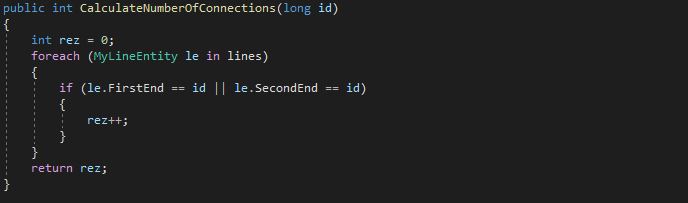


ЛИСТИНГ 22 - метода за конверзију и цртање елемената

* *newpoints -* Раније је било речи о овој листи а она ће овде бити детаљније објашњена. Листа садржи географске позиције и идентификаторе трафостаница, чворова и прекидача. Разлог чувања координата је очигледан док се идентификатор чува ради утврђивања броја конекција елемента.
* *boardpoints* – Листа 3Д координата на које ће се цртати елементи.

Метода пролази кроз све елементе, рачуна њихову позицију унутар 3Д сцене на основу њихових географских координата и смешта их у листу „boardpoints“. Ова листа ће бити од значаја приликом цртања елемента на мапу.

Рачунање броја конекција неког елемента се своди на пролазак кроз све водове и утврђивање броја водова код којих је идентификатор неког краја вода једнак идентификатору елемента за који се тражи број конекција. Изглед ове методе је приказан на следећем листингу:



ЛИСТИНГ 23 - Метода за проналазак броја конекција неког елемента

Када се утврди број конекција елемента може се започети са његовим цртањем на мапи. За цртање елемената задужена је метода „DrawCube“.

Разлог због којег смо прво учитали све елементе мреже, пронашли њихове координате у 3Д простору па их тек онда цртали је тај што нам је тако омогућено да за сваки елемент који цртамо лакше испитамо да ли већ постоји неки који се налази на тој позицији, тј. да ли је елемент потребно подићи и за колико.



ЛИСТИНГ 24 - МЕтода која црта елемент на мапу

* *elevation* – Вредност која представља размак између вертикално поређаних елемената. Износи 0.006.
* *exists* – Вредност која нам говори колико елемената се налази на позицији на којој се налази елемент који цртамо.
* *newz* – Z координата елемента којег исцртавамо.
* *newx* – X координата елемента којег исцртавамо.

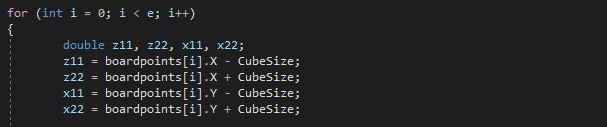
Метода „DrawCube“ прима два параметра. Први је редни број елемента који се црта а други је број конекција тог елемента.

На почетку методе претпоставка је да нећемо наћи ни један елемент који се налази у непосредном окружењу елемента којег цртамо. Због тога је број елемената у његовој близини постављен на 0 као и елевација елемента. Координате елемента преузимамо из „boardpoints“ листе а како елементи те листе одговарају елементима у листи „newpoints“ може се користити редни број елемента који је прослеђен као параметар методи.

Након што преузмемо 3Д координате елемента, пролазимо кроз све елементе који су већ нацртани, односно кроз све елементе пре елемента којег цртамо. За сваки од тих елемената преузимамо 3Д координате. Како 3Д координате описују центар елемента користимо величину коцке да добијемо 4 координате.

Добијене координате, од којих су 2 Z а 2 X, описују 2Д простор на мапи унутар којег не сме да се нађе центар новог модела уколико желимо да избегнемо преклапање.

Ово је приказано на листингу испод:



ЛИСТИНГ 25 - Одређивање површи на којој се не сме наћи центар новог модела да би се избегло преклапање

Како коцка има дужину странице „CubeSize“, растојање између центра и странице је између и .

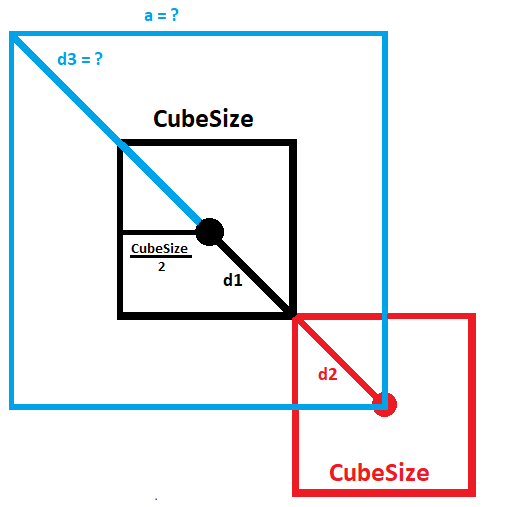
Уколико изаберемо прву вредност за параметар на основу којег ће се формирати забрањена површина може се десити да се елемент чији се центар не налази унутар те површине ипак преклапа са већ постојећим елементом па ово решење није адекватно.

Одабиром друге вредности као параметра добија се површина која обезбеђује да елемент, чији је центар ван ње, нема преклопних тачака са постојећим елементом. Прецизност оваквог решења није потпуна јер може да се деси да неки елемент, чији је центар унутар површи, нема преклопних тачака са постојећим елементом. Једноставна имплементација оваквог решења оправдава његову, занемарљиво мању, прецизност.

Како смо као параметар узели половину дијагонале, односно , удаљеност центра новог елемента од центра постојећег је два пута већа од ове вредности у случају кад се елементи додирују ћошковима. Разлог због којег је растојање 2 пута веће је јер прва половина дијагонале одговара првом елементу а друга половина другоме чиме се добије растојање .

Како је ово удаљеност по једном смеру правца дијагонале кад се та удаљеност дода и на други смер добија се површ чија је дијагонала . Ово нам говори да је дужина странице забрањене површине . Како постојећи модел и забрањена површ имају исти центар, координате површи се добијају додавањем вредности на оба смера обе координате што се види и на листингу 25.

СЛИКА 5 - Графички приказ преклапања коцки из птичије перспективе

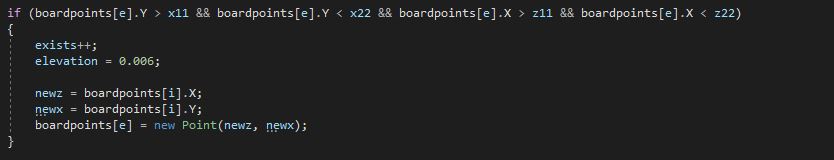


Црна коцка – постојећи елемент.

Црвена коцка – нови елемент.

Плава коцка – забрањена површ.

Након што креирамо забрањену површ, на основу координата елемента проверавамо да ли се он налази унутар те површине. Елемент се налази унутар забрањене површи уколико се Z координата његовог центра налази унутар Z координата површи и уколико се Y координата његовог центра налази унутар Y координата површи.



ЛИСТИНГ 26 - провера преклапања елемената

На горе приказаном листингу вреди напоменути да се у листи „boardpoints“ X координата 3Д система елемента чува у „Y“ својству док се Z координата чува у „X“ својству.

Уколико утврдимо да се нови елемент налази унутар забрањене површи неког постојећег елемента, повећавамо број елемената који се налазе на тој позицији и мењамо вредности променљиве задужене за подизање елемента у вис.

Након што променимо ове две вредности мењају се и 3Д координате новог елемента, тј. његове 3Д координате постају координате елемента изнад којег се нацртао. Овим се постиже поравнање елемената и поједностављује свака наредна провера. Након што променимо координате унутар методе, оне се мењају и у листи.

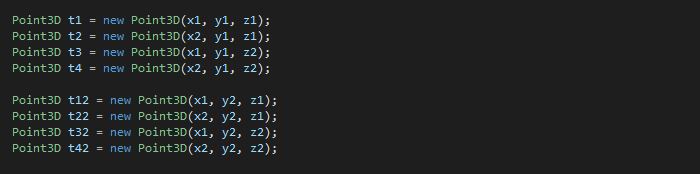
Када су сви постојећи елементи обрађени рачуна се позиција коцке на 2Д мапи односно координате темена основе коцке. На координате центра коцке додаје се и одузима како би дужина странице била CubeSize. Након израчунатих X и Z координата, на основу „exists“ и „elevation“ рачунају се Y координате доње односно горње странице.

Y координата темена горње странице ће бити подигнута за дужину странице коцке у односу на основу коцке.



ЛИСТИНГ 27 - Рачунање координата коцке

Након што израчунамо све координате креирамо 3Д тачке које описују модел.

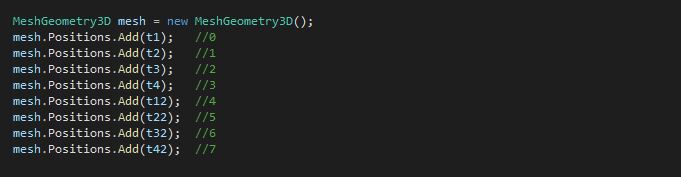


ЛИСТИНГ 28 - 3Д тачке које описују коцку

Тачке t1, t2, t3 и t4 су темена основе коцке, а тачке t12, t22, t32 и t42 су темена горње странице коцке. Као што се примети на листингу 28. темена горње странице су идентична својим парњацима са доње стране али су подигнути по Y координати.

Када креирамо тачке које описују модел креирамо геометрију модела. Као што је већ објашњено[[2]](#footnote-2), геометрија модела се састоји из темена модела и троуглова који их спајају.

Креирање темена коцке од тачака које је описују:

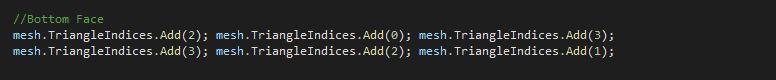


ЛИСТИНГ 29 - Темена коцке

* *mesh* – Променљива која описује геометрију модела.

Кад додамо ове тачке у сам модел коцке одређују се и троуглови који описују коцку. Како је коцка описана са 6 страница квадратног облика, а сваки квадрат описан са 2 троугла, потребно је израчунати 12 троуглова који описују модел у потпуности.

На листингу испод се види пример креирање 2 троугла који описују основу коцке:



ЛИСТИНГ 30 - Троугао који описују основу коцке

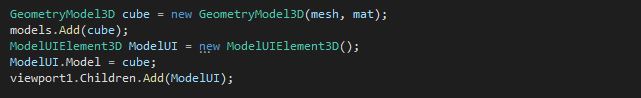
Остале странице се описују на исти начин па њихови троуглови неће бити приказани.

Као и приликом креирања мапе[[3]](#footnote-3), модел креирамо од његове геометрије и материјала. Како је геометрија описана, морамо креирати материјал. Материјал који користимо је дифузни а обојен је црвенкастом бојом:



ЛИСТИНГ 31 - Креирање Материјала

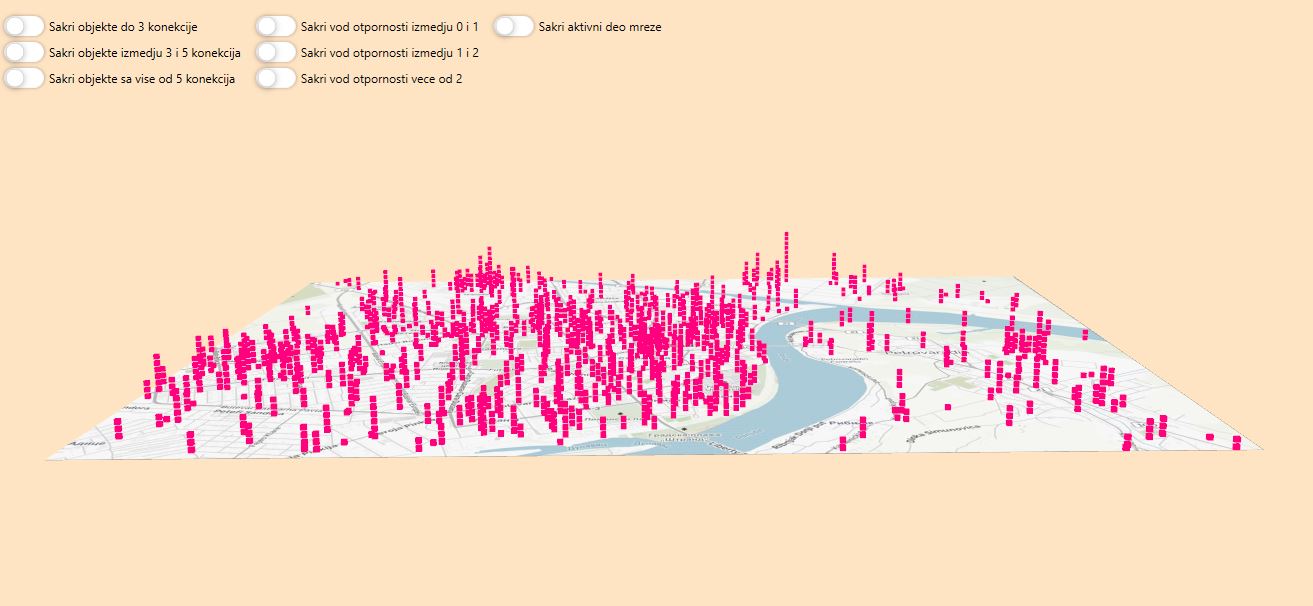
Након креирања материјала, можемо креирати и нацртати модел. Пре цртања модела он се додаје у листу свих модела. Већ је било речи о формирању модела и његовом додавању на сцену[[4]](#footnote-4) па ће овде детаљно објашњење изостати.



ЛИСТИНГ 32 - ФОрмирање и цртање коцке

* *models* – Листа свих модела. Ова листа ће се користити приликом хит тестирања.

Након цртања свих трафостаница, чворова и прекидача добија се следећи изглед сцене:



СЛИКА 6 - ИЗГЛЕД СЦЕНЕ НАКОН ЦРТАЊА ТРАФОСТАНИЦА, ЧВОРОВА и ПРЕКИДАЧА

Након што нацртамо све елементе мреже, потребно је нацртати и водове које их спајају.

DrawLines() – Метода која је задужена за цртање водова.

Како „Verices“ координате не одговарају елементима мреже оне неће бити исцртане на мапи. То нам представља проблем приликом цртања пута јер наш пут не спаја коцке већ спаја странице одређених коцки. Уколико не будемо правили разлику између самих координата елемената и „Vertices“ координата, наш вод ће имати рупе, облика коцке, чији ће центри бити „Vertices“ координате кроз које вод пролази.

Овај проблем је решен тако што се методи „DrawLine“, осим координата, прослеђује и параметар који говори на којем крају пута се налази елемент.

Уколико је један крај пута елемент, метода ће спојити вод до његове ивице, тако да неће доћи до преклапања вода и елемента.

Уколико је један крај пута „Vertices“ локација, метода ће спојити вод до његовог центра, тако да неће доћи до формирања рупа.

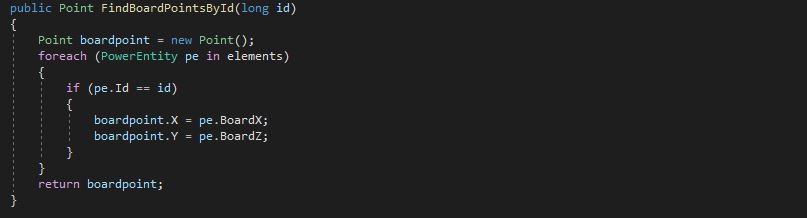
Метода започиње итерацијом кроз све учитане водове. За сваки вод се проверава да ли на свом путу од почетног до крајњег елемента пролази кроз неке додатне локације, односно да ли је његова „Vertices“ листа празна.

Водови који немају „Vertices“ , спајају директно крајње елементе:



ЛИСТИНГ 33 - Вод који директно спаја крајње елементе

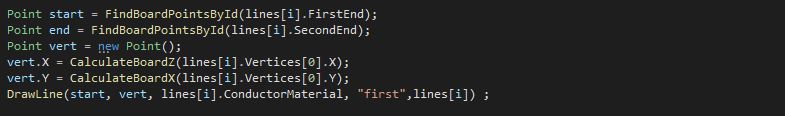
За проналазак координата крајњих елемената вода се користи метода „FindBoardPointsById“ која као параметар прима идентификатор елемента. Ова метода пролази кроз све елементе и проверава њихов ИД. Уколико се пронађе идентификатор чији је ИД идентичан прослеђеном, преузимају се његове координате у 3Д простору и враћају као резултат претраге.



ЛИСТИНГ 34 – FindBoardpointsbyid

Када се пронађу 3Д координате које треба да се повежу позива се метода „DrawLine“ која је задужена за цртање пута између две локације. Овој методи се прослеђује параметар „both“ (оба) тако да метода „зна“ да су оба краја које вод спаја елементи електроенергетксе мреже.

Водови који имају „Vertices“ својство се цртају тако да се прво повеже почетни елемент са првом локацијом из својства „Vertices“.

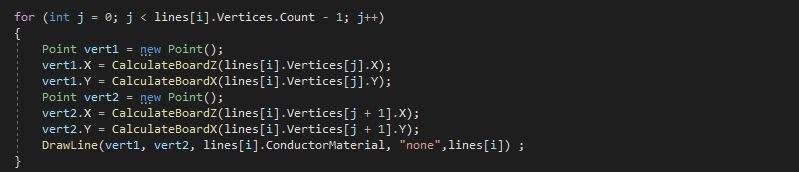


ЛИСТИНГ 35 - спајање почетка пута са првом ликацијом унутар "Vertices" листе

* *start* – 3Д координата почетка вода.
* *end* – 3Д координата краја вода.
* *vert* – 3Д координата прве локације унутар „Vertices“ листе.

Када одредимо 3Д координате прве локације кроз коју вод пролази на свом путу, спајамо почетак вода са њом. Како се елемент мреже налази на првом крају овог дела пута а на другом „Vertices“ локација као параметар методи за цртање се прослеђује „first“ (први).

Када повежемо први крај вода са првом локацијом унутар „Vertices“ листе, потребно је проћи кроз листу и повезати међусобно локације унутар ове листе.

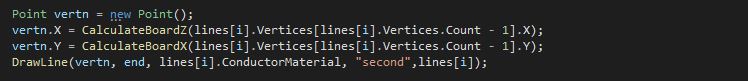


ЛИСТИНГ 36 – Међусобно повезивање „Vertices“ локација

* *j* – Позиција локације унутар „Vertices“ листе.
* *i* – Редни број вода који се обрађује.
* *vert1* – Прва „Vertices“ локација.
* *vert2* – Друга „Vertices“ локација.

Разлог због којег итерација иде од почетне локације до претпоследње је тај што ће се претпоследња локација спојити са последњом а последња ће се спојити са крајњим елементом вода. Овде метода за цртање дела вода прима параметар „none“(ниједан), јер су крајеви ових делова вода сви из листе „Vertices“ тј. ни један ни други крај нису елемент електроенергетске мреже.

Када повежемо све „Vertices“ координате из листе међусобно, последњу координату из листе повезујемо са крајњим чвором вода. Метода за цртање пута прима параметар „second“ (други) јер је први крај пута координата из листе „Vertices“ а други крај пута је и крај самог вода па је самим тим и елемент мреже.



ЛИСТИНГ 37 - ПОвезивање последње "Vertices" координате са крајњим чвором вода

* *vertn* - Последња координата из „Veritces“ листе.

*DrawLine()* – Метода која је задужена за повезивање две 3Д локације на мапи.

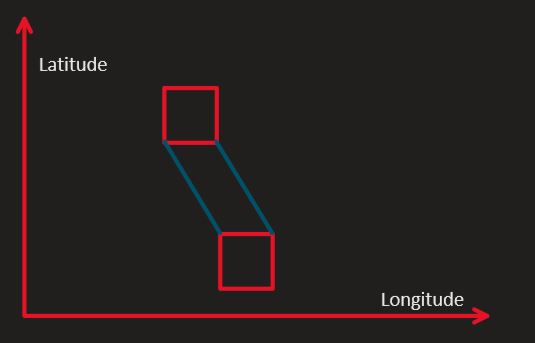
Осим локација које повезује, она као параметар прима и материјал вода од којег зависи његова боја, позицију елемента мреже која је раније описана и податке о воду.

Како се наш вод састоји из делова, приликом клика на било који од ових делова потребно је исписати исте информације о воду и обојити његове крајње елементе. Да би знали којем воду припада део на који смо кликнули и које елементе цео вод спаја (а не локације повезане делом на који смо кликнули) као пети параметар приликом цртања пута прослеђује се вод којем овај пут припада.

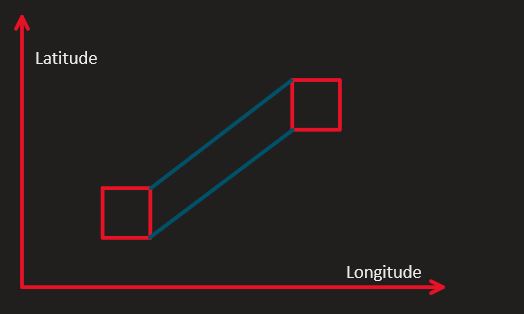
Приликом спајања елемената на мапи разликујемо 2 случаја:

1. Елементи се спајају по географској дужини.
2. Елементи се спајају по географској ширини.

У зависности од координата елемената бира се начин на који ће елементи бити повезани. Уколико је удаљеност елемената по географској дужини већа или једнака удаљености елемената по географској ширини бира се први начин, у супротном се елементи повезују на начин број 2. На сликама испод се види графички приказ повезивања елемената на 2Д мапи из птичије перспективе.



СЛИКА 8 - Други начин повезивања елемената



СЛИКА 7 – Први начин повезивања елемената

1. Детаљан опис ове листе је на страни 12. [↑](#footnote-ref-1)
2. Детаљније објашњење на страни 9. [↑](#footnote-ref-2)
3. Страна 8. [↑](#footnote-ref-3)
4. Страна 9. [↑](#footnote-ref-4)