* 1. **TЕХНОЛОГИЧНО УЧИЛИЩЕ ЕЛЕКТРОННИ СИСТЕМИ**



**към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**

**ДИПЛОМНА РАБОТА**

**по професия код 481020 „Системен програмист“**

**специалност код 4810201**  **„Системно програмиране“**

Тема: Оптимизация на светофари в симулация на трафик

Дипломант: Дипломен ръководител:

*Борис Веселинов Ханджиев Иван Кръстев*

СОФИЯ

2 0 2 5

**TЕХНОЛОГИЧНО УЧИЛИЩЕ ЕЛЕКТРОННИ СИСТЕМИ**



**към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**

Дата на заданието: 28.10.2024 г. Утвърждавам:..............................

Дата на предаване: 28.01.2025 г. /проф. д-р инж. П. Якимов/

**ЗАДАНИЕ**

**за дипломна работа**

**ДЪРЖАВЕН ИЗПИТ ЗА ПРИДОБИВАНЕ НА ТРЕТА СТЕПЕН НА ПРОФЕСИОНАЛНА КВАЛИФИКАЦИЯ**

**по професия код 481020**  **„Системен програмист“**

**специалност код 4810201**  **„Системно програмиране“**

на ученика Борис Веселинов Ханджиев от 12 В клас

1. Тема: Оптимизация на светофари в симулация на трафик
2. Изисквания:

- Да се разработи симулация на трафик с автомобили и светофари;

- Поведението на автомобилите да имитира това в реалния свят;

- Да може да се оптимизират светофарите чрез алгоритъм за машинно обучение;

- Да има елементарен потребителски интерфейс;

- Да има алгоритъм за генериране на случайна пътна мрежа;

- Да може да се събират данни от симулацията.

1. Съдържание 3.1 Теоретична част
   1. Практическа част
   2. Приложение

Дипломант :...........................................

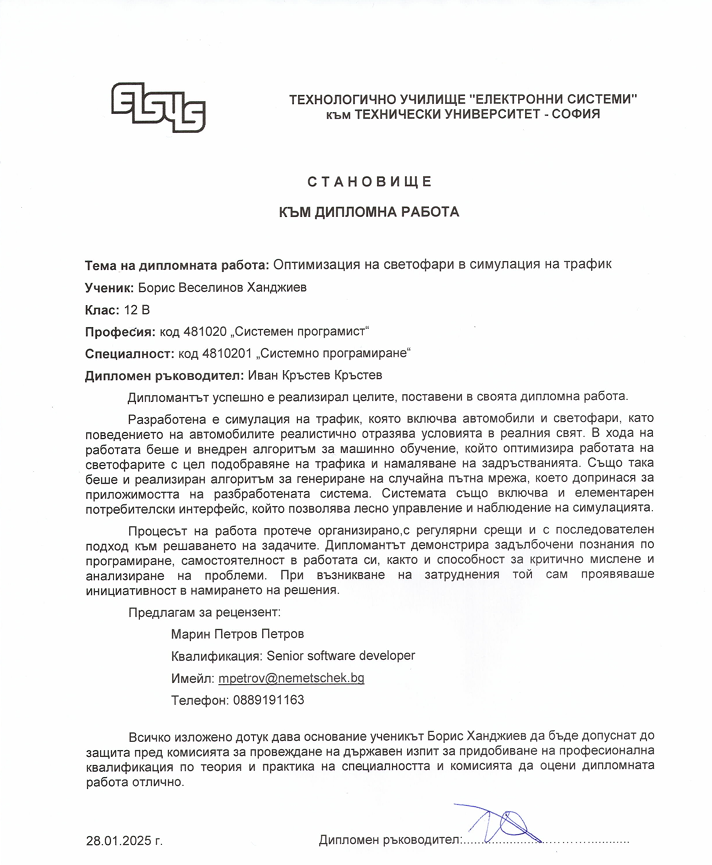
/ Борис Ханджиев /

Ръководител:..........................................

/ Иван Кръстев /

ВРИД Директор:...............................................

/ ст. пр. д-р Веселка Христова /



**Увод**

Най-големият недостатък на светофарните системи е, че те често работят по статични графици, което ги прави неспособни към ефективното адаптиране към промените на трафика. Този проблем, заедно с разрастването на градовете и нарастващия обем на превозни средства по съществуващата инфраструктура, води до множество проблеми, като: повишени вредни емисии от автомобилите, по-дълго време за пътуване до работни места и домове и повишена раздразненост у водачите, което е опасно, както и за тях, така и за останалите водачи на пътя.

Светофарните системи спадат към понятието за критични инфраструктури. Това са системи, или инфраструктура, чиито неизправности или спирания биха донесли значителни опасения за безопасността и здравето на населението и околната среда. В случая на механизмите, които управляват светлинните сигнали, тези опасения са катастрофите и евентуалната загуба на живот.

Поради тази причина, светофарните алгоритми и контролери винаги се изпробват в симулирана среда преди тяхното внедряване в истинския живот. Освен безопасност, също може да се анализират различни показатели, като цялостна ефикасност, отделени емисии, цени и т.н. Още едно предимство на експериментирането в симулационна среда, е възможността за нагласяването на отделни параметри, като гъстота на задръстването, различни ограничения на скоростта, различен брой ленти, час в денонощието и др.

Тази дипломна работа реализира примитивна апликация, която симулира трафик и подпомага оптимизирането на светофарите на кръстовищата чрез алгоритъм за машинно обучение.

**Първа глава**

**Проучване на технологии и съществуващи решения**

* 1. **Технологии за създаване на симулация на трафик**
     1. **Игрови двигатели**

Гейм енджини като Unreal Engine, Unity и Godot, които са едни от най-известните на пазара, предлагат мощни възможности за 2D/3D визуализация и физични симулации, което ги прави подходящи за създаване на сравнително точна трафик симулация. С употребата на игрови двигател също се премахва нуждата за огромно количество „boilerplate“ код, което позволява бързо прототипизиране и разработване на софтуер.

* + 1. **Програмни езици и библиотеки от ниско ниво**

За по-леки, бързи и контролируеми, спрямо изискванията на разработчика, симулации, бързи езици като C/C++ и библиотеки като raylib или SDL (Simple DirectMedia Layer) са идеалния избор. Те предоставят опростен и сравнително лесен за използване интерфейс за програмиране на видеоигри. Минусите на този подход са плюсовете на игровите двигатели, а именно, че прототипизирането е много по-бавно и „boilerplate“ кода е повече, а също и графичните и физичните възможности са много по-ограничени.

* 1. **Съществуващи симулации на трафик**
     1. **SUMO**

SUMO (Simulation of Urban MObility) е софтуерен пакет с отворен код, който може микроскопично да симулира големи мрежи от пътища. SUMO е разработен от Германския аерокосмически център, както и от потребители от общността и е написан на C++, Java и Python. Той е достъпен като СОК от 2001 г., а от 2017 г. е проект на фондацията Eclipse. SUMO позволява използването на различни софтуерни инструменти за настройване на параметри и анализиране на пътния трафик и се използва за изследователски цели, като: прогнозиране на трафика, оценка за ефективността на светофарите и избор на маршрути.

SUMO може да симулира различни автомобили, обществен транспорт и пешеходци и идва с множество средства за създаването на уникални трафик сценарии. Понеже SUMO е по-лек и по-елементарен софтуер в сравнение със следващия (Vissim), той е по-подходящ и се използва по-често за мащабни, ориентирани към данни, изследователски цели. Независимо от това възножността за интегриране с външни софтуерни инструменти, факта, че SUMO е отворен код и цялостната гъвкавост на софтуера, поставят SUMO на едно от челните места, при избиране на софтуер за трафик симулация.

* + 1. **PTV Vissim**

PTV Vissim е многомодулен софтуерен пакет за микроскопично симулиране на трафиков поток. Той е разработен от PTV Planung Transport Verkehr AG в Карлсруе, Германия през 1992 г. Името му е абревиатура на "Verkehr In Städten - SIMulationsmodell" (На английски - "Traffic in cities - simulation model"). Приложенията на PTV Vissim варират от пътно инженерство, обществен транспорт, противопожарни защити (симулации на евакуации) до 3D визуализация за илюстративна цел.

В Vissim може да се симулират различни участници в движението, като: коли, автобуси, камиони, рейсове, велосипеди, мотоциклети, пешеходци, каруци и др., както и различна обществена инфраструктура, като: паркинги, летища, гари и големи сгради. В сравнение с предишния софтуер (SUMO), Vissim се използва повече за детайлно планиране на трафик в истинския живот, тъй като Vissim е с по-голяма прецизност и физична акуратност от SUMO. Широкият му спектър от трафик агенти, заедно със възможността за интеграция с други PTV продукти като PTV Visum (водещ световен софтуер за планиране на трафик), превръщат Vissim в индустриален стандарт.

* + 1. **Aimsun Next**

Aimsun Next е мощен многомащабен софтуерен пакет за симулация на трафик, който може да работи на микроскопично, мезоскопично и макроскопично ниво. Разработен е от компанията Aimsun, част от Yunex Traffic, и е използван от инженери, градски плановици и изследователи по целия свят. Благодарение на способността си да моделира сложни транспортни мрежи, Aimsun Next е широко приложим в области като интелигентни транспортни системи (ITS), управление на трафика и анализ на въздействието на инфраструктурни проекти.

Aimsun Next може да симулира различни участници в движението, включително автомобили, обществен транспорт, пешеходци и велосипедисти, както и специфични сценарии като автономни превозни средства и споделена мобилност. В сравнение със SUMO и PTV Vissim, Aimsun Next предлага гъвкавост при избора на мащаб на симулацията, което го прави подходящ както за детайлно моделиране на отделни кръстовища, така и за мащабни транспортни стратегии на градско и регионално ниво. Възможността за интеграция с други транспортни модели и данни в реално време утвърждава Aimsun Next като водещ инструмент за анализ и оптимизация на транспортните системи.

* 1. **Съществуващи методи за оптимизиране на трафик**

В реалния свят ефективното оптимизиране на трафика е възможно само с устройства, които позволяват комуникацията Vehicle-to-Everything (V2X), което позволява на превозни средства, светофари и инфраструктура да обменят данни в реално време.

Методи за проследяване на трафик като сензори с индуктивна верига, радар, LiDAR, камери и проследяване, базирано на GPS, предоставят важна информация за потока на превозните средства и нивата на задръстванията. Комуникационни методи като Dedicated Short-Range Communication (DSRC), Cellular V2X (C-V2X) и 5G позволяват на свързаните превозни средства да взаимодействат със системите за трафик за по-добра координация.

В контекста на симулационна среда обаче, не е нужно да се тревожим за комуникационни предизвикателства в реалния свят – позицията, скоростта и намерението на всяко превозно средство вече са известни, което улеснява прилагането и тестването на стратегии за оптимизиране на трафика. Това, което следва са някои от методите за ефективно управление на светофари:

* + 1. **Сигнали фиксирани във времето**

Контролът с фиксирано време е метод за оптимизиране на светофара,при който продължителността на зелената, жълтата[[1]](#footnote-1) и червената светлина е предварително програмирана въз основа на фиксирани времетраения или исторически данни за трафика. Циклите на сигнала се изпълняват по повтарящ се график, независимо от действителните условия на трафика, което го прави ефективен в предвидими, стабилни среди на трафика, но неефективен при променливи количества автомобили.

Разновидност и подобрение на този метод са предварително зададени пътни сигнали, които използват различни планове спрямо часа в денонощието. Примери за такива планове са: план за час пик (17:00 – 18:00), нощен план (12:00 – 6:00) и др. Въпреки че все още не отговаря на трафика в реално време, този подход подобрява ефективността чрез адаптиране към очакваните трафикови обеми през цялото денонощие.

Текущо това е оспоримо най-разпространеният метод по света, поради леснотата на имплементация и ниската цена. Все повече обаче се наблюдава преминаването към по-адаптивни методи за управление на светофарите, като тези изброени по долу.

* + 1. **Задействане на сигнали**

Метод на задействане на сигналите е подход за оптимизиране, който настройва времената на сигнала въз основа на наличието на трафик в реално време. В случая на обикновено светофарно кръстовище - системата открива присъствието на ППС-та и съответно адаптира продължителността на зеления сигнал. В реална обстановка, както бе посочено горе, това става възможно с различни сензори, но в симулация – всички тези данни вече се знаят.

Този метод позволява по-гъвкав и по-отзивчив контрол в сравнение със системите с фиксирано време, като намалява ненужното време на чакане и подобрява трафика, особено в периоди на слабо натоварване.

Разновидност на задействаните сигнали са полузадействани пътни сигнали, където се умешват този и горния метод. Пример е, когато главен път работи с фиксирен график, а пък страничните улици, работят на принциа на задействане, тогава, когато се появи трафик по страничните улици, те се изчистват възможно най-рано, като това осигурява минимално време за чакане.

* + 1. **Адаптиране на сигнали**

Адаптивният контрол е последният метод, който ще бъде разгледан и той представлява динамично настройване на времената на сигналите в реално време въз основа на данни за непрекъснат трафик.

Използвайки усъвършенствани алгоритми, системата непрекъснато следи условията на трафика и коригира продължителността на различните светлини, за да оптимизира трафика. Този метод е много ефективен при справяне с непредсказуеми вълни от трафик, внезапни скокове или часове извън пиковите натоварвания.

Разновидност на адаптивния контрол е централизиран адаптивен контрол, където данните от множество кръстовища в мрежа се събират и обработват централно, което позволява координирани корекции в по-широка област за допълнително оптимизиране на потока на трафика и за получаването на явления като зелени вълни[[2]](#footnote-2).

* 1. **Техники за създаване на адаптивни трафикови системи**
     1. **Прости алгоритми базирани на правила**

Простите алгоритми, базирани на правила, представляват детерминирани методи за управление на светофарите, които разчитат на предварително зададени логически правила и прагови стойности. Те не използват машинно обучение, а директно боравят с предварително дефинирани критерии, като: дължината на опашката от автомобили, времето на чакане, потоци от автомобили или исторически данни за трафика.

Примери за такива алгоритми са Webster’s method[], SCOOT (Split Cycle Offset Optimization Technique)[] или SCATS[], които анализират и правят корекции в реално време.

Предимствата им са, че те са прости, детерминистични и предсказуеми, с по-малка изчислителна сложност и евтини за интегриране. Недостатъците са като цяло плюсовете на метода с машинно обучение – скалируемост и адаптация.

* + 1. **Машинно обучение**

Различните подкатегории на машинно обучение включват: Supervised Learning, Deep Learning, Deep Reinforcement Learning (deep RL) и др. Едни от най-популярните библиотеки за обобщено машинно обучение са Tensorflow и PyTorch, а пък за RL са OpenAI Gym, Learning Agents в контекста на Unreal Engine и ML-Agents в контекста на Unity.

Идеята е, че чрез трафик данни, използвайки една от горепосочените библиотеки, може да се тренира модел, който да се научи да се адаптира към състоянието на пътната обстановка и да превключва светофарните сигнали по ефикасен начин.

Предимството на ИИ-базиран подход за адаптивните трафикови системи спрямо стандартни алгоритми е, че алгоритъмът с машинно обучение е значително мащабируем – възможност е приложението му за цели градове, а не само за единични кръстовища. Също, има потенциал за по-добра крайна оптимизация на светофарите – особено измежду няколко кръстовища и пътни коридори.

**Втора глава**

**Изисквания, избор на средства, преглед на проекта**

2. 1. **Изисквания към софтуерния продукт**
      1. **Изиквания от заданието на дипломната работа**

От заданието на дипломната работа (2 стр.), това са някои от задължителните изисквания:

* Да се разработи симулация на трафик с автомобили и светофари;
* Поведението на автомобилите да имитира това в реалния свят;
* Да може да се оптимизират светофарите чрез алгоритъм за машинно обучение;
* Да има елементарен потребителски интерфейс;
* Да има алгоритъм за генериране на случайна пътна мрежа;
* Да може да се събират данни от симулацията.
  + 1. **Допълнителни изисквания**

Освен гореизброените изисквания, тези долу са изисквания, които би било хубаво да има в този проект, понеже те обогатяват възможните действия, преживяването на потребителя или улесняват някои задачи:

* Да има система за запазване/зареждане на симулации чрез файлове;
* Да има настройки и параметри на симулацията, които да открият по-голям брой възможности на симулацията;
* Да има прости настойки за контролиране на времето: паузиране и забързване;
* Да има възможност за селектиране на автомобили или кръстовища, където да се визуализират пътя на автомобила и допълнителни данни.

2. 2. **Технологии**
      1. **Игрови двигател – Unity**

За дипломната работа и разработката на трафик симулация е избран гейм енджинът Unity. Той е мощен и широко използван гейм енджин предимно за разработката на игри, но напълно е приложим за филми, анимации, симулации, научни цели и т.н. Той предлага гъвкавост и богата екосистема от инструменти за работа с 3D и 2D среди. Някои от чертите на Unity, правещи го добър избор, са следните: кросплатформена поддръжка, C# като скриптов език, който е лесен за използване и има гигантска колекция от функционалности в .NET фреймуърка, Universal RP, Unity Asset Store, вградена физика, надграждаща PhysX на NVIDIA и много други.

Причините за избиране на Unity пред Unreal Engine са следните:

* Unreal Enginе има по-малък брой asset-и от Unity, заема по-малък процент от пазара и използва C++, който е по-сложен и по-времеемък за програмиране.
* Unreal Engine всъщност набляга повече на графичните си способности – това го прави по-употребяван специфично за CGI цели и за AAA игри за разлика от Unity. Това не е проблем обаче, тъй като една симулация за трафик не се нуждае от тежки графики.

Също причината, поради която не е избран Godot, е че на него му липсват повече asset-и, ресурси онлайн и интеграция с различни софтуери.

Последно, избора на Unity пред език от ниско ниво, като C++ и библиотеки като raylib и SDL, може да се аргументира лесно с факта, че много аспекти би трябвало да се имплементира почти от нулата и работата за физика, рендиране и потребителски интерфейс би изисквала много повече време.

* + 1. **Алгоритъм за оптимизиране на трафика - самообучение с утвърждение**

За оптимизацията на светофари, дипломната работа използва алгоритъм с самообучение с утвърждение. Самообучение с утвърждение (на английски: reinforcement learning) е подходящ за оптимизация на трафика, защото позволява на алгоритмите да се учат чрез проби и грешки, като постепенно подобряват управлението на светофарите. RL може да вземе предвид множество фактори като дължина на опашките, средно време на чакане и натовареност на различни пътни участъци.

Основните предимства на RL спрямо традиционните алгоритми включват:

* Този метод на трениране може да се адаптира към динамични и непредсказуеми, както и различни от обичайното, трафикови условия, особено чрез експериментация и достатъчно време за трениране;
* Възможно е автоматичното откриване на най-оптималните за ситуацията стратегии, това е голямо предимство пред адаптивни трафик механизми с конкретни зададени правила и прагови стойности;
* Подходите базирани на машинно обучение често са скалируеми, в конкретния случай на трафик симулация, този алгоритъм може да се транслира към съседни кръстовища, региони и градове.
  + 1. **Библиотека за самообучение с утвърждение - ML-Agents**

За целите на самообучение с утвърждение е избран ML-Agents. ML-Agents е Unity проект, който улеснява трениране чрез Deep Reinforcement Learning (deep RL) на модели в симулации вървящи в гейм енджина. Сам по себе си ML-Agents не извършва тренирането, а питонски бакенд, който използва TensorFlow или PyTorch, като за deep RL има различни методи на трениране, като PPO, DQN и други.

ML-Agents е първият кандидат, тъй като е най-популярен в Unity екосистемата и също има много документация и ресурси онлайн. Пакетът, който може да се свали в игровия двигател, предоставя кодов интерфейс за лесно взаимодействие с трениращия процес.

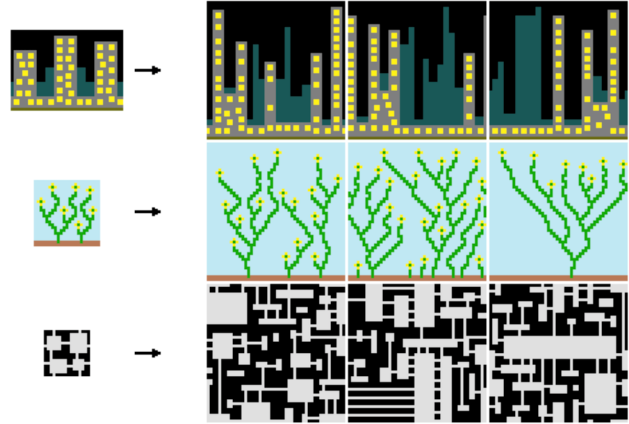
* + 1. **Интегрирана среда за разработка – Visual Studio**

За разработката на проекта е избран Visual Studio като основна интегрирана среда за разработка (IDE). Това е мощен и широко използван инструмент за програмиране, който предлага пълна поддръжка за C# - Unity.

Причините за избора са: възможност за директно дебъгване на Unity програмата чрез точки на прекъсване, богат IntelliSense и широкия набор от инструменти за анализиране на апликацията.

Други IDE-та, които може да се използват за разработка на приложение са JetBrains Rider и MonoDevelop, който идва с Unity, но изборът на средата за разработка е най-вече избор на разработчика, понеже, с достатъчно опит, с всичките IDE-та, може да се напишат едни и същи скриптове.

* 1. **Теория на дипломната работа – подход и алгоритми**
     1. **Алгоритъм за генериране на произволен път – WFC**

Ключовата част от генерирането се изпълнява от алгоритъма WFC. Това е алгоритъм за рестриктивно процедурно генериране, който се основава на локални шаблони и правила за съседство между елементи. Той най-често се използва в контекста на решетка от елементи, като съседи са докосващи се клетки. Долу са посочени примерни резултати от алгоритъма:

Фиг !. *Примери на WFC.*

*Отляво – тайлсет, който е вход на алгоритъма.*

*Отдясно – генерирана решетка от клетки.*

WFC е подходящ избор за генериране на пътища, поради неговия мрежов характер. Конкретно тази трафик симулация, може да генерира една пътна мрежа в следните няколко стъпки:

* **Дефиниране на тайлсет**:

WFC работи над даден тайлсет – това е множеството от възможните елементи и техните правила за съседство в една клетка от решетката. Може да дефинираме често срещани елементи от пътната инфраструктура (и всичките им ориентации) като:

* + Прав път:

Фиг !. *Прав път с вертикална ориентация и съседство на север и юг.*

Фиг !. *Прав път с хоризонтална ориентация и съседство на изток и запад.*

* + Завой:

Фиг !. *Завой със съседство на запад и север.*

Фиг !. *Завой със съседство на север и изток.*

Фиг !. *Завой със съседство на запад и юг.*

Фиг !. *Завой със съседство на изток и юг.*

* + Т-образно кръстовоще:

Фиг *Т-образно кръстовище със съседство на север, изток и запад.*

Фиг !. *Т-образно кръстовище със съседство на запад, изток и юг.*

Фиг !. *Т-образно кръстовище със съседство на север, изток и юг.*

Фиг !. *Т-образно кръстовище със съседство на север, юг и изток.*

* + Обикновено кръстовище:

Фиг !. *Кръстовище със съседство във всички посоки.*

* + Задънена улица:

Фиг !. *Задънена улица със съседство на север.*

Фиг !. *Задънена улица със съседство на запад.*

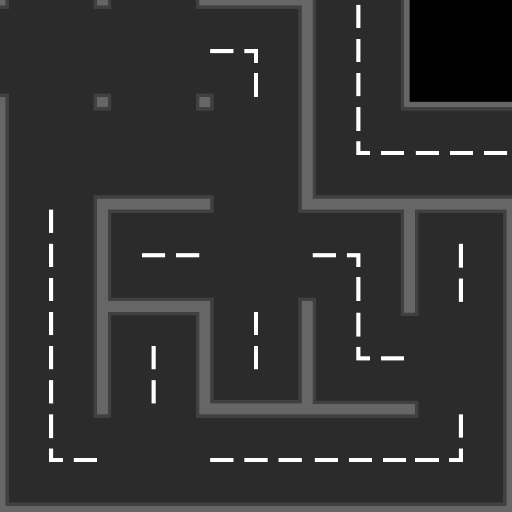
Фиг !. *Задънена улица със съседство на юг.*

Фиг !. *Задънена улица със съседство на изток.*

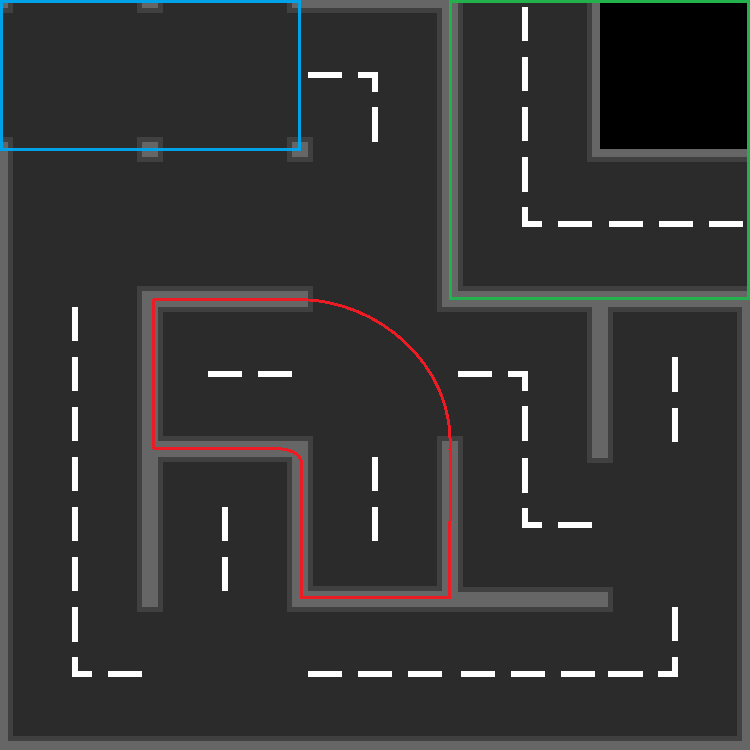
* **„Колапс на вълнова функция“:**

Това е същинската част от алгоритъма. Избира се начална клета и понеже се генерира град, това може да е обикновено кръстовище в центъра на града. Разглеждат се всички околни клетки и се прилагат ограниченията, която тя поставя върху съседите ѝ, според дефинираните ѝ съседства. След това се избира съседа с най-малко възможности и се избира произволен елемент от пътя, като след поставянето целият процес се повтаря до изпълване на решетката. В случая, че няма възможност за клетка, тя може да се остави празна, както в примера долу.

Понеже клетките, които представляват елементи от пътя, са със съседства, където води пътя, в края на този алгоритъм решетката би трябвало да се изпълни с пътни клетки, които са свързани логично – или път за път, или тротоар за тротоар.

В края на тази стъпка би трябвало да има решетка, която изглежда по подобен начин на долупоказания пример:

Фиг !. *Примерен генериран град с размер на решетката 5.*

Алгоритъмът засега има 3 основни проблема:

Фиг !. *Три проблема в генериран град по-първия метод. Съответно:*

*В синьо – пътищата излизат от ограничението на решетката.*

*В зелено – изолирани части от пътната мрежа.*

*В червено – задънени улили близо до центъра на града (подобно многото кръстовища близо до ръба на града – горе ляво).*

Тези проблеми може да се отстранят по следните начини:

* + За клетки на ръба на решетката, се ограничават възможностите до само задънени улици. Ако няма останали възможности пък, както клетката най-горе вдясно в примера, тогава клетката се оставя празна.
  + За изолираните части от пътя може да се използва DFS алгоритъм, който да открие всички свързани сегменти от пътища, после да се избере централния и да се съкратят останалите, като се заместят с празни клетки .
  + За последния проблем, може да се дефинират вероятностни функции за всички очастъци от пътя, които дават шанса даден участък да се появи спрямо разстоянието му от центъра на града. Кръстовища ще имат по-голяма вероятност да се появат близо до центъра, а задънени улици не толкова. Далече от центъра, близо до ръба на решетката, обратното е вярно.
* **Разширение на клетките**

Последната стъпка на алгоритъма за генериране е разширяване на клетките с някакъв константен брой клетки, за да се придобие вид на град, понеже до момента съседни кръстовища и завои са с никакво разстояние между тях. Това може да се постигне като текущите клетки се преместят в увеличена решетка, която е с размер:

Фиг !. *Формула за размера на разширената решетка, където:*

*n2* – *размер на разширената решетка*

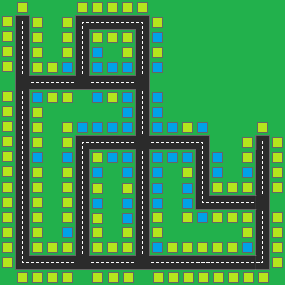
*n1* – *размер на оригиналната решетка*

*k – брой клетки на разширение*

Причината за добавяне на 2, е за да може да се осигури място за клетки сгради, които са по ръбовете на решетката - по най-далечните улици.

След разширяването на решетката, може да удължим тайлсета на алгоритъма, като добавим клетка представяща сграда. Итерирайки през решетката, постигаме две цели:

* + Запълването на разстоянията между кръстовищата с прави улици;
  + Добавяне на сгради в клетките, които се допират до улица.

Последно се избира тип на сграда – дом или работно място. Отново, този избор е базиран на вероятностна функция спрямо разстоянието от центъра на града. Накрая пътната мрежа изглежда, подобно, на тази долу и алгоритъмът е готов, а симулацията може да започне.

Фиг !. *Примерен резултат на алгоритъма за генериране на произволна пътна мрежа:*

*Синя сграда – работно място*

*Зелена сграда – дом*

**Трета глава**

**Реализация на дипломаната работа**

* 1. **…**

**Четвърта глава**

**Ръководство за потребителя**

* 1. **…**

**Заключение**

**Литература**

**Съдържание**

[Увод 4](#_Toc189927260)

[Първа глава 5](#_Toc189927261)

[1.1 Технологии за създаване на симулация на трафик 5](#_Toc189927262)

[1.1.a Игрови двигатели 5](#_Toc189927263)

[1.1.b Програмни езици и библиотеки от ниско ниво 5](#_Toc189927264)

[1.2 Съществуващи симулации на трафик 5](#_Toc189927265)

[1.2.a SUMO 5](#_Toc189927266)

[1.2.b PTV Vissim 6](#_Toc189927267)

[1.2.c Aimsun Next 7](#_Toc189927268)

[1.3 Съществуващи методи за оптимизиране на трафик 8](#_Toc189927269)

[1.3.a Сигнали фиксирани във времето 8](#_Toc189927270)

[1.3.b Задействане на сигнали 9](#_Toc189927271)

[1.3.c Адаптиране на сигнали 10](#_Toc189927272)

[1.4 Техники за създаване на адаптивни трафикови системи 10](#_Toc189927273)

[1.4.a Прости алгоритми базирани на правила 10](#_Toc189927274)

[1.4.b Машинно обучение 11](#_Toc189927275)

[Втора глава 12](#_Toc189927276)

[2.1 Изисквания към софтуерния продукт 12](#_Toc189927279)

[2.1.a Изиквания от заданието на дипломната работа 12](#_Toc189927280)

[2.1.b Допълнителни изисквания 12](#_Toc189927281)

[2.2 Технологии 13](#_Toc189927285)

[2.1.a Игров двигател – Unity 13](#_Toc189927286)

[2.2.b Алгоритъм за оптимизиране на трафика - самообучение с утвърждение 14](#_Toc189927287)

[2.2.c Библиотека за самообучение с утвърждение - ML-Agents 14](#_Toc189927288)

[2.2.d Интегрирана среда за разработка – Visual Studio 15](#_Toc189927289)

[2.3 Теория на дипломната работа – подход и алгоритми 16](#_Toc189927290)

[2.3.a Алгоритъм за генериране на произволен път – WFC 16](#_Toc189927291)

[Трета глава 21](#_Toc189927292)

[3.1 … 21](#_Toc189927293)

[Четвърта глава 22](#_Toc189927294)

[4.1 … 22](#_Toc189927295)

[Заключение 23](#_Toc189927296)

[Литература 24](#_Toc189927297)

[Съдържание 25](#_Toc189927298)

1. В закона в различните държави продължителността на жълтата светлини обикновено е фиксирана, като тя често е порпорционална на максималната разрешена за пътя скорост. [↑](#footnote-ref-1)
2. Явление, при което няколко светофари координирано пропуснат една непрекъсната вълна от трафик през няколко кръстовища в една главна посока [↑](#footnote-ref-2)