Miskolci Egyetem
Gépészmérnöki és Informatikai Kar
Alkalmazott Matematikai Intézeti Tanszék

Szakdolgozat

Miskolci Egyetem

Közlekedés modellezése WebGL-ben a Unity Engine segítségével

Készítette:

Kása Dániel Zoltán

Programtervező informatikus BSc

Témavezető:

Piller Imre, egyetemi tanársegéd

Miskolc, 2019

Miskolci Egyetem
Gépészmérnöki és Informatikai Kar
Alkalmazott Matematikai Intézeti Tanszék **Szám:**

Szakdolgozat Feladat

Kása Dániel Zoltán (KYDK4Z) programtervező informatikus.

A szakdolgozat tárgyköre: Szabály alapú rendszerek, számítógépi grafika, szimuláció

A szakdolgozat címe: Közlekedés modellezése WebGL-ben a Unity Engine segítségével

A feladat részletezése:

...

Témavezető: Piller Imre (egyetemi tanársegéd)

A feladat kiadásának ideje: 2018. 10.

.....
szakfelelõs

Eredetiségi Nyilatkozat

Alulírott **Kása Dániel Zoltán**; Neptun-kód: KYDK4Z a Miskolci Egyetem Gépészmérnöki és Informatikai Karának végzős programtervező informatikus szakos hallgatója ezennel büntetõjogi és fegyelmi felelősségem tudatában nyilatkozom és aláírásommal igazolom, hogy *Közlekedés modellezése WebGL-ben a Unity Engine segítségével* című szakdolgozatom saját, önálló munkám; az abban hivatkozott szakirodalom felhasználása a forráskezelés szabályai szerint történt.

Tudomásul veszem, hogy szakdolgozat esetén plágiumnak számít:

- szószerinti idézet közlése idézőjel és hivatkozás megjelölése nélkül;
- tartalmi idézet hivatkozás megjelölése nélkül;
- más publikált gondolatainak saját gondolatként való feltüntetése.

Alulírott kijelentem, hogy a plágium fogalmát megismertem, és tudomásul veszem, hogy plágium esetén szakdolgozatom visszautasításra kerül.

Miskolc, ... év ..... hó ..... nap

.....

Hallgató

1.

szükséges (módosítás külön lapon)

A szakdolgozat feladat módosítása

nem szükséges

....

dátum témavezető(k)

2. A feladat kidolgozását ellenőriztem:

témavezető (dátum, aláírás): konzulens (dátum, aláírás):

....

....

....

3. A szakdolgozat beadható:

....

dátum témavezető(k)

4. A szakdolgozat ..... szövegoldalt

..... program protokollt (listát, felhasználói leírást)

..... elektronikus adathordozót (részletezve)

.....

..... egyéb mellékletet (részletezve)

.....

tartalmaz.

....

dátum témavezető(k)

5.

bocsátható

A szakdolgozat bírálatra

nem bocsátható

A bíráló neve: .....

....

dátum szakfelelős

6. A szakdolgozat osztályzata

a témavezető javaslata: .....

a bíráló javaslata: .....

a szakdolgozat végleges eredménye: .....

Miskolc, ...

a Záróvizsga Bizottság Elnöke

Tartalomjegyzék

1. fejezet

Bevezetés

2. fejezet

Témaköri áttekintés

2.1. Hasonló célú szoftverek

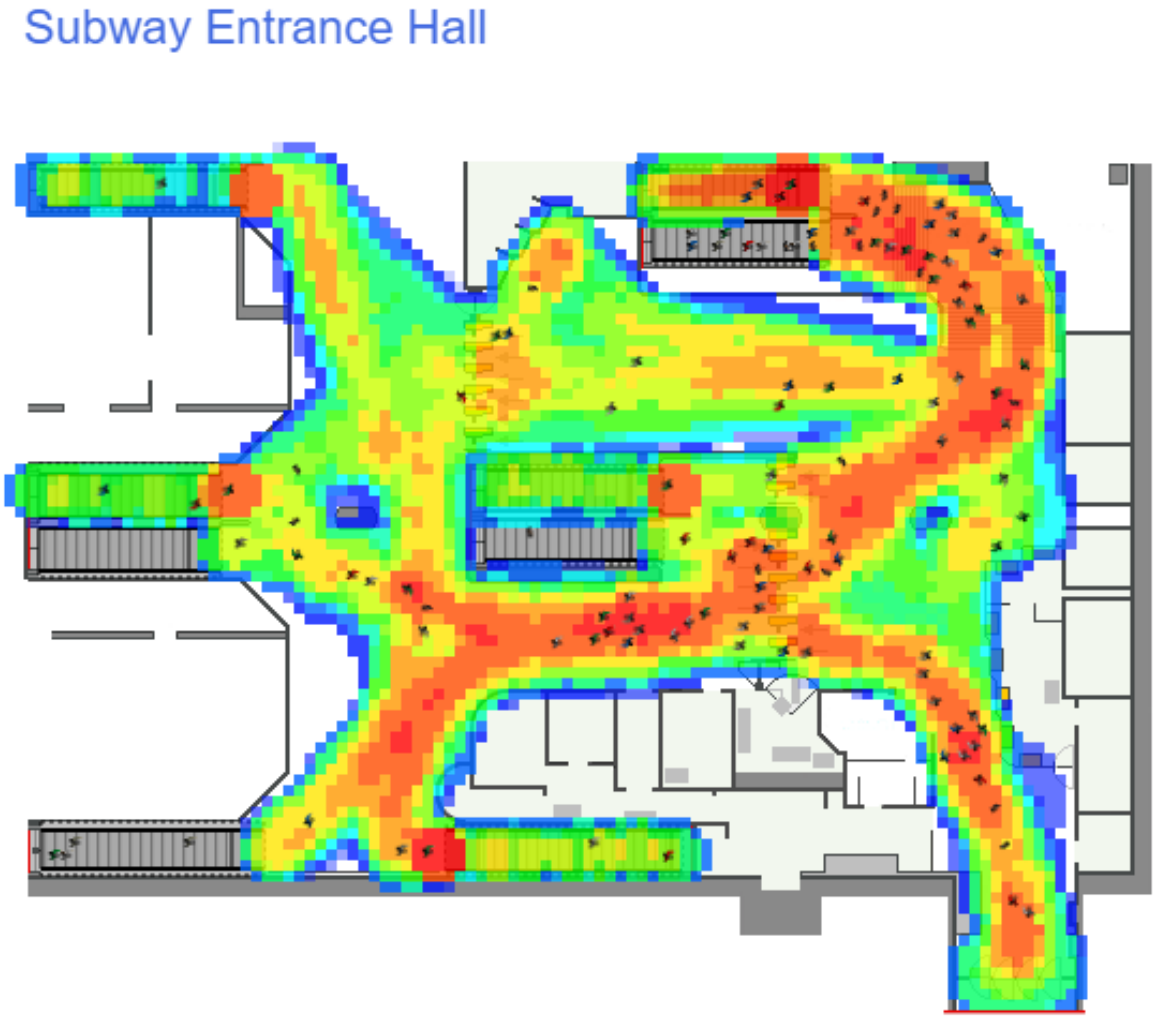
A feladatot azzal kezdtem, hogy utánanéztem milyen közlekedés modellezésére, annak szimulálására készült szoftverek készültek eddig. Ezek felépítése, valamint működése alapján fogom összeállítani a modellt, és továbbá az alapvető elvárásokat a generálást és a szimulációkat illetően.

2.1.1. Road Traffic Library

Az eddigi közlekedés-szimuláló szoftverek közül kiemelném először az AnyLogic által készített Road Traffic Library-t. Ezen szoftver segítségével létre lehet hozni egy úthálózatot különböző elemekből, mint például egyenes útszakasz, útkanyarulat, útkereszteződés, híd, gyalogátkelőhely, lámpás útkereszteződés, valamint buszmegállók és parkolók. A szoftver képes különböző közlekedési szabályok alkalmazására, valamint a járművek intelligens módon választják meg a haladáshoz szükséges útvonalat. Valósidejű szimuláció futtatására is van lehetőség, 2D-ben és 3D-ben is.

2.1. ábra. Kép a Road Traffic Library-ből

A szimulációkat felhasználva hőtérkép is előállítható, amely megmutatja mennyire voltak forgalmasak az adott területek, így egyértelművé válik hol alakul ki könnyedén forgalmi dugó.



2.2. ábra. Hőtérkép egy metróállomás gyalogosforgalmáról az AnyLogic szimulációs szoftverrel

Lehetőség van a geoinformációs rendszerekben használt vektorgrafikus formátum, azaz shapefile importálására is. Ezen fájl alapján a szoftver automatikusan előállítja az úthálózatot. A szoftver bővíthető még gyalogos- és vasútforgalomra vonatkozó könyvtárakkal is.

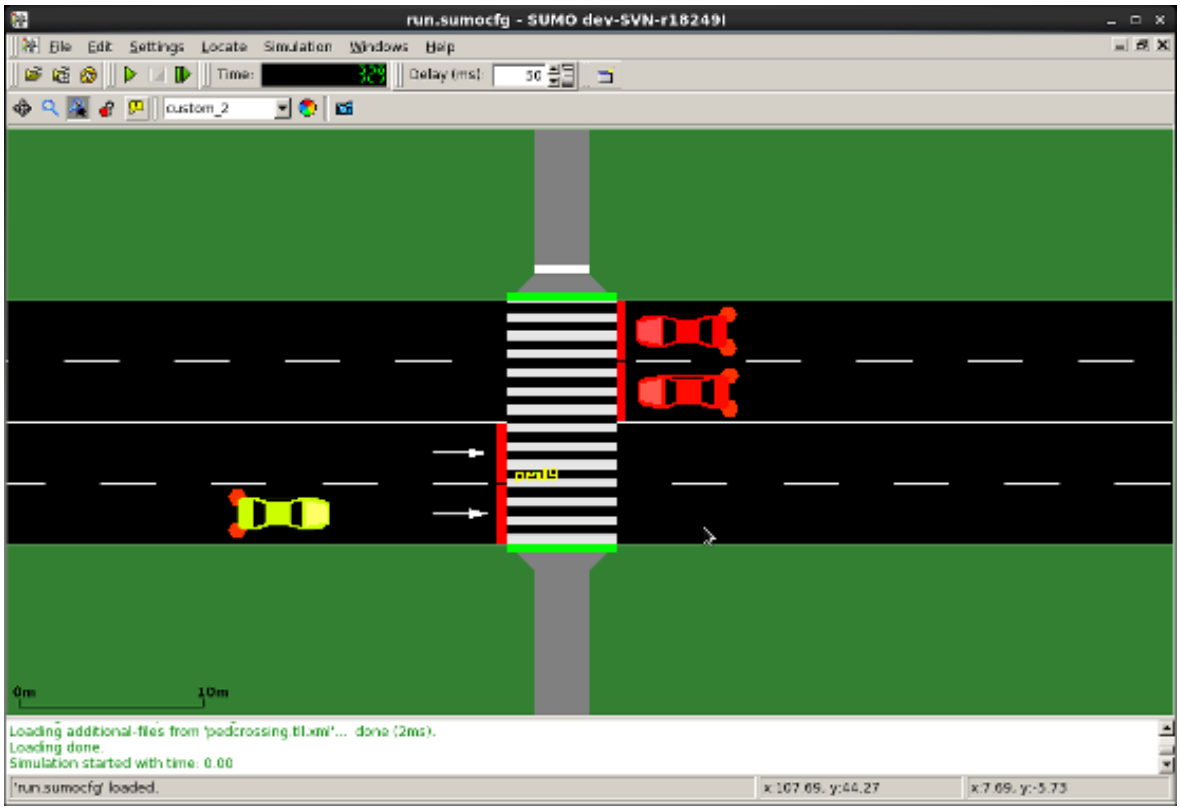
A gyalogos könyvtár segítségével szimulálható a gyalogosforgalom sűrűsége különböző környezetekben. Ilyen környezet lehet például egy buszmegálló, vagy vasútállomás területe. Az egyes épületek felépítése is befolyásolja a forgalom sűrűségét, a gyalogosok minden akadályt, ebbe beleértve egymást is, próbálnak elkerülni.

A vasút könyvtár segítségével egy pályaudvar vasútforgalma komplex módon szimulálható. A forgalmat befolyásolja a tehervonatok rakodási ideje, a karbantartás, üzleti folyamatok lefolyása, és sok más további esemény is.

2.1.2. SUMO

A SUMO, azaz Simulation of Urban MObility egy ingyenes, nyílt, közlekedés-szimuláló C++-ban írt szoftver. Az első verzió 2001-ben került kiadásra, azóta már az 1.1.0-ás

verziószámnál jár. Ez a szoftver lehetőséget ad több közlekedési forma (tömegközlekedés, gyalogosok, stb) modelezzésére a személygépjárművek mellett. Továbbá nagy eszköztárral bír az olyanokhoz mint útvonaltervezés, káros-anyag kibocsátás, valamint vizuális megjelenítés. Lehetséges továbbá meglévő úthálózat importálása fájlból, valamint személyre szabott modellek használata. A SUMO nem tartalmaz semmilyen megkötést az úthálózat méretét, vagy az egyszerre szimulálható járművek számát illetően.



2.3. ábra. Kép a SUMO programból

A program lehetőséget ad a szimuláció online történő kezelésére is, a TraCI, azaz Traffic Control Interface használatával. Ilyen esetben a SUMO tulajdonképpen egy szerver, amelyet egy vagy több kliensen keresztül tudunk vezérelni. TCP alapú kliens/-szerver architektúrát használ.

A közlekedési lámpák viselkedését egyénileg be lehet állítani, vagy akár előre megírt TLS programból is lehetséges importálni. Az importáláshoz számos python szkript áll rendelkezésre. Ha nem áll rendelkezésre TLS program, a SUMO képes automatikusan generálni egyet. Ilyenkor alapértelmezett értékeket rendel minden paraméterhez amelynek értéke nem volt kapcsolókkal definiálva. Ezek közé tartoznak:

- –tls.cycle.time - lámpák váltási ideje.
- –tls.yellow.time - mennyi időre sárga a lámpa, ez alapértelmezetten az utak sebességhatára alapján történik kiszámításra.
- –tls.minor-left.max-speed - az útkereszteződésben balra kanyarodást megengedő

sebességhatár (ha az út sebességhatára ezen érték felé esik, nem engedélyezett a balra kanyarodás.)

- –tls.left-green.time - ha egyszerre kap zöldet az egyenesen haladó, és a vele szemből balra kanyarodó, ezen érték idejéig a balra kanyarodó kap elsőbbséget.
- –tls.allred.time - minden zöld lámpa előtt ezen értéknek megfelelő ideig piros az összes lámpa (alapértelmezetten 0.)
- –tls.default-type - actuated értékre állítva változó hosszúságú ideig tartanak a zöld lámpák.

A szoftver alkalmas a közlekedési lámpák teljesítményének vizsgálatára, olyan útvonaltervezésre amely igyekszik a káros-anyag kibocsátást és a légszennyezést minimalizálni. A gyakorlatban történt híresebb alkalmazásai tartozik például a 2006-os labdarúgó világbajnokság-, valamint a Pápa 2005-ös Kölni látogatása során a közlekedés előrejelzése.

Maga a programcsomag tartalmazza a parancssori SUMO szoftvert, a grafikus felülettel ellátott GUISIM-et, egy úthálózat importáló NETCONVERT szoftvert, úthálózat generáló NETGEN szoftvert, egy OD2TRIPS nevű alkalmazást amely O/D mátrix alapján generál útvonalat, számos útvonal generátort, valamint egy grafikus úthálózat-szerkesztő alkalmazást.

2.2. Meglévő térkép-adatbázisok modellje

A Google Maps az egyik legelterjedtebb úthálózat-vizualizálásra alkalmas szoftver. Ez egy JavaScript alapú webalkalmazás, mely a geoinformációs adatok megjelenítése mellett valós-idejű forgalom elemzésre is képes.

Ehhez az adatokat több féle forrásból állítják elő. A Base Map program keretében rengeteg különböző forrásból szereztek vektoradatokat a létező úthálózatokról. Később ennek a nevét Geo Data Upload-ra váltották, viszont a lényege ugyan az maradt. Továbbá szereznek adatokat műholdképekből, Android rendszerű okostelefonok szolgáltatásai által, valamint a nemrég bevezetett "Helyi Idegenvezetők" funkcióval, amin keresztül bárki tölthet fel adatokat.

A valós-idejű forgalomelemzéshez kezdetekben a helyi önkormányzatok által szolgáltatott adatokat használták, melyeket az egyes helyeken felszerelt szenzorok segítségével gyűjtöttek be. Viszont ezek csak a legforgalmasabb utakról adtak információt, ezért később áttértek a "crowd-source" módszerhez. Ezzel a módszerrel egy Google Maps-et futtató okostelefon GPS adatait felhasználva ad becslést egy út forgalmasságára. Lényegében azt elemzi, mennyien küldtek GPS adatot egy adott útszakaszon azonos időben.

Az utak és más objektumok kirajzolásához a böngésző leegyszerűsített vektoradatokat kap, melyek alapján felépíti az azoknak megfelelő formákat. Minden egyes objektumhoz tartoznak különböző metaadatok, mint például a sebességkorlát, haladási irány, út típusa (főút, stb), valamint egy név. Az objektumok JSON formában vannak tárolva, maga a Google Maps API pedig támogatja a GeoJSON forrásból történő adatok betöltését térképekre. Ezen információk alapján épül fel a modell amit a böngésző 2D vagy 3D formában ábrázol egy térképen.

2.3. GeoSpatial adatbázisok

A GeoSpatial adatbázisok adják az előbbi szolgáltatások hátterét. Ezek az adatbázisok kifejezetten a térbeli objektumokat leíró adatok tárolására, azok egyszerű lekérdezésére vannak optimalizálva. A leggyakoribb adatok pontok, vonalak, vagy poligonok formájában fordulnak elő.

A hagyományos adatbázisokhoz képest sokkal nagyobb teljesítményt nyújtanak ilyen területen, valamint gyakran az ilyen típusú adatok feldolgozásához bővíteni kell azok funkcionalitását. Ennek érdekében az Open Geospatial Consortium 1997-től kezdődően definiálta az ISO 19125 szabványt ezen funckionalitás megvalósításához, melynek második része leírja az SQL-ben történő megvalósítást is. Az OpenGIS szabvány ugyan magába foglalja a CORBA és az OLE/COM implementációkat is, ezek nem részei az ISO szabványnak.

A szabvány többek között a következő műveleteket definiálja:

- Térbeli számítások, azaz objektumok közti távolság, vonalak hossza, stb
- Térbeli predikátumok, objektumok közötti kapcsolatokat leíró logikai lekérdezések
- Geometriai konstruktorok, alakzatot leíró adatok alapján új geometriai elem létrehozása
- Egy tetszőleges objektumhoz tartozó információk lekérdezése

3. fejezet

Matematikai modell részletezése

3.1. Modell specifikálása

A közlekedésre irányuló modell fő tulajdonságai közé kell hogy tartozzon az elegendő információ tárolása ahhoz, hogy alapvetően tudjanak a szimulált járművek rajta közlekedni. Ezt rengeteg féle képpen meg lehet valósítani, sokféle eleme lehet egy adott térképnek, akár minden egyes előforduló úttípusra, szabályra, táblára tekinthetünk úgy mint a modell egyedi része. Ez a megközelítés viszont lehetségesen túlbonyolítaná a modellt, ezáltal a procedurális generálás nem adna annyira elfogadható úthálózatot, lehetségesen életszerűtlen helyzetek épülhetnek fel a sok elem megléte, azok elhelyeződése miatt. Sokkal inkább célszerű a modell részeként tekinteni a főbb építőelemeket, melyek elegendőek magához az úthálózat generálásához, valamint néhány alapvető szabályt megvalósító elemet, mint például a lámpás útkereszteződés, az egyirányú út, valamint a többsávos út.

3.2. A modellben használt elemek

Matematikailag a modell egy gráfból áll, melynek csomópontjai jelölik az útkereszteződéseket, vagy két egyenes útszakasz között az összekötő részt. Az élek jelölik magát az útszakaszokat, ezeknek az éleknek pedig számos tulajdonságai lehetnek amivel különbséget teszek több elem között.

3.2.1. Egyenes út

Alapvető egyenes útszakasz, ennek lehet több paramétere is. Ennek az elemnek elsősorban tartalmaznia kell a rá vonatkozó sebességhatárt, valamint az úton létező sávok számát. Az elem leírásához szükség van annak kezdő és végpontjára, valamint a sávok számára. További információt ad a modell többi részének, ha tartalmazza egy megközelítőleges égtájnak megfelelő irányt. Ebből következtetni lehet arra ha például kanyar következik, és ha igen akkor milyen irányban, csak arra van szükség rá hogy megnézzük milyen módon változik az utak hozzávetőleges iránya.

Alapvetően az út olyan szélességű, hogy elfér egymás mellett két jármű, ezért ha több sáv van csak arra kell figyelni hogy megfelelő helyen közlekedjenek az adott járművek. A gráfon való pozicíójának behatárolásához a kezdő és végpont az ő általa

összekötött két csomópontot jelenti.

3.2.2. Egyirányú út

Hasonló az egyenes úthoz, viszont egyértelműen meg kell határozni a két végpont közül melyik a kiindulási, melyik a célpont. Hasonlóan az egyenes szakaszhoz, meg kell adni a sebességkorlátot, viszont ez az út kizárólag 1 sávos.

A gráfot tekintve itt a két összekötött csomópont között egy irányított él lesz, melynek kezdő és végpontjaként egyértelműen el kell jelölni a két pontot.

3.2.3. Kanyar

Vehető tulajdonképpen külön elemnek, de tulajdonképpen csak két egymást követő egyenes szakasz, melyek az őket összekötő elem mentén nem egyenes irányban folytatódnak. A kanyar iránya adódik az általa összekötött két egyenes útszakasz iránya által.

A gráfon kanyarnak tekinthető az olyan csomópont, melyből csak két él indul ki, ha ez a két él iránya egymástól eltérő, és nem ellentétes.

3.2.4. Útkereszteződés

Három vagy több útszakasz találkozásánál használatos elem. Tulajdonságai közé tartozhat az hogy tartalmaz-e jelzőlámpát az útkereszteződés, vagy sem. Az útkereszteződés jellemezhető annak középpontjával, valamint az azt érintő utak halmazával.

A gráfon ez azt jelenti, hogy amennyiben egy csomópontból legalább 3 él indul ki, az egy útkereszteződés.

3.2.5. Személygépjármű

Legalapvetőbb közlekedési jármű. Az autókról nyilván kell tartani azok jelenlegi pozícióját, mely abból adódik hogy éppen melyik élen haladnak, melyik csomópont felé, és attól milyen távolságra vannak. Minden jármű véletlenszerűen kiválasztott útvonalat kap, amin végig kell haladjon. Ezek az útvonalak a teljes gráfon egy-egy részgráfok. A jármű jellemzői közé tartozik annak sebessége, valamint a következő csomópontot elérve a várható haladási iránya, tehát előre, balra, vagy jobbra megy-e tovább.

Tudnia kell a vele egy úton közlekedő járművekől is, azok pozíciójától függ a viselkedése. A járművek kezdetben véletlenszerű helyen kezdenek, majd ha sikeresen eljutottak a célpontjukhoz eltűnnek a modelből (leparkolt kocsinak tekintve). Ezen útvonal kijelölése egy véletlenszerűen kiválasztott csomóponttól indul, ahonnan véletlenszerű irányba indulunk el, majd a következő csomópontot elérve szintén új irányt választunk addig, ameddig vagy zsákutcába nem ér az autó, vagy eléri a paraméterként maximálisan kijelölt úthosszat. Szintén a paraméterezést tekintve ilyenkor új jármű jelenik meg ha jelenleg kevesebb van mint a megadott maximális érték.

3.2.6. Autóbusz

Kissé különlegesebb járműtípus, vonatkozik rá néhány további szabály. Többek között elsőbbsége van megállóból elindulva, valamint ha lehet próbál jobbra tartani, csak akkor

használja a belső sávot ha többsávos úton balra kanyarodik. A buszmegállók között megkötött útvonalon kell haladnia, mindegyiknél pedig meg kell állnia. Ha elérte az útvonal végét megfordul, és visszafelé halad végig rajta. A szimuláció teljes ideje alatt az útvonalat követi, kezdetben az útvonal végéről elejétől indul.

Ez az útvonal a gráfot tekintve egy legmélyebb szinten elhelyezkedő véletlenszerűen választott csomópont, valamint a tőle legtávolabb lévő csomópont közötti utat jelenti. Kijelölése könnyen megoldható ezen két csomópont szüleit végigkövetve a középen lévő kiindulási pontig.

3.2.7. Buszmegálló

Autóbusznak elhelyezett megállási pont. Két csomópont közötti él felezőpontján helyezkedik el, ezért leírásához elegendő az adott élt megadni. Szükséges információ a buszmegállók közötti minimum távolság, azaz hány csomópontnak kell lennie legalább két buszmegálló között. Ezt a számot véletlenszerűen fogom meghatározni minden buszmegálló elhelyezése után újra számítva.

Ezen elem egy olyan csomópont, amelyből nem indul él, és nem is tart belé él.

3.2.8. Közlekedési szabályok

Néhány alapvető közlekedési szabály amelyet a járműveknek követniük kell, általában attól az elemtől függ ahol haladnak, valamint attól amelyhez tartanak

- Egyirányú útra csakis a kijelölt útirányban hajthatnak rá a járművek
- Ha az út több sávot tartalmaz, a külső sáv csak jobbra kanyarodásra alkalmas, a belső sáv balra kanyarodásra, illetve egyenes haladásra
- Ha az útmenti buszmegállóból elindulni készül a busz, a többi járműnek elsőbbséget kell neki adnia
- Jelzőlámpával ellátott útkereszteződésbe nem hajthat be ha a lámpa piros, ha aközben vált hogy benne van akkor még áthaladhat

3.3. A modell kritériumai

3.3.1. Generálási szempont

Magának az úthálózatnak a generálására vonatkozóan a következők a célok:

- Viszonylag kevés elemből álljon össze a generált úthálózat
- Ne legyen életszerűtlen az összeállított úthálózat
- Tartalmazza az alapvető közlekedési helyzetek szimulálására szükséges elemeket
- Vizuálisan emlékeztessen egy átlagos városra
- Bizonyos mértékig paraméterezhető legyen, főleg a méreteket érintve

Elemek száma

Az említett viszonylag kevés elem, vagy pontosabban kevés különbözű típusú elem azt jelentené, hogy ne legyenek túl specifikusak az elemek, hanem akár könnyedén egyik elem bővítésével elő lehessen állítani egy másikat. Ilyen például a modellben az egyenes szakasz, melyből kis változtatással előáll az egyirányú út.

Életszerű úthálózat

Röviden annyit tesz, hogy nem tartalmaz olyan területeket ahol össze van szűkítve kis területen rengeteg út. Tehát legyen minimális hely kihagyva az utak között az épületeknek. Életszerűtlennek tekinthető továbbá az is, ha egy útkereszteződésből rengeteg út indul ki, azaz 5-6 avagy annál több. Ezért a modellben az útkereszteződés maximálisan 4 utat köthet össze.

Közlekedési helyzetek

Legyen benne jelzőlámpával ellátott útkereszteződés, többsávos út, egyirányú út, valamint buszmegálló. Ezek az elemek szükségesek minden esetben, hogy a szimuláción meg lehessen jeleníteni a lámpák működését, a többsávos utak eseten a járművek sávválasztását és tartását, egyirányú út irányának a betartását, és buszmegállónál a busz megállását, neki járó elsőbbség megadását.

Átlagos város képe

Átlagos városra hasonlít akkor, ha a központtól kifelé haladva ritkul az úthálózat, valamint a többsávos utak az előreláthatólag forgalmasabb helyeken vannak elhelyezve. Ennek érdekében a modellben elhelyezett csomópontok átlagosan kevesebb élt indítanak magukból, minél távolabb vannak a gráf kiindulási pontjától, azaz a középponttól.

Paraméterezés

Ki lehessen jelölni hogy a generálás meddig tartson, azaz hány csomópontot terjesszen ki.

3.3.2. Szimulációs szempont

A már legenerált úthálózaton a szimulációra tekintve ezek a célok:

- A járművek kövessék a modellben definiált alapvető szabályokat
- A szimuláció bizonyos mértékben paraméterezhető legyen, főleg a járművek számát illetően
- Reagáljanak egymásra a járművek
- Ne legyen életszerűtlen a járművek viselkedése
- Teljesítmény szempontjából elfogadható legyen egy adott méretig

Szabályok

A már előbbiekben leírt szabályoknak megfelelően közlekedjenek a járművek, haladási útukat azok alapján válasszák meg.

Paraméterezés

Meg lehessen adni mennyi jármű legyen maximálisan egyidőben az úthálózaton, külön értve a buszokat ettől. Továbbá az is megadható legyen milyen gyakorisággal jelenik meg személygépjármű, és mekkora lehessen a neki kijelölt útvonal maximális hossza csomópontokban mérve.

Reagálás

Ha egy jármű közelít egy másik felé kezdjen lassítani. Ez a lassítás függjön a távolság mértékétől, ha elég közel kerül hozzá teljesen álljon meg. Csak akkor induljon el egy jármű, ha haladási útját nem állja el másik jármű. Közúti baleset, azaz ütközés esetén a két érintett jármű kis idő után tűnjön el, ezzel szemléltetve az elszámolás, elvontatás folyamatát.

Életszerű viselkedés

A járművek tartsák az út vonalát, ne térjenek át a szembe sávba. Gyorsítás és lassítás viszonylag realisztikusan történjen, ne pedig hirtelen. Kanyarodás előtt igyekezzenek lelassítani, kanyart pedig végig lassan bevenni.

Teljesítmény

A szimuláció mérete mind az úthálózat csomópontjait, mind az egyszerre megjelenő autók számát értem. Abban az esetben ha ezek az értékek nem kimagaslóan nagyok, a szimulációnak lényegesebb teljesítményromlás nélkül kell futnia.

4. fejezet

Generálás

4.1. Úthálózat Generálása

Maga a generáló algoritmus megalkotásakor első lépésben a nagyvonalakban leírt elvárt működést vizsgáltam. A hálózatnak középponttól kifelé haladva ritkulnia kell, ezzel közelítve egy valódi várost. A paraméterezéssel kapcsolatos elvárásokat könnyen meg lehet valósítani. Egy gráfként képzelhető el a megalkotott úthálózat, melynek csomópontjai jelölik az egyes útelemek elejét, élei pedig azoknak tartományát. A gráf 0. szintjét jelöltem el a hálózat középpontjának, innen mindig 4 irányba indulnak élek. Mivel az egy csomópontból kiinduló maximális élek számának 4-et választottam, ez garantálja hogy a középpontnak kijelölt ponttól minden irányba nagyjából egyenletes mértékben terjedjen az úthálózat. Minden további csomópont generálódásakor kapja meg annak értékét, hogy mennyi irányba indulnak belőle élek.

4.1.1. Generáló algoritmus

A csomópontok generálása a következőképpen történik:

1. A kiválasztott csomópont kap egy véletlenszerű irányt, észak, dél, kelet, vagy nyugat értékében
2. Ha a kapott érték megegyezik azzal az iránnyal amerre a csomópont szülője van, vagy már indult arra belőle él, újat kap
3. A kiválasztott iránynak megfelelő véletlen generált X és Y koordinátákat kap
4. A kapott koordinátákból alkotott csomópont felkerül a gráfra
5. Ha a jelenlegi csomópontból még kell éleket indítani, akkor kezdődik előről

Az előbbiekben említett X és Y koordinátát kissé pontosítanám. Ha a csomópontot nyugat vagy kelet irányba generáljuk, az X koordináta értéke a jelenlegi pont X koordinátája, hozzáadva egy előre definiált értékek közötti véletlen szám. Az Y koordináta ilyenkor egy minimális eltérés az út fekvésében, hasonlóan generálódik, de lényegesen alacsonabb véletlen számot kap. Ez tulajdonképpen a valóságban is látható "tökéletlenségekre" vezethető vissza, ugyanis a legtöbb esetben ott sem teljesen merőleges egymásra kettő út. Az észak és dél irányba induló pontok esetén ugyan ez az eljárás,

csak a két koordináta szerepe felcserélődik. Fontos viszont az is, hogy a középponttól távolodva átlagosan kevesebb elágazása legyen egy csomópontnak, ezért ehhez felhasználható annak gráfbeli szintje. Minden csomópontról eltárolódik annak szintje (szülő szintje + 1), ami befolyásolja az elágazások számát. Egy másik probléma az, hogy az így generált élek gyakran metszhetik egymást, amely az aluljárók és a hídak hiányában itt nem megengedett, ezért utolsó lépésnek ellenőrizni kell hogy az új él metszi-e bármely meglévő élek közül akár az egyiket is, és ha igen akkor nem kerülhet fel a gráfra. Az él viszont hozzáadható az eredetileg kijelölt csomóponthoz legközelebbi csomóponttal történő helyettesítéssel, feltéve ha így nem történik létező út metszése. A buszmegállók elhelyezésével kapcsolatban a következő feltételeket adtam:

- A legcélszerűbb a hálózat két, egymástól távol lévő, a gráf mélyebb szintjein elhelyezkedő pont közötti út létrehozása
- Az útvonalnak érintenie kell a középpontot, a gráf 0. szintjét
- Lehető legkevesebbszer érintse többször ugyan azt a csomópontot

Ennek érdekében először kijelölöm az út egyik végét, mint megállót. Ezek után keresek egy utat a gráf 0. szintjéhez, melyen legfeljebb minden második csomópontot kijelölöm megállónak. A középpont elérése után kijelölöm a második végpontot, mely a középponttól az eddigi iránynak ellentétesen, a gráf mélyebb szintjei közül kell lennie. Az ehhez a középpontból vezető úton, az utóbbihoz hasonló eljárással kijelölök megállókat. Az utak sávszámának meghatározására a gráfbeli mélységüket használom. A magasabb szinten lévő élek nagyobb valószínűséggel lesznek többsávosak. Ez a generálás végén választódik ki. Az egyirányú utak kijelölése is a folyamat legvégén történik. Ennek az oka magából az utak jellegéből adódik, ugyanis egyirányú út nem végződhet zsákutcában, és nem is zárhatja el a hálózat egy részét a többi elől. Éppen ezért egy olyan részgráfot kell találni amely Hamilton-kört alkot, amelyen egy részgráfot már nyugodtan egyirányúvá lehet tenni.

4.1.2. Az egyirányú út

Egy könnyű módszer annak megoldására, hogy mely utak lehetnek egyirányúak az lenne, ha a generálás során egymást metszett utakból kialakult éleket jelölöm el egyirányúnak. Ezek az élek ugyanis biztos hogy egy létező Hamilton-út részei, mivel az eredetileg fagráfnak megfelelő úthálózat bármely két csomópontja közé húzott él Hamilton-utat ad.

4.2. Alapelemek megjelenítése HTML Canvas segítségével

4.2.1. Fejlesztői környezet felállítása

Az algoritmus működését, eredményét egy HTML Canvas objektumon szemléltetem. Magát a kódot JavaScript-ben írtam, annak ECMAScript 6-os verziójában. Fejlesztői környezetnek A JetBrains által készített WebStormot használtam. Először is létrehoztam egy HTML fájlt, ahol a body tag-en belül elhelyeztem a canvas elemet. Erre az

elemre JavaScriptből a HTML-ben megadott id-je alapján lehet hivatkozni. Ezután létrehoztam egy JavaScript fájlt, melyben egy (document).ready() szintaktikában elhelyezett függvényhívással indítom a generálást. A (document).ready() a jQuery függvénykönyvtárnak része. Azért szükséges ebben az esetben, mert ameddig a html fájl nem töltött be teljesen, a script nem futtatható mert a canvas elemre nem létezne semmilyen referencia. Ez a részlet biztosítja hogy csak akkor kezdődjön a script futtatása, ha a HTML documentum teljesen betöltött.

4.2.2. Létrehozott osztályok

Node

A gráfon egy csomópontot leíró objektum. Egy konstruktort tartalmaz, amely beállítja az osztály 6 adattagjának értéket. Ezek az adattagok a következőek:

- x: A csomópont X koordinátája
- y: A csomópont Y koordinátája
- branches: Hányfelé kell tovább ágaztatni a csomópontot
- level: A gráf hanyadik szintjén helyezkedik el a csomópont
- connectedFrom: Milyen irányból lett kiterjesztve a csomópont
- parentNodeIndex: A gráfban lévő csomópontokat tartalmazó vektoron belül ezen csomópont szülőjének indexe

Edge

A gráfon egy él leírására szolgáló objektum. Egy konstruktort tartalmaz, amely inicializálja 4 adattagját. Ezek az alábbiak:

- from: Az a csomópont, ahonnan az él kezdődik
- to: Az a csomópont, amiben az él végződik
- lanes: Az él által jelölt úton hány sáv van
- oneway: Logikai változó, az út egyirányú-e vagy sem

Graph

Magát a gráfot leíró objektum, mely tartalmaz 2 függvényt annak canvas-on történő kirajzolására. Paraméter nélküli konstruktora 2 adattagját inicializálja üres értékkel. Adattagjai és függvényei a következők:

- Nodes: A csomópontokat tartalmazó vektor
- Edges: Az éleket tartalmazó vektor
- drawAllNodes: Függvény, mely végig iterál a Nodes vektoron, és mindegyik elemére meghívja a kirajzoló függvényt
- drawAllEdges: Függvény, végig iterál az Edges vektoron, egyirányú út esetén nyilat rajzol, ellenkező esetben egyenes vonalat

4.2.3. Segédfüggvények

intersects

Ez a függvény arra szolgál, hogy kiszámítsa két él metszi-e egymást. Ehhez két paramétert vár, edge1 és edge2 néven. Visszatérési értéke logikai típusú.

distance

Két csomópont közötti távolság kiszámítására szolgáló függvény. Ehhez a két vektor által meghatározott szakasz hosszát számolja ki pitagorasz tétellel. Ezzel az eredménnyel tér vissza.

maxBranchesReached

Ezzel a függvénnyel megmondható hogy egy adott csomópontba már lett-e 4 él húzva. A függvény paramétereiként megkapja a keresett csomópontot, a gráfot, és a jelenleg kiterjesztés alatt álló csomópont gráfbeli indexét. Amennyiben a gráfban megtalált csomópont branches értéke kisebb mint 3, akkor növeli 1-el és hamis értékkel tér vissza, azaz még nem volt 4 él húzva. Ellenkező esetben igaz értéket ad.

getBranchCount

Ez a függvény egy egész számot kap paraméterként, mely egy csomópont gráfbeli mélységére utal. Ez alapján a szám szerint visszaadja hogy mennyi irányba ágazzon el az adott csomópont. Első szinten 80% az esélye hogy 4 irányban ágazik el, ez szintenként 20%-al csökken. Ha nem 4 irányban ágazik el, véletlenszerűen generált számot ad vissza 0 és 2 között, mivel a csomópont szülőjéhez vezető élt ilyenkor nem számoljuk.

findMaxLevelNodes

A gráf legmélyebb szintjén lévő csomópontok megtalálására szolgáló függvény. Paraméterként megkapja a gráfot. Első lépésben beállítja a max értéket a gráf első csomópontjának szintjére, és inicializálja a maximális mélységben elhelyezkedő csomópontok vektorát. Ezután maximum kereséssel beállítja max értéket. Majd a gráf Nodes vektorán végig iterálva hozzáadja az ezen max értéknek megfelelő mélységű csomópontokat az eredmény vektorhoz. Végül ezzel a vektorral tér vissza.

DrawBStops

Buszmegálló rajzolására használt függvény. Egy csomópontot kap paraméterként. A kontextust felhasználva felrajzolja a canvasra egy zöld négyzet formájában. Visszatérési értéke nincs.

makeBusStops

Ez a függvény szolgál a buszmegállók elhelyezésére. Egyelőre csak a hozzávetőleges helyüket jelöli, nem az él közepére teszi a helyüket, hanem a csomópontokra. Paraméterként megkapja a gráfot. Először meghívja a findMaxLevelNodes függvényt, ebből a kapott értéket eltárolja egy változóba.

Kijelöl az útvonal kezdetének az előbb kapott vektorból egy véletlenszerű elemet. Ezt felrajzolja a DrawBStops függvénnyel. Ezután végig iterál a maximum mélységű tömbökön, maximum keresést végezve a kiindulási ponttól mért távolságot illetően. Ehhez a distance függvényt használja. Ha megvan a végpont azt is felrajzolja. Ezután elindul a kiindulási ponttól, sorra véve a csomópontok szüleit. Minden buszmegálló kirajzolása után kap egy véletlen értéket, amely megmondja hogy hány csomópont után kell rajzolnia mégegyet. Ha eljutott a gráf kezdőpontjáig megcsinálja ugyan ezt az út végpontjától is. Visszatérési értéke nincs.

drawCanvasNode

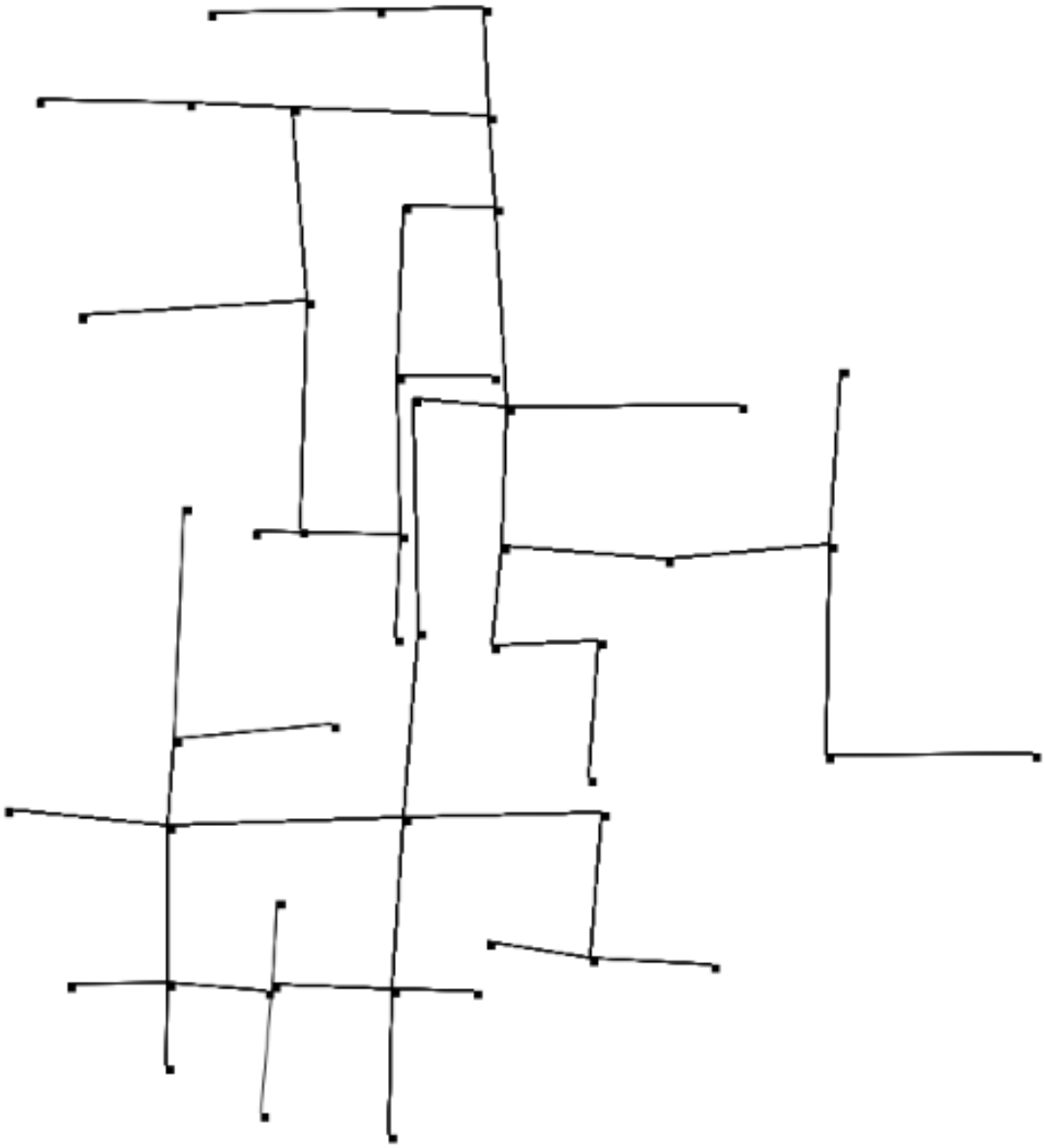
Egy paraméterként megkapott csomópont koordinátáinak megfelelő helyre négyzetet rajzol a canvasre.

drawCanvasEdge

Egy élt kap meg paraméterként, melynek kezdő és végpontjai között egyenes vonalat húz a canvasre.

4.2.4. A generáló függvény

Ha ez megtörtént, meghívódik a DrawOnCanvas függvény. Először is ez eltárolja változóként a canvas-ra vonatkozó referenciát. szintén új változóba, utána a myCanvas.getContext("2d") létrehoz egy CanvasRenderingContext2D objektumot, mellyel a canvasra rajzolni lehet. A következő lépésben megtörténik a paraméterek beállítása, azaz mennyi legyen a maximális csomópontok száma, átlagosan milyen hosszúak legyenek az utak. A kontextust elmozdítjuk alapértelmezett helyéről, (0,0)-ról az (1000,500) pontba. Létrehozzuk a kezdő csomópontot ezen a pozíción



4.1. ábra. Kép a generáló algoritmus eddigi eredményéről

Ezek után következik az utak sávszámának kiadása, az egyirányú utak elhelyezése, valamint a buszmegállók generálása.

5. fejezet

Tervezés, implementáció

- Bemutatni a használt eszközkészletet. - UML osztálydiagram, szekvencia diagramok, blokkvázlat és hasonló dolgok. - JavaScript implementációval kapcsolatos tudnivalók. - WebGL-ről részletesen írni.

6. fejezet

Szimulációk

- Különféle paraméterezéseket megnézni. - Statisztikai jellegű vizsgálatokat végezni a kapott eredményekre.

7. fejezet

Összegzés

Irodalomjegyzék

- https://www.researchgate.net/publication/228966705_A_Review_of_Traffic_Simulation_Software
- https://www.anylogic.com/road-traffic/
- https://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931_read-41000/

CD-melléklet tartalma

A dolgozat PDF változatát a `dolgozat.pdf` fájlban találjuk.

A dolgozat LaTeX segítségével készült. A forrásfájlok a `dolgozat` jegyzékben találhatók.

...