Міністерство освіти і науки України
Український державний університет науки і технологій
«Факультет Комп'ютерних технологій і систем»
(назва факультету)

«Комп'ютерні інформаційні технології»
(повна назва кафедри)

Пояснювальна записка
до кваліфікаційної роботи
магістра
(ступінь вищої освіти)

на тему: Дослідження інтелектуального керування транспортним рухом на регульованих перехрестях.
за освітньою програмою Інженерія програмного забезпечення
зі спеціальності: 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

Виконав:студент групи: ПЗ2121
/ Микита КОСТЮК /
(підпис студента) (Ім'я ПРІЗВИЩЕ)

Керівник: / доц. Вадим ГОРЯЧКІН /
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Нормоконтролер: / доц.Світлана ВОЛКОВА /
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Консультанти:
Техніко-економічні розрахунки / доц.Микола ГНЕНИЙ /
(назва розділу) (підпис) (посада, Ім'я ПРІЗВИЩЕ)

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент
(підпис)

Дніпро – 2022 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Дніпропетровський національний університет залізничного транспорту імені академіка В. Лазаряна

Кафедра Комп'ютерні інформаційні технології

«ДО ЗАХИСТУ»

Завідувач кафедри

___ проф. Шинкаренко В. І.

ДИПЛОМНА РОБОТА

на здобуття ОС «магістр»

Галузь знань 12 Інформаційні технології
(шифр) (назва)

Спеціальність 121 Інженерія програмного забезпечення
(код) (повна назва)

Освітньо-професійна програма_____
(повна назва)

Тема «Алгоритм керування транспортним рухом на регульованих перехрестях»

Theme «Traffic management algorithm at regulated intersections»
(theme in English)

Керівник дипломної роботи _____ _____ _____
(посада) (підпис) (ПІБ)

Консультант розділу з БЖД _____ _____ _____
(посада) (підпис) (ПІБ)

Консультант економічного розділу _____ _____ _____
(посада) (підпис) (ПІБ)

Нормоконтролер _____ _____ _____
(посада) (підпис) (ПІБ)

Студент групи _____ _____ _____
(номер групи) (підпис) (ПІБ)

Student _____
(Family name)

Дніпро 2022

Дніпропетровський національний університет залізничного транспорту імені академіка В. Лазаряна

Факультет Технічна кібернетика кафедра Комп'ютерні інформаційні технології
Спеціальність Інженерія програмного забезпечення

«ЗАТВЕРДЖУЮ»
Завідувач кафедри
_____ проф. Шинкаренко В. І.
(підпис)
«___»_____2019 р.

ЗАВДАННЯ

до дипломної роботи на здобуття ОС _магістр____
(освітній ступінь)

студента групи _____ _____
(номер групи) (ПІБ)

1 Тема дипломної роботи: _____
_____

затверджена наказом по університету від «___» _____ 201__ р. № ____.

2 Термін подання студентом закінченої роботи_____

3 Вихідні дані до дипломної роботи _____
_____
_____
_____

4 Зміст пояснювальної записки (перелік питань до розробки) _____
_____
_____
_____

5 Перелік демонстраційного матеріалу _____
_____
_____
_____
_____

6. Консультанти (з назвами розділів):

Розділ	Консультант	Підпис, дата	
		завдання видав	завдання прийняв
Техніко-економічні розрахунки			
Охорона праці			

ТЕХНІЧНЕ ЗАВДАННЯ:

Вихідні дані на роботу	Теорія систем масового обслуговування; Література з проблем організації транспортних мереж.
Перелік підлягають дослідженню, проектуванню та розробці питань	Вивчення середовища розробки Visual Studio; Вивчення Правил дорожнього руху; Вивчення теорії масового обслуговування.
Перелік графічного матеріалу	Структура завдань теорії масового обслуговування; Схеми пішохідних переходів; Структурні схеми моделей транспортних вузлів; Цикл роботи світлофора;

КАЛЕНДАРНИЙ ПЛАН

№ пор.	Назва розділів дипломної роботи	Термін виконання розділів роботи	Примітка
1	Вступ		
2	Аналіз сучасного стану дослідження проблеми за		від 70 джерел

	науковими літературними джерелами		
3	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження		
4	Постановка задачі, технічне завдання		30%
5	Техніко-економічні показники		
6	Розробка інструментальних засобів дослідження		
7	Виконання досліджень 60%		
8	Оформлення тез доповідей		
9	Оформлення статті у фаховий журнал		
10	Оформлення пояснювальної записки		
11	Розробка демонстраційних матеріалів 100%		

Дата видачі завдання «___» _____ 2022р.

Керівник дипломної роботи _____ _____
(підпис) (ПІБ)

Завдання прийняв до виконання _____ _____
(підпис) (ПІБ)

ЗМІСТ

РЕФЕРАТ

Випускна кваліфікаційна робота складається з текстового документа обсягом 115 сторінок, 36 рисунків, 14 таблиць, 81 посилань, 1 додаток.

Ключові слова: моделювання, транспортний потік, транспортний вузол, транспортна система, світлофорна система керування.

Об'єктом дослідження є статистичні дані та їх аналіз.

Метою даної роботи є розробка моделі системи управління транспортним потоком.

У процесі дослідження використовується теорія системи масового обслуговування.

В результаті дослідження була створена найпростіша модель транспортного перехрестя, а на її основі – модель перехрестя зі світлофорами.

Модель може бути використана для оптимізації роботи системи перехрестя: створення різних режимів роботи світлофора в залежності від навантаження на перехрестя.

ABSTRACT

The graduation qualification work consists of a text document of 115 pages, 36 figures, 14 tables, 81 references, 1 appendix.

Key words: modeling, transport flow, transport node, transport system, traffic light control system.

The object of the research is statistical data and their analysis.

The purpose of this work is to develop a model of the traffic flow control system.

The theory of the mass service system is used in the research process.

As a result of the research, the simplest model of a traffic intersection was created, and based on it, a model of an intersection with traffic lights.

The model can be used to optimize the operation of the intersection system: creating different modes of operation of the traffic light depending on the load on the intersection.

ПЕРЕЛІК УМОВНИХ ПОЗНАК, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Транспортна система — це комплекс різних видів транспорту, способів зв’язку та транспортної інфраструктури, які взаємодіють і залежать одна від одної в процесі транспортування.

ВДМ – вулично-дорожня мережа – територія загального користування, призначена для забезпечення проїзду транспортних засобів і пішоходів, забезпечення транспортної та пішохідної доступності населених пунктів.

ТМ - Транспортна мережа - Сукупність вулиць, доріг, площ, дорожніх споруд, технічні умови яких придатні для руху рухомого складу автомобільного транспорту.

Моделювання - полягає в заміні об’єкта дослідження (оригіналу) його умовним зображенням або іншим об’єктом (моделлю) і дослідженні властивостей об’єкта-оригіналу шляхом дослідження властивостей моделі.

СМО - система масового обслуговування - це моделі системи, де заявки (запити) надходять випадковим чином ззовні або зсередини.

ВСТУП

У переліку критичних технологій України пріоритетними є безпека руху та управління транспортом [1].

За останні десятиліття великі українські міста вичерпали або майже вичерпали можливості функціонування транспортної мережі без її перевантаження і можливості подальшого розвитку. Це зумовлено такими факторами, як підвищення рівня оснащеності автомобільного населення, фізична та економічна неможливість розширення пропускної здатності транспортної мережі тощо.

Особливе значення має планування дорожніх мереж (SDN), організація дорожнього руху та оптимізація систем маршрутизації громадського транспорту. Вирішити ці проблеми без моделювання транспортної мережі неможливо [2].

Справжні переваги моделювання можна отримати лише за умови виконання двох умов [3]:

– модель забезпечує коректне відображення вихідних властивостей, істотних з точки зору досліджуваної операції [3];

– ця модель усуває властиві проблеми вимірювання реальних об'єктів [3].

Математичні моделі, які використовуються для аналізу транспортних мереж, різноманітні за категоріями завдань, які вони вирішують, що унеможливлює конкретну вичерпну класифікацію [4]. Ця робота присвячена імітаційному моделюванню та імітації руху на перехресті.

Імітаційне моделювання відповідає на питання, наскільки детально відбуватиметься рух, якщо відомі середній розмір потоку, напрямок руху та пріоритет перетину перехрестя.

РОЗДІЛ 1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Постановка проблеми

Транспорт є однією з ключових систем міського організму, і його значення можна порівняти з кровопостачанням. Саме транспорт дозволяє містам повною мірою використовувати свої зв’язки, комунікації та допоміжні функції. Зусилля щодо систематизації та поширення відповідних знань набули ще більшої актуальності, оскільки питання транспорту торкаються чи не кожного мешканця міста.

Для управління рухом на міських транспортних мережах широко використовуються системи керування, алгоритми яких базуються на моделях транспортних потоків. Вимоги до точності та складності моделі надзвичайно високі. Можна сказати, що на найпростішому перехресті транспортний засіб може рухатися в 12 напрямках. Для ділянки дорожньої мережі з 10 такими перехрестями мова йде про 120 напрямків, і якщо інтенсивність руху постійно змінюється в часі та просторі, затримки в кожному напрямку повинні бути мінімізовані. Крім того, без моделювання дорожнього руху неможливо планувати нове та модернізоване будівництво існуючих об’єктів дорожнього руху, об’єктів житлового та комерційного будівництва, варіанти організації дорожнього руху, дії в надзвичайних ситуаціях

При цьому вирішується багато інших практичних завдань [2].

В даний час в індустріально розвинутих країнах проблема заторів на дорогах стає все більш помітною, особливо в мережі доріг великих міст, де зосереджена більшість приватних автомобілів, а також на основних магістралях і магістралях.

Транспортні коридори, на які припадає значна частка вантажних і пасажирських перевезень.

На сьогоднішній день у світі накопичено позитивний досвід реалізації заходів щодо вирішення цієї проблеми, в тому числі спрямованих на:

- збільшення максимальної пропускної здатності вулично-дорожньої мережі (будівництво та реконструкція дорожніх об'єктів);
- підвищення ефективності використання пропускної спроможності вулично-дорожньої мережі (удосконалення організації перевезень, у тому числі прикладних);
- регулювання кількості та структури транспортного попиту (введення різноманітних обмежень на рух і стоянку транспортних засобів, зниження попиту економіки та населення на транспорт за рахунок заходів у таких сферах, як регіональне планування).

Методологічною основою проведення такої діяльності та прийняття науково обґрунтованих рішень щодо її реалізації є математична модель транспортної системи. Такі моделі дозволяють:

– оцінити ефективність запланованої діяльності, включаючи оперативно-економічні показники;
– визначити можливі негативні наслідки його реалізації;
– розробити науково обґрунтований план реалізації.

Зростає роль математичного моделювання як складової частини методичного забезпечення розробки та впровадження будь-яких заходів контролю в транспортній галузі. У Великобританії, Данії, Швеції та інших країнах були розроблені моделі національних транспортних систем для використання при формуванні транспортних планів і програм на всіх рівнях управління. На сучасному ринку програмного забезпечення існує безліч додатків для розробки різних транспортних моделей.

З подальшим розвитком української транспортної системи, розширенням і вдосконаленням зв'язку дорожньої мережі, підвищенням ролі мультимодального транспорту, впровадженням інтелектуальних транспортних систем, проблема перевантаженості дорожньої мережі в основних транспортних коридорах і зміни в існуючих великих містах З більшою гостротою зростатиме потреба у використанні моделей трафіку [3].

Основною метою цієї випускної кваліфікаційної роботи є розробка імітаційної моделі транспортних перехресть, частини мережі руху, результати якої можуть бути використані для оптимізації регулювання, особливо системи світлофорного регулювання.

Світовий досвід показує, що повністю ліквідувати затори неможливо. Проте закономірним є запитання: яке ж навантаження насправді несе так звана «перевантажена» дорога? Аналіз показав, що фактична інтенсивність руху сучасних міських доріг значно коливається: від 300 до 1200 транспортних засобів на годину на смугу, а в деяких випадках значно нижча за норму. Тому в багатьох великих містах не дивно, що цю проблему вирішують впровадженням так званих інтелектуальних транспортних систем (ІТС) [5].

1.1.1 Характеристика побудови програмно-технічних комплексів управління дорожнім рухом

Концепція інтелектуальних транспортних систем застосовується як до управління транспортним потоком у містах, так і до контролю за умовами руху на магістралях. Існують чотири основні завдання використання ІТС-технології [6]:

1. Оптимізація містобудівних рішень з урахуванням специфіки організації перевезень на етапі технічно-економічне обгрунтування (ТЕО);

2. Забезпечення максимальної пропускної здатності існуючої вулично-дорожньої мережі міста;

3. Пріоритетний рух маршрутних транспортних засобів та автомобілів аварійної служби;

4. Стоянка;

5. Зменшити екологічне навантаження міста.

Звичайно, програмне забезпечення ІТС і підтримка методів у всіх цих областях означає взаємодію всіх завдань. Проте кожна така система повинна будуватися та експлуатуватися незалежно, оскільки спроби побудувати

комплексну систему контролю за дорожнім рухом у місті з часом втрачатимуть керованість через спотворення пріоритетів [6].

Наприклад, для систем, які контролюють міський громадський транспорт, слід оцінити швидкість розподілу по лініях і охоплення цих ліній у межах міста, для систем управління дорожнім рухом слід оцінити середню швидкість і середню швидкість. Витрата палива на вуличну мережу. Зрозуміло, що ці цифри непорівнянні. Тому, якщо ці сервіси об’єднати і надати єдиний комплексний звіт по місту, ймовірно, користувачі не зможуть обґрунтовано оцінити якість кожного з них. Це призведе до неефективної роботи транспортної інфраструктури та системи організації руху [6].

1.1.2 Концепція побудови підсистеми керування дорожнім рухом

Завданням системи автоматичного керування дорожнім рухом є підвищення стандартів якості системи керування дорожнім рухом окремо або в комплексі [6]:

- ємність критичних компонентів;
- середня швидкість в основному напрямку;
- середня витрата палива.

Практика показала, що вдале механічне перенесення технологій і якісного обладнання з одного міста в інше призводить лише до створення дорогих і неефективних методів, які не контролюють ситуацію, а дозволяють лише спостерігати за нею [6].

Ця особливість накладає деякі обмеження на технічні методи покращення показників якості УДС. Таких методів шість [6]:

- Оптимізувати режим роботи світлофорів;
- Організація міських швидкісних доріг для контролю доступу;
- Інформувати учасників дорожнього руху про поточні умови руху;
- Удосконалити систему реагування на інциденти;

– Оптимізувати підключення перехрестя;

– Обмежити проїзд автотранспорту на основні ділянки (елементи) дорожньо-транспортної мережі та запровадити платний проїзд на найбільш завантажених міських магістралях.

Найбільш ефективним і найскладнішим у реалізації з усіх методів є оптимізація режимів роботи світлофора. Тому необхідно створити злагоджено працюючу світлофорну мережу, характеристики якої будуть задовольняти наступним двом параметрам [7]:

– Перш за все, координаційна робота світлофорних об’єктів повинна бути організована відповідно до особливостей просторового співвідношення та особливостей вимог до транспортного потоку.

– По-друге, режим роботи світлофорів повинен визначатися на основі достовірної статистики параметрів транспортного потоку та оптимізуватись на основі завантаження дорожньої мережі.

1.1.3 Сфери застосування моделі транспортної системи

Існуючі технології аналізу та обробки статистичної інформації дають змогу виявляти закономірності та аналізувати процеси в складних організаційно-технологічних системах, у тому числі транспортно-дорожнього комплексів [1]. Для цього створюються інформаційні системи, системи підтримки прийняття рішень, імітаційного моделювання, статистичного аналізу та прогнозування. Однак результати такого дослідження є правильними лише в тому випадку, якщо інформація, на якій базується аналіз, є повною, добре структурованою та формалізованою. Оскільки управління транспортними потоками є областю, в якій важко або неможливо проводити практичні експерименти, у багатьох випадках імітаційне моделювання стає єдиним засобом ефективного прийняття рішень у цій галузі [1].

Розробка та модельний експеримент імітаційної моделі міської дорожньої мережі включає три етапи [1]:

1. Розробка імітаційної моделі дорожньої мережі;

2. Проведення комп’ютерних експериментів з моделлю – вибрати вузол з найбільшим навантаженням;

3. Формулювання рекомендацій на основі аналізу результатів експерименту, щодо перерозподілу транспортного потоку.

Оскільки транспортна інфраструктура українських мегаполісів не відповідає сучасним стандартам і підвищеному рівню автомобілізації, створення таких моделей є важливим для планування транспортних мереж. В даний час в Україні є достатня кількість молодих міст, які спроектовані з урахуванням можливого зростання кількості населення і пасажиропотоку. Проте навіть ці заходи (широкі бульвари, прямокутне планування, винесення трамвайних колій із зон руху) не можуть усунути проблему завантаженості доріг, особливо в так звані «години пік» [1], але використання розробленого підходу можуть зменшити навантаження

1.1.4 Перехрестя є найскладнішими для управління елементами транспортної мережі

Найбільш складним в управлінні елементом транспортної мережі є перехрестя.

Перехрестя – це перехрестя, перехрестя або розгалуження доріг на одному рівні, обмежені уявними лініями, що з’єднують витоки кривизни смуги, найдальші від центру перехрестя, відповідно. [Чотири]

При вивченні швидкості руху перехрестя вулично-дорожньої мережі за рухом транспортних засобів поділяють на дві категорії:

– Регульовані;

– Нерегульовані.

Регульовані перехрестя поділяються на:

– керується світлофором;

– регулюється контролером.

Нерегульовані перехрестя поділяються на:

– перетинання рівних доріг;

– перехід нерівних доріг;

– з урахуванням знака пріоритету.

Існує чотири типи перехресть:

– хрестоподібні;

– Т-подібні;

– П-подібні;

– кругові рухи.

На швидкість руху на перехрестях впливає кілька факторів:

– тривалість циклу регулювання світлофора;

– змінити напрямок руху транспортного засобу;

– взаємодія транспортного засобу з конфліктними транспортними потоками;

– дорожньо-транспортна пригода.

Умови руху транспортних засобів на перехресті можуть характеризувати кілька факторів:

– кількість смуг на основних дорогах;

– інтенсивність руху на основних дорогах;

– кількість смуг руху на перехресті;

– інтенсивність руху на перехрестях.

Кількість смуг на основних дорогах і перехрестях визначає геометрію перехрестя, що впливає на маневреність під час руху по перехрестю. те саме для цього

Можливість впливає на інтенсивність руху на перехрестях. [5]

1.2 Теорія масового обслуговування

Системи масового обслуговування можна знайти практично скрізь, де є або може бути черга. На Заході метод масового обслуговування навіть називають теорією масового обслуговування. Оскільки черги, як правило, є небажаним явищем, природно запропонувати збільшити потужність (пропускну здатність) обслуговуючого пристрою для його усунення. Але оскільки заявки надходять нерегулярно, то зі збільшенням потужності буде збільшуватися і частка простою обладнання, що теж небажано. Таким чином, з економічної точки зору завдання масового обслуговування зводиться до пошуку компромісу між двома суперечливими вимогами - вимогою усунення черги і вимогою повного завантаження обладнання. Втрати від появи черг пов'язані з втраченим часом покупців в магазинах, простоєм автомобілістів на заправках, мостах і перехрестях, очікуванням розвантаження і завантаження кораблів, вартістю палива для літаків, що очікують на посадках в аеропорту. Простій означає марну витрату грошей чи часу, вкладених у нього, які можуть бути корисними в іншому місці.

1.2.1 Основні поняття та термінологія теорії масового обслуговування

Метою теорії масового обслуговування є розробка математичних методів для аналізу процесів масового обслуговування та оцінки якості систем обслуговування. [9].

Предметом дослідження теорії масового обслуговування є система масового обслуговування (СМО) [10].

Метою теорії масового обслуговування є надання пропозицій щодо побудови СМО, раціональної організації роботи та регулярізації потоку запитів для забезпечення високої ефективності СМО [10].

Для досягнення поставленої мети поставлена задача теорії масового обслуговування, яка включає встановлення залежності ефективності роботи СМО від її організації (параметрів): характеру потоку заявок, кількості каналів, продуктивності та правил СМО. [9].

Усі завдання в черзі мають схожу структуру, яку можна описати схемою на рисунку 1.1:

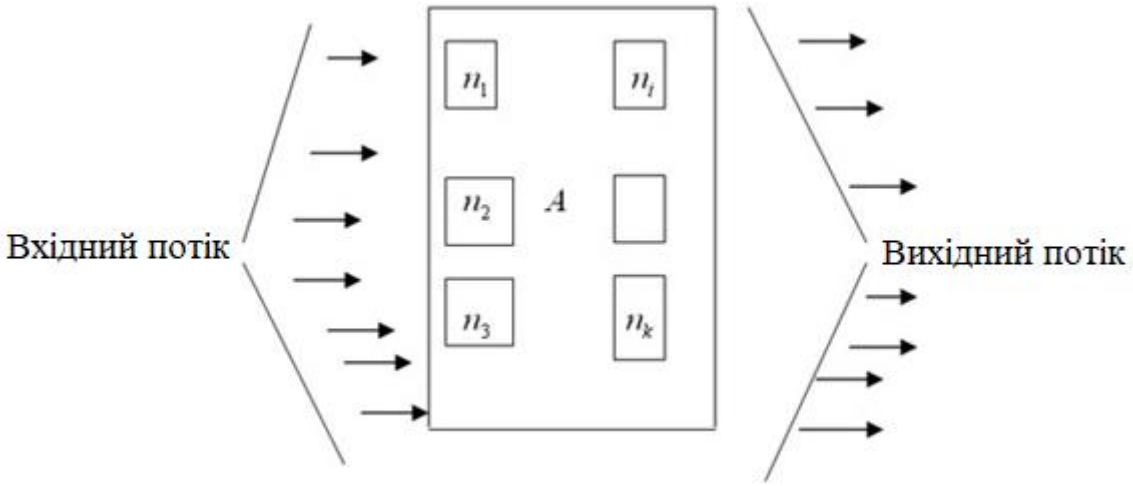


Рисунок 1.1 – Структура проблеми в теорії масового обслуговування

Елементами цієї структури є станції обслуговування (пристрої, сервери, касири, термінали, банкомати тощо), які обслуговують вхідні вимоги (додатки), які потім формують подію часового ряду, яка називається потоком попиту .

Потік запитів, що очікують на обслуговування, називається вхідним потоком, а потік запитів, що залишають систему обслуговування, називається вихідним потоком [10].

Існує кілька класифікацій СМО. Перша класифікація заснована на наявності черги:

– несправні (запити надходять, коли всі канали зайняті, виходять із системи та більше не обслуговуються);

– використовуйте черги (запити, які надходять, коли всі канали зайняті, не йдуть, а чекають часу обслуговування).

СМО з чергами, в свою чергу, поділяються на різні типи залежно від того, як організовані черги (обмежено чи ні). Обмеження можуть бути накладені на довжину черги, час очікування, дисципліну обслуговування.

Крім того, СМО ділиться на два види:

– відкрита СМО (характеристики трафіку програми не залежать від того, скільки каналів зайнято);

– закрита СМО.

1.2.2 Марковський випадковий процес

Припустимо, що існує деяка система S, стан якої змінюється з часом (під системою S можна розуміти технічне обладнання, виробничий процес, комп'ютер, інформаційну мережу тощо). Якщо стан системи S змінюється з часом випадковим, непередбачуваним чином, кажуть, що в системі відбувається випадковий процес [11].

Стохастичний процес, що працює в системі S, називається процесом Маркова (або «процесом без наслідків»), якщо він має таку властивість: для кожного моменту часу t0 ймовірність того, що система перебуватиме в будь-якому майбутньому стані (коли t> t0) залежить тільки від його поточного стану (при t=t0), а не від того, коли і як система потрапила в цей стан (тобто як процес розвивався в минулому). [одинадцять]

На практиці бувають випадки, коли перехід системи із стану в стан відбувається не у фіксований час, а у випадковий час, який неможливо визначити заздалегідь - переходи можуть бути реалізовані в будь-який час [11].

Для опису такого процесу можна застосувати схему стохастичного процесу Маркова з дискретними станами та безперервним часом. Ці типи процесів називають безперервними ланцюгами Маркова. Неперервний ланцюг Маркова (Марковський процес) — це процес, який виконується при $0 \le t_1 \le t_2 \le .... \le t_{n+1}$:

{ ( ) | } { ( ) | }

Тут ми розглядаємо деякі дискретні стани: S_1, S_2...S_n. Перехід системи зі стану в стан може відбуватися у випадковий момент часу [11].

1.3 Принципи сучасного організації дорожнього руху.

Транспортне навантаження на основні дороги міста з кожним роком зростає, внаслідок чого швидкість транспортного потоку значно зменшується, внаслідок чого виникають затори. Виникнення заторів навіть при певній пропускній здатності можна пояснити неоднозначною роботою системи управління світлофорами, що призводить до збільшення довжини транспортних черг і збільшення часу обслуговування.

Пріоритетним напрямом у сфері організації дорожнього руху є питання організації транспортного потоку на магістральних магістралях в умовах вузлових заторів.

У зарубіжній практиці при високому ступені автомобілізації особливий акцент робиться на формулюванні нових принципів і нових норм управління магістральними вулицями.

Головна вулиця під сучасною системою управління розуміється як система управління, яка проявляється у двох напрямках:

1. Оптимізувати режим роботи світлофорів;

2. Контроль доступу: односторонній рух, без розвороту, платні дороги.

1.4 Моделювання

Початок математичного моделювання транспортного потоку можна простежити до 1930-х років, коли автомобілізація, особливо в Сполучених Штатах у Північній Америці, досягла такого рівня, що почали розглядатися заходи щодо збільшення пропускної здатності доріг та усунення заторів. Вважається, що Б.Д. першим розробив і застосував математичні моделі. Гріншилдс [4, 5], які досліджували пропускну спроможність доріг та фактори, що на неї впливають (наприклад, щільність руху). Він є автором фазових діаграм, пов'язаних з величинами потоку та густини. Проте в перші роки після народження автомобіля серйозно вивчалися різні аспекти дорожнього руху, зокрема монографія російського професора Г. Д. Дубеліра [6], який у 1912 році сформулював свою роботу на основі математичного моделювання та оцінити пропускну здатність доріг і перехресть.

Основним завданням для розробки моделювання транспортних потоків є аналіз пропускної здатності магістралей і перехресть. Під пропускною спроможністю розуміється максимально можлива кількість автомобілів, які можуть проїхати відрізок дороги за одиницю часу. У фаховій літературі використовуються поняття пропускної здатності теоретична, номінальна, ефективна, власна, фактична, фактична та ін. В даний час пропускна здатність є найважливішим критерієм оцінки якості роботи ліній зв'язку.

У 1955 році Лайтхілл і Вітхем [7, 8] запропонували першу макроскопічну модель, що розглядає транспортний потік з точки зору механіки континууму. Вони показують, що методи, що описують безперервні транспортні процеси, можна використовувати для моделювання заторів.

Лайтхілл і Вітем представили опис руху потоку на основі рівняння безперервності, припускаючи, що швидкість залежить тільки від щільності, тобто існує миттєва адаптація. Пригожин і Герман розробили динамічну теорію транспортного потоку [9]. Вони визначають модель Лайтхелла-Вітема як

окремий випадок кінетичної теорії. Теорія динаміки описує багато явищ, які відбуваються під час руху транспортних потоків, однак, ймовірно, через те, що математичне моделювання досить трудомістке, теорія не була розроблена до останнього часу [10].

Натомість у 1979 році Пейн замінив припущення про миттєву адаптацію в теорії Лайтхелла-Вітема інерційним рівнянням, подібним до рівняння Нав'є-Стокса [11]. У 1984 році Кішне додав термін в'язкість і почав використовувати нелінійні динамічні методи для аналізу рівнянь [12-15].

Тим часом Муша та Хігучі запропонували рівняння Бюргерса як модель для опису транспортного потоку та забезпечили автоматичне вимірювання даних про обсяг трафіку [16].

Ф. Хейт [8] вперше виділив математичне дослідження транспортного потоку в окрему частину прикладної математики.

Протягом 1960-х і 1970-х років відновився інтерес до вивчення транспортних систем. Цей інтерес в основному виражається у фінансуванні численних контрактів, залучення провідних вчених - фахівців з математики, фізики, процесів управління, таких як Нобелівський лауреат І. Пригожин, фахівець з автоматичного управління М. Атанс, автор основ статистики Л. Брейман. У нашій країні рух автотранспорту було пов'язано в кінці 1970-х років з підготовкою до московської Олімпіади 1980 року. Результати цих досліджень неодноразово доповідалися на науковому симпозіумі І.І. Звєрєва механіко-математичного факультету МДУ. М. В. Ломоносова [17].

На даний момент існує велика кількість літературних та інтернет-ресурсів, присвячених вивченню автомобільного транспортного потоку, в тому числі велика кількість теоретичних робіт з його математичного моделювання. Необхідно відзначити наукові видання, присвячені вивченню руху транспортних засобів та динаміки транспортних потоків, такі як Traffic Research, Traffic Science, Mathematical Computer Simulation, Operations Research, Automation, Physical Review E, Physical Reports тощо. Нове – це не просто звіти, регулярно

публікуються статті та вичерпні огляди. І підручники. На жаль, мова йде здебільшого про розвинені країни, де протягом десятиліть транспортні проблеми найбільше впливають на життя людей.

Наприкінці 1980-х і на початку 1990-х років дослідження транспортних систем було піднесено до рівня питань національної безпеки в Сполучених Штатах. Лос-Аламосська національна лабораторія (LANL) є одним із найбільших ядерних дослідницьких центрів у США, і її найкраще «фізичне мислення» та комп'ютерні засоби залучені до вирішення завдань у цій галузі [18].

Моделювання — це метод розв'язування задач, при якому досліджувана система замінюється більш простим об'єктом, який називається моделлю, що описує реальну систему.

Він використовується в ситуаціях, коли експерименти на реальній системі неможливі або недоцільні, наприклад, у транспортних ситуаціях.

Розрізняють фізичне і математичне моделювання.

При використанні фізичного моделювання активним об'єктом є фізична подібність досліджуваної системи.

При використанні математичного моделювання поведінка системи описується за допомогою формул. Імітаційна модель — це спеціальна математична модель.

Імітаційні моделі — це комп'ютерні програми, які описують структуру та поведінку реальних систем у часі.

Імітаційні моделі дозволяють отримати детальну статистику щодо різних аспектів функціональності системи на основі вхідних даних.

Імітаційне моделювання — це розробка комп'ютерної моделі та налаштування для експериментування з нею. Зрештою, метою моделювання є прийняття обґрунтованих і зручних управлінських рішень.

Переваги використання імітаційних моделей:

– вартість (вартість використання імітаційної моделі включає лише вартість програмного забезпечення та вартість консультаційних послуг);

– час (імітаційні моделі дозволяють визначити ефективність за хвилини);

– повторюваність (використовуючи імітаційну модель, можна проводити нескінченну кількість експериментів з різними параметрами, щоб визначити найкращий вибір);

– точність (імітаційне моделювання може описувати структуру системи та її процеси природним чином без використання формул і строгих математичних залежностей);

– наочність (імітаційні моделі здатні візуалізувати процес роботи системи в часі, схематично задати її структуру та відобразити результати графічно, що дозволяє візуалізувати отримане рішення та передати закладені в ньому ідеї);

– універсальність (імітаційне моделювання дозволяє розв’язувати проблеми з будь-якої області, і в кожній області модель імітує, відтворює реальне життя та дозволяє широко експериментувати, не впливаючи на реальні об’єкти).

Існує три види імітаційного моделювання:

– агентне моделювання;
– моделювання дискретних подій;
– системна динаміка.

Агентне моделювання — це техніка імітаційного моделювання, яка досліджує поведінку децентралізованих агентів і те, як ця поведінка визначає поведінку всієї системи. При розробці агентної моделі інженер вводить параметри агента (агент - особа, компанія, проект, транспортний засіб, місто тощо), визначає їх поведінку, поміщає в певне середовище, встановлює можливі зв'язки, а потім запускає симуляцію. Індивідуальна поведінка кожного агента формує глобальну поведінку модельованої системи.

Дискретне подієве моделювання – це метод побудови імітаційної моделі, який пропонує представити реальні дії та процеси, що відбуваються у світі, у вигляді серії незалежних значущих моментів-подій.

Моделювання дискретних подій використовується для побудови моделей, які відображають еволюцію системи, коли стан змінної змінюється негайно в певний момент часу.

Структура моделювання дискретних подій включає такі компоненти: сутності, дії та події, ресурси, глобальні змінні, генератори випадкових чисел, збирачі статистики.

Сутності — це динамічні об’єкти, які створюються, переміщуються та часто видаляються в деякій частині системи, що моделюється. Виконання моделі завершується, коли в системі немає активних об’єктів. Кожна сутність має унікальний набір властивостей. Значення властивостей можуть змінюватися в міру просування сутності та впливати на шлях сутності в моделі.

Дії - це логіка процесу та модель, яка відбувається під час моделювання.

Подія - це ситуація, яка змінює стан системи в певний момент часу. Події виникають, коли над сутностями виконуються операції.

Ресурс — це все, що має обмежену ємність.

Глобальні змінні - це змінні, які легко доступні будь-де в моделі.

Збирач статистики — це частина моделі, яка збирає статистику про деякий стан системи, значення глобальної змінної або властивість об’єкта.

Системна динаміка – це підхід до імітаційного моделювання, який дозволяє використовувати його методи та інструменти для розуміння структури та динаміки складних систем. Він використовується для довгострокових стратегічних моделей і використовує високий рівень абстракції. Люди, продукти, події та інші дискретні елементи представлені не як окремі елементи в моделі системної динаміки, а як єдине ціле. Якщо окремі елементи моделі важливі, то моделювання на основі агентів або дискретних подій найкраще використовувати для всієї обробки або її частини. [6]

1.5 Етапи розробки імітаційних моделей

Імітаційне моделювання складається з двох основних етапів:

- створювати моделі;
- проаналізувати результати, отримані за допомогою моделі, щоб прийняти рішення.

Після детального розгляду побудову імітаційної моделі можна розділити на декілька етапів:

- розробник сам визначає цілі та завдання, які вирішуються за допомогою моделі;
- визначити, які процеси слід ідентифікувати та відобразити в моделі, від яких процесів слід абстрагуватися, а також які особливості цих процесів слід, а які не слід враховувати;
- визначити зв’язок між змінними та параметрами моделі, які повинні бути відображені в моделі.
- будувати програми, які обчислюють зміни змінних і характеристик моделі з часом;
- анімоване зображення процесу розробки;
- калібрування моделі - збір даних і вимірювання тих властивостей в реальній системі, які повинні бути введені в модель у вигляді значень параметрів і розподілів випадкових величин;
- валідація моделі - перевірка правильності моделі, перевірка виходу моделі в декількох тестових режимах, де поведінкові характеристики реальної системи відомі або очевидні;
- експериментуйте – виконуйте модель при різних значеннях її існуючих параметрів і спостерігайте за її поведінкою, записуючи поведінкові характеристики.

Перераховані вище етапи імітаційного дослідження рідко виконуються в строго встановленій послідовності. У ході симуляційного дослідження можуть

виникнути помилки моделі, неправильні припущення, які пізніше доведеться відкинути, переформулювання цілей дослідження, повторна оцінка моделей і реструктуризація. Цей ітеративний процес дозволяє розробити імітаційну модель, яка правильно оцінює альтернативи та полегшує процес прийняття рішень.

1.6 Огляд існуючих методів рішення задачі

Методи та алгоритми моделювання трафіку, описані в попередніх розділах, на даний момент реалізовані у великій кількості спеціалізованих програмних комплексів. Програмне забезпечення, яке використовується для розробки різних моделей транспорту, швидко розвивається завдяки постійному зростанню обчислювальної потужності. Розглянемо найбільш важливі та практичні програмні продукти, призначені для моделювання транспортних систем, які використовуються у світовій практиці [3].

Як приклад розповсюдження та використання програмних продуктів моделювання трафіку наведено список програмного забезпечення, рекомендованого Міністерством транспорту США (FHWA).

Інструменти планування ескізу: кращі рішення, HDM (проектування та управління магістралями), IDAS (система аналізу розгортання ITS): IMPACTS, icroBENCOST, uickZone, CRITS (скринінг ITS), методи ескізу, SMITE, SPASM (модель електронної таблиці аналізу планування ескізу),

STEAM (модель аналізу ефективності наземного транспорту), TEAPAC/SITE, TrafikPlan, TransDec, Generation Trip, Turbo Architecture.

Інструменти моделювання попиту на подорожі: модель b-Node, CUBE/MINUTP, CUBE/TP+/Viper, CUBE/TRANPLAN (Транспортне планування), CUBE/TRIPS (Транспортна система планування покращення), EMME/2™, IDAS, MicroTRIMS, QRS II (Rapid Response System II), SATURN

(моделювання та розподіл трафіку в міських дорожніх мережах), Tmodel, TransCAD®, TRANSIMS (симуляційна система аналізу трафіку), VISUM.

Аналітичні/детерміновані інструменти (підхід HCM): 5-Leg Signal Intersection Capacity, aaSIDRA, ARCADY (Round Island Capacity and Delay Assessment), ARTPLAN (Arterial Road Planning), CATS (Computer Assisted Transportation Software), CCG (Canada Capacity Guide) / Calc2: CINCH, CIRCAP (пропускна здатність по колу), DELAYE (підвищення затримки), dQUEUE-TOLLSIM, FAZWEAVE, FREEPLAN (планування магістралі): FREWAY

(Програма розрахунку затримки на магістралі), FRIOP, Загальна модель черги, Узагальнена річна середньодобова шкала обслуговування, Узагальнена шкала спрямованого обслуговування в годину пік, GradeDec 2000: HCM/Cinema©, HCS (Програмне забезпечення пропускної здатності магістралі) 2000, HiCAP™ (Пакет аналізу пропускної здатності магістралі), HIGHPLAN (Планування автомагістралей), Аналіз безпеки на дорогах, ICU, IQPAC (Пакет інтегрованого аналізу черги), Процедура перевірки сигналу лівого повороту/фази, NCAP (Пакет аналізу пропускної здатності перехресть), PICADY (Пропускна здатність і затримка пріоритетних перехресть), PROGO, Якість /Level of Service Handbook, RoadRunner, SIG/Cinema©, SIPA, SPANWIRE, SPARKS (Smart Parking Analysis), Synchro, TEAPAC/NOSTOP, TEAPAC/SIGNAL2000, TEAPAC/

WARRANTS, TGAP (Програма аналізу прогалин трафіку), TIMACS, Traffic Engineer Toolbox, Traffic Noise Modeling, TRAFFIX™, TSDWIN™, TS/PP-Draft, WEST, WHICH, WinWarrants.

Інструменти оптимізації трафіку: PASSER™ II-02, PASSER III-98, PASSER IV-96, PROGO, SOAP84, Synchro, TEAPAC/NOSTOP, TEAPAC/SIGNAL2000, TEAPAC/WARRANTS, TRANSYT-7F, TSDWIN, TS/PP-Draft.

Моделі макросимуляції: BTS (Bottleneck Traffic Simulator), FREQ12, KRONOS, METACOR/METANET, NETCELL, PASSER II-02, PASSER III-98,

PASSER IV-96, SATURN, TRAF-CORFLO (Corridor Flow), TRANSYT-7F, Бачення.

Мезорімітаційні моделі: CONTRAM (модель постійного розподілу трафіку), DYNAMIT-P, DYNAMIT-X, DYNASMART-P, DYNASMART-X, MesoTS.

Моделі мікросимуляції: AIMSUN2 (розширений інтерактивний мікросимулятор для міських і позаміських мереж), ANATOLL, AUTOBAHN, CASIMIR, CORSIM/TSIS (транспортне програмне забезпечення).

SIMNET, SimTraffic, SISTM (Стратегія моделювання дорожнього руху), SITRA B+, SITRAS, SmartPATH, TEXAS (Модель дорожнього руху на перехрестях Техасу), TRANSIMS, TRARR, TWOPAS, VISSIM, WATSim©.

Інструменти інтеграції аналізу трафіку: AAPEX (Arterial Analysis Package Execution), ITRAF, PROGO, UNITES.

За даними огляду інтернет-ресурсів, програмні продукти, призначені для моделювання трафіку на макрорівні, пройшли повну перевірку в розвинутих країнах світу, включаючи наступні програмні пакети: TransCad® (Caliper Corp., США); EMME/2™ (Université де Монреаль); TRIPS (MVA, Великобританія); CUBE; SATURN (Університет Лідса, Великобританія, лише трафік); VISSUM (частина пакету PTV Vision, PTV AG, Карлсруе, Німеччина).

Ринок пакетів моделювання пропонує досить широкий спектр інструментів, призначених для моделювання транспортного потоку на мікроскопічному рівні. Серед них, наприклад, можна відзначити пакети програм, які широко використовуються в країнах Європи: AIMSUN2; Dracula; Parameter; System; Visim [18].

Виходячи з припущення, що конкурентоспроможність пакета прикладних програм (APP) можна охарактеризувати ступенем його використання споживачами, можна спробувати визначити, які PAP користуються найбільшим попитом у експертів у галузі моделювання трафіку. На основі ресурсів наукової електронної бібліотеки eLIBRARY.RU проведено статистичний аналіз публікацій із згадуванням окремих застосувань [33]. Результати статистичного

аналізу публікацій, пов’язаних з мікроскопічним моделюванням, наведено на рисунку 1.2, Макроскопічне (мезо) моделювання – 1.3.

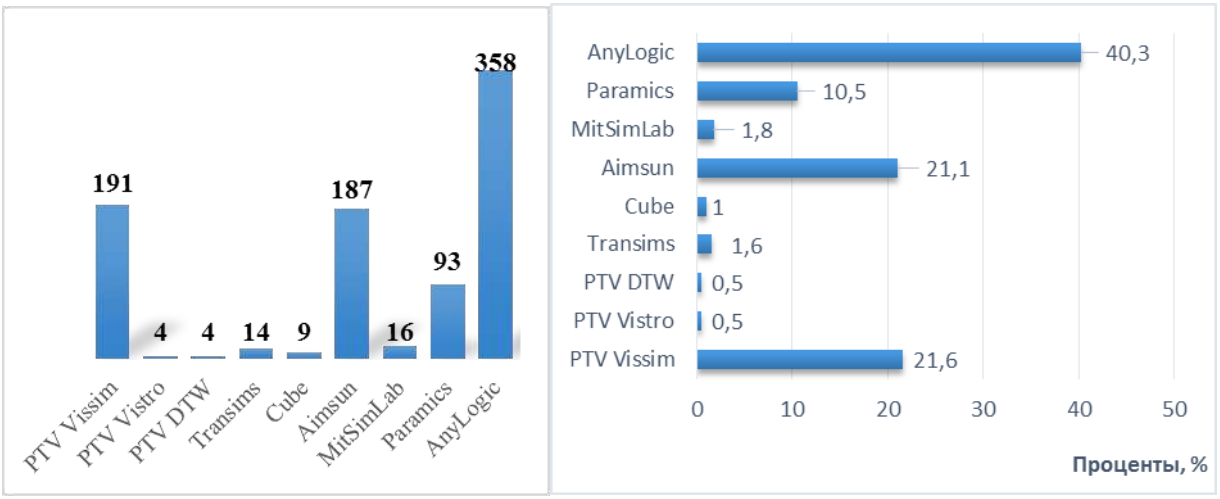


Рисунок 1.2 – Кількісні та відсоткові оцінки програмних продуктів моделювання трафіку на мікрорівні

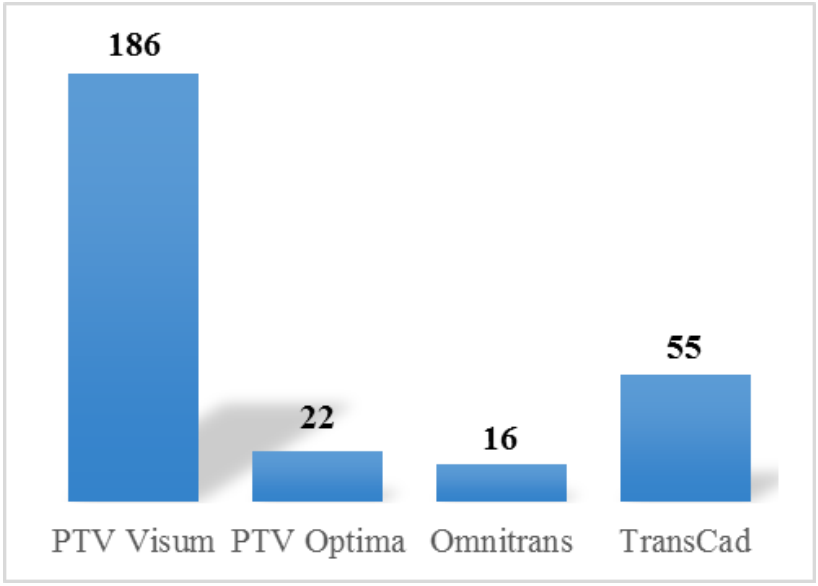


Рисунок 1.3 - Кількісні та відсоткові оцінки програмних продуктів моделювання трафіку на макро (мезорівні)

Як видно з рисунків 1.2 та 1.3, найпопулярнішими програмними продуктами для моделювання транспортного потоку на мікрорівні є такі програми: AnyLogic, PTV Vissim та Aimsun, на макро (мезо) рівні: PTV Visum, TransCad і PTV Optima.

1.7 Постановка цілей

Першим кроком у розробці моделі є сегментація основних транспортних вузлів (пішохідних переходів, Т-подібних перехресть, Х-подібних перехресть) та моделювання їх роботи.

На другому етапі необхідно додати до вузлів світлофорне регулювання: змоделювати транспортні вузли з різними навантаженнями та запропонувати рішення для оптимізації режимів їх роботи.

РОЗДІЛ 2. МОДЕЛЮВАННЯ РІЗНИХ ВИПАДКІВ В ТРАНСПОРТНОМУ РУСІ

2.1 Моделювання вузлів простої транспортної мережі

У моделюванні трафіку історично існувало два основних підходи - детермінований і імовірнісний (стохастичний). Детерміновані моделі базуються на функціональних зв’язках між різними показниками, такими як швидкість і відстань між автомобілями в потоці. У стохастичних моделях транспортний потік розглядається як імовірнісний процес. Усі моделі транспортних потоків можна розділити на три категорії: імітаційні моделі, моделі слідування лідера та ймовірнісні моделі.

В імітаційній моделі рух транспортного засобу порівнюється з якимось фізичним потоком (гідродинамічна та газодинамічна моделі). Такі моделі називаються макромоделями. У моделі слідування за лідером необхідно припустити, що існує зв'язок між рухом слідкуючого та провідного автомобіля. У міру розвитку теорії моделі групи враховували час реакції водія, вивчали рух на багатосмугових дорогах і вивчали стабільність руху. Такі моделі називаються мікромоделями. До мікроскопічних моделей належать також моделі, побудовані на клітинних автоматах.

У моделі клітинних автоматів дорога поділена на клітинки, і кожна клітинка може містити автомобіль або бути порожньою. У імовірнісних моделях транспортний потік розглядається як результат взаємодії транспортних засобів на елементах транспортної мережі. Через жорсткий характер мережевих обмежень і високу інтенсивність руху в транспортному потоці існують чіткі закономірності у формуванні черг, інтервалів, навантажень уздовж смуг руху тощо. Ці закономірності мають випадковий характер.

Робота з моделювання пішохідного переходу на односторонньому сегменті мережі руху (рис. 2.1) необхідна для аналізу навантаження на мережу та визначення умов перевантаження мережі.

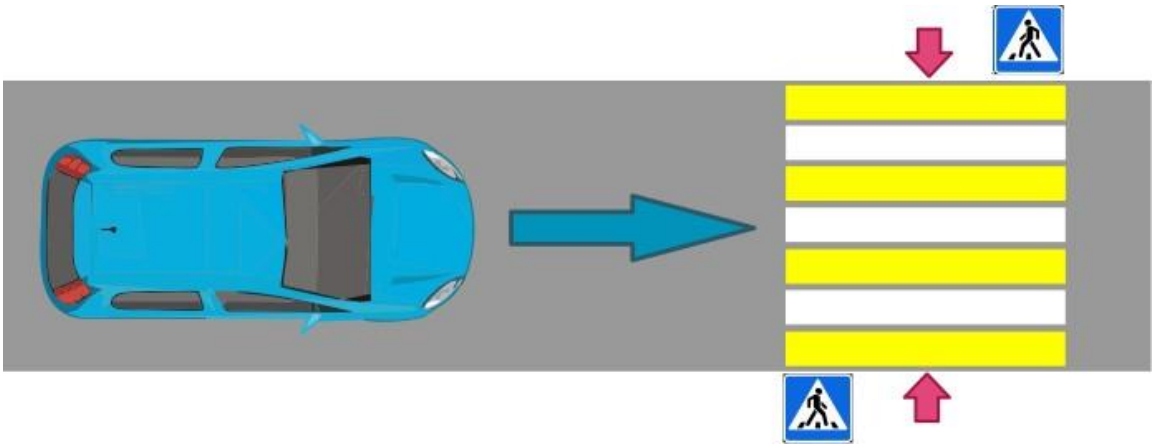


Рисунок 2.1 - Пішохідний перехід на ділянці з одностороннім рухом транспортних засобів

2.1.1 Постановка задачі з погляду теорії систем масового обслуговування.

Запити на обслуговування: автомобілі та пішоходи

Сервісний вузол: пішохідний перехід

Дисципліна обслуговування: Пріоритетна черга.

Порядок запитів на обслуговування: запити на обслуговування від автомобілів надходять частіше, ніж запити від пішоходів.

Час обслуговування: Час обслуговування пішоходів довший, ніж час обслуговування автомобіля.

2.1.2 Розробка моделі.

Повна модель пішохідного переходу, що розглядається, показана на рисунку 2.2.

Рисунок 2.2 – Схема структури моделі

Модель пішохідного переходу показана на рисунку 2.3.

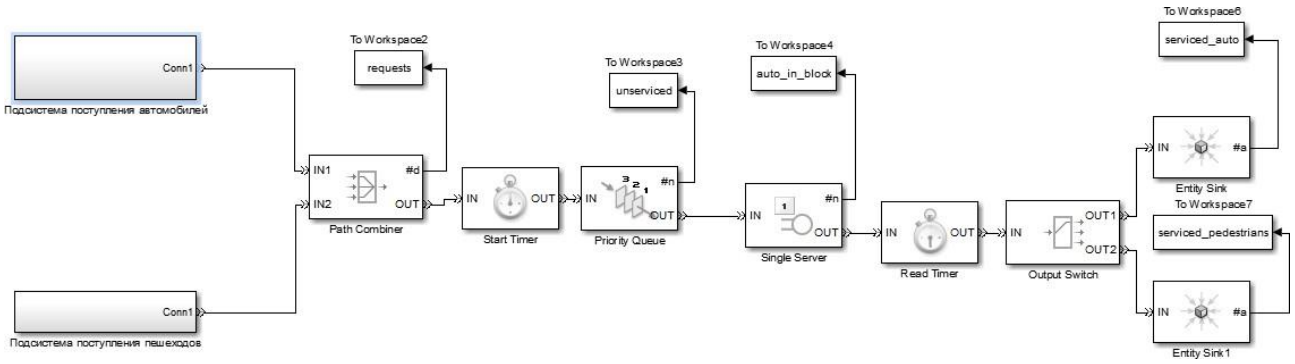


Рисунок 2.3 – Модель пішохідного переходу

Розглянемо підсистему, яка використовується для генерації запиту (рис. 2.4). У цій моделі їх дві, різниця лише в параметрах (генерація інтервалу надходження запиту та генерація часу обслуговування), тому ми розглянемо лише блоки підсистеми, а повернемо аналіз параметри та результати в розділі.

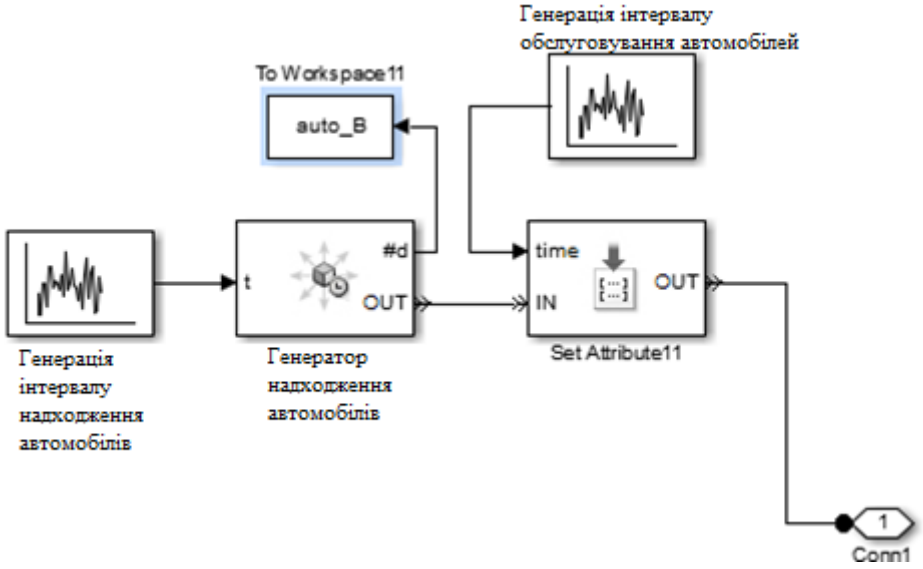


Рисунок 2.4 - Підсистеми, що генерують програми

2.2 Моделювання роботи пішохідного переходу на ділянці з двостороннім рухом.

Робота з моделювання пішохідних переходів на двосторонніх ділянках транспортної мережі (рис. 2.5) необхідна для аналізу навантаження на мережу та визначення умов її перевантаження.

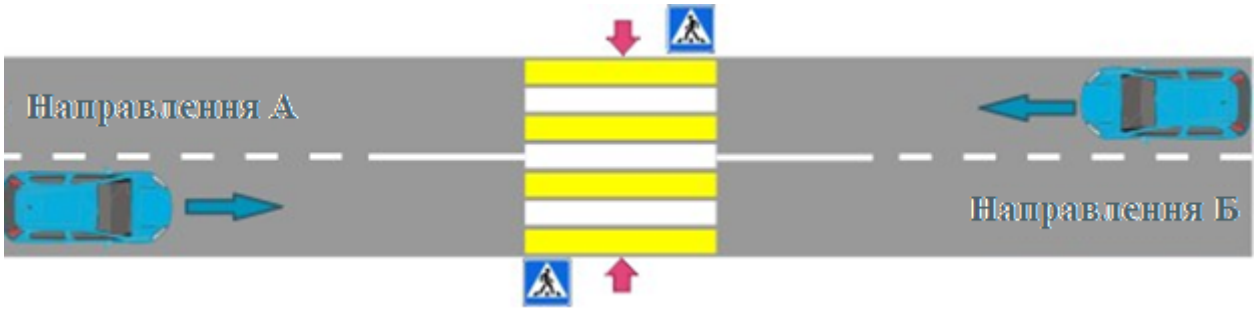


Рисунок 2.5 - Пішохідний перехід на ділянці руху транспортних засобів з двостороннім рухом

2.2.1 Постановка задачі з точки зору теорії системи масового обслуговування

Заявки на обслуговування: пішоходи, транспорт з обох сторін.

Сервісний вузол - пішохідний перехід

Правило обслуговування: для кожної смуги створюється окрема пріоритетна черга, і запити від пішоходів копіюються в кожну чергу.

Порядок запитів на обслуговування: запити на обслуговування від автомобілів надходять частіше, ніж запити від пішоходів.

Час обслуговування: Час обслуговування пішоходів довший, ніж час обслуговування автомобіля.

2.2.2 Розробка моделі

Повна модель пішохідного переходу, що розглядається, показана на рисунку 2.7. Рисунок 2.6 ілюструє блок-схему моделі.

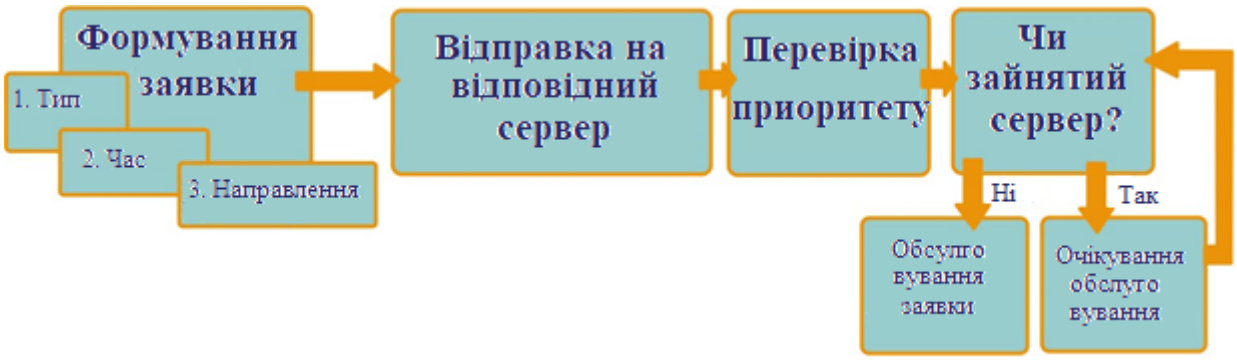


Рисунок 2.6 - Блок-схема моделі

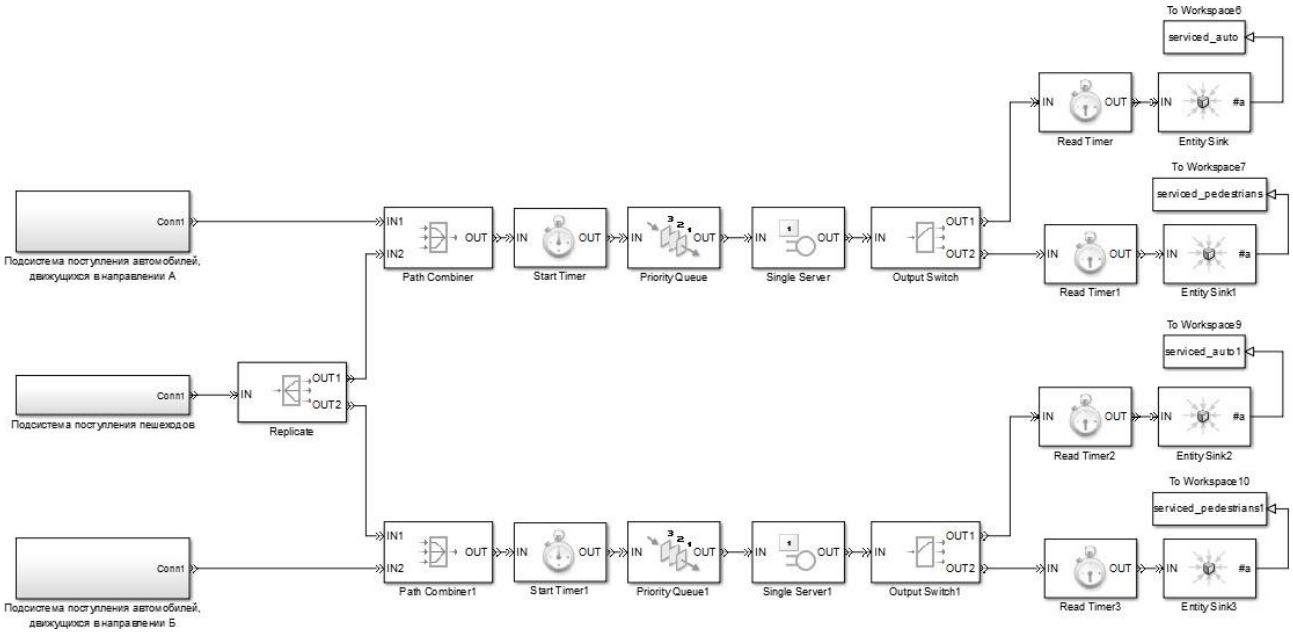


Рисунок 2.7 – реалізація моделі

Проблема при модифікації моделі полягає в наступному: при додаванні іншого напрямку руху автомобіля і підключенні цієї підсистеми до блоку «output switch», який роздає сервісний запит, сервер включає тільки один напрямок, що суперечить дійсності, оскільки дві смуги. можна керувати двома автомобілями одночасно.

Надалі було вирішено розглядати кожен діапазон як окрему підсистему зі своїм сервісним сервером. Виникає інша проблема: для кожного сервера має бути згенерований тип запиту «пішохідного» типу, щоб запит досягав кожного сервера одночасно, таким чином сигналізуючи про прибуття тієї самої людини в реальному житті.

Найпростішим рішенням є розміщення блоку «копія» (рис. 2.6), який створює копію сутності.

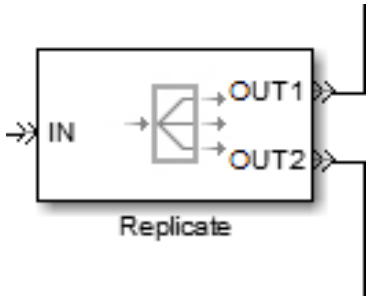


Рисунок 2.8 – Блок «Копіювати» Знову модель схожа на описану в пункті 2.1.2.

Кожен пішохідний квиток копіюється та надсилається на два сервери обробки для кожного напрямку руху. Це дозволяє блокувати всі смуги руху одночасно, поки пішоходи не закінчать переходити пішохідний перехід.

2.3 Моделювання Т-подібного перехрестя з пішохідними переходами в усіх напрямках.

Необхідно змоделювати роботу Т-подібних перехресть на односторонніх ділянках транспортної мережі (рис. 2.9), проаналізувати навантаження на мережу та визначити перевантаженість мережі.

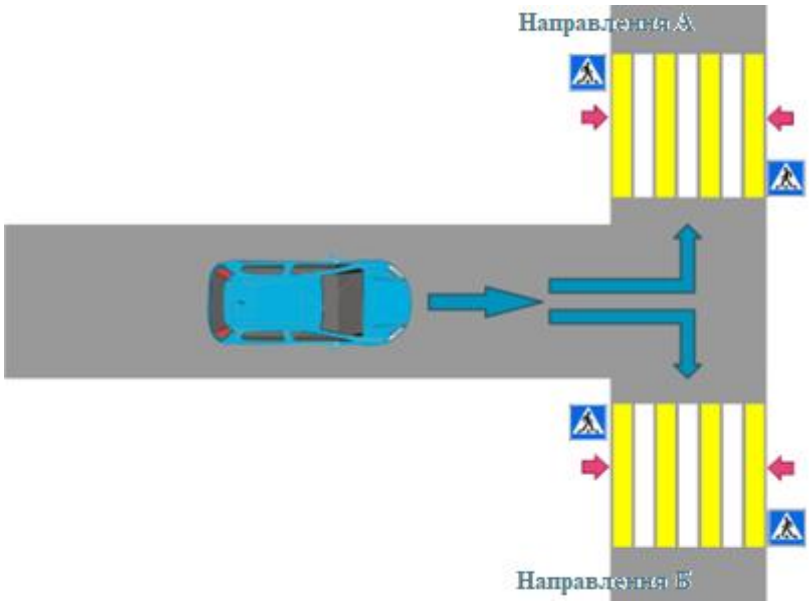


Рисунок 2.9. Пішохідні переходи перетинають Т-подібні перехрестя в усіх напрямках.

2.3.1 Постановка задачі з точки зору теорії системи масового обслуговування.

Запити на обслуговування: автомобілі та пішоходи

Сервісні вузли: Т-подібне перехрестя, Пішохідний перехід

Дисципліна обслуговування: Т-подібні перехрестя - черги FIFO, пішохідні переходи - пріоритетні черги.

2.3.2 Реалізація моделі

На рисунку 2.10 показана блок-схема моделі. На рисунку 2.11 представлена модель.

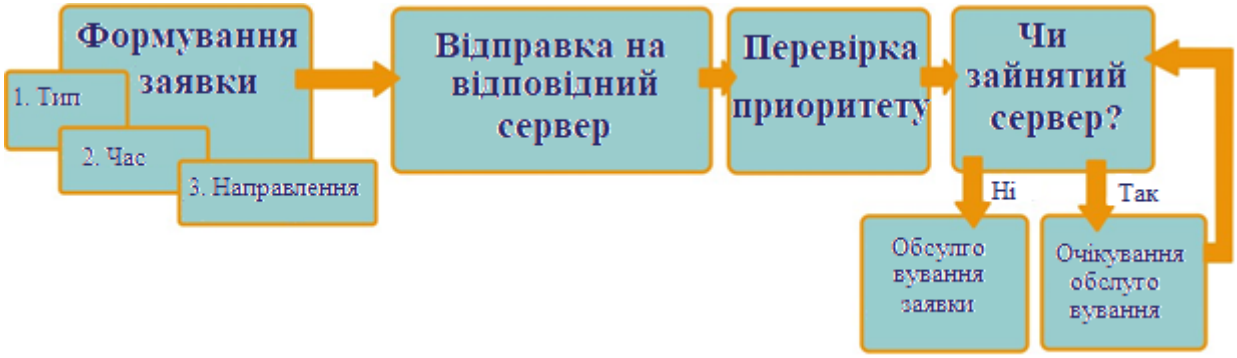


Рисунок 2.10 – Блок-схема моделі

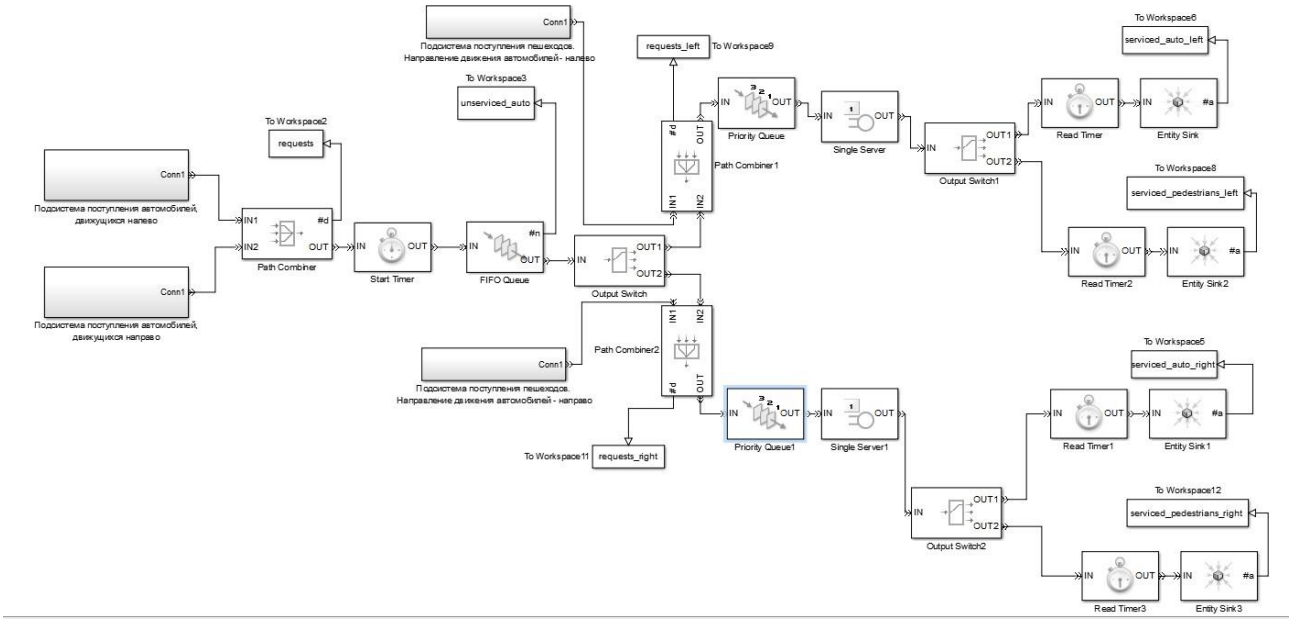


Рисунок 2.11 – Модель Т-подібного перехрестя з пішохідним переходом

2.4 Моделювання Х-перехід пішохідного переходу, що перетинає лівий і правий напрямки руху.

Необхідно змоделювати роботу Х-перетину ділянки транспортної мережі з двостороннім рухом, проаналізувати навантаження на мережу та визначити умови перевантаження мережі.

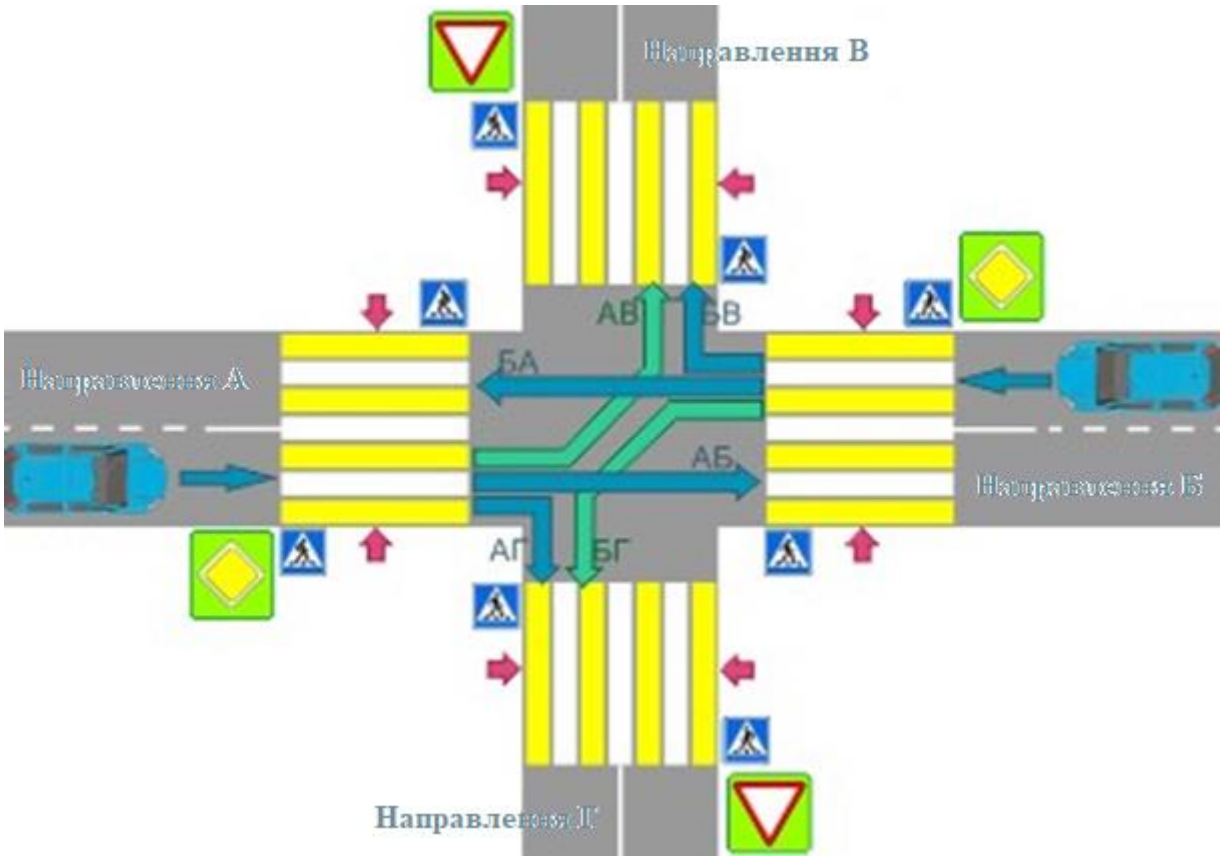


Рисунок 2.12 - Х-перетин двонаправленої транспортної ділянки мережі.

2.4.1 Постановка задачі з точки зору теорії системи масового обслуговування.

Сервісні запити: автомобілі в напрямку A (праворуч, ліворуч, прямо), автомобілі в напрямку B (праворуч, ліворуч, прямо), пішоходи в напрямках C і D на перехрестях.

Сервісний вузол - Х-перехід, пішохідний перехід

Дисципліна обслуговування: Х-перехрестя - пріоритетна черга, пішохідний перехід - пріоритетна черга.

Порядок запитів на обслуговування: запити на обслуговування від автомобілів надходять частіше, ніж запити від пішоходів.

Час обслуговування: час обслуговування пішохідного переходу довший, ніж у автомобіля, а час обслуговування автомобіля на перехресті такий самий.

2.4.2 Реалізація моделі

На рисунку 2.13 показана блок-схема моделі, а на рисунку 2.14 модель.

Рисунок 2.13 – Блок-схема моделі

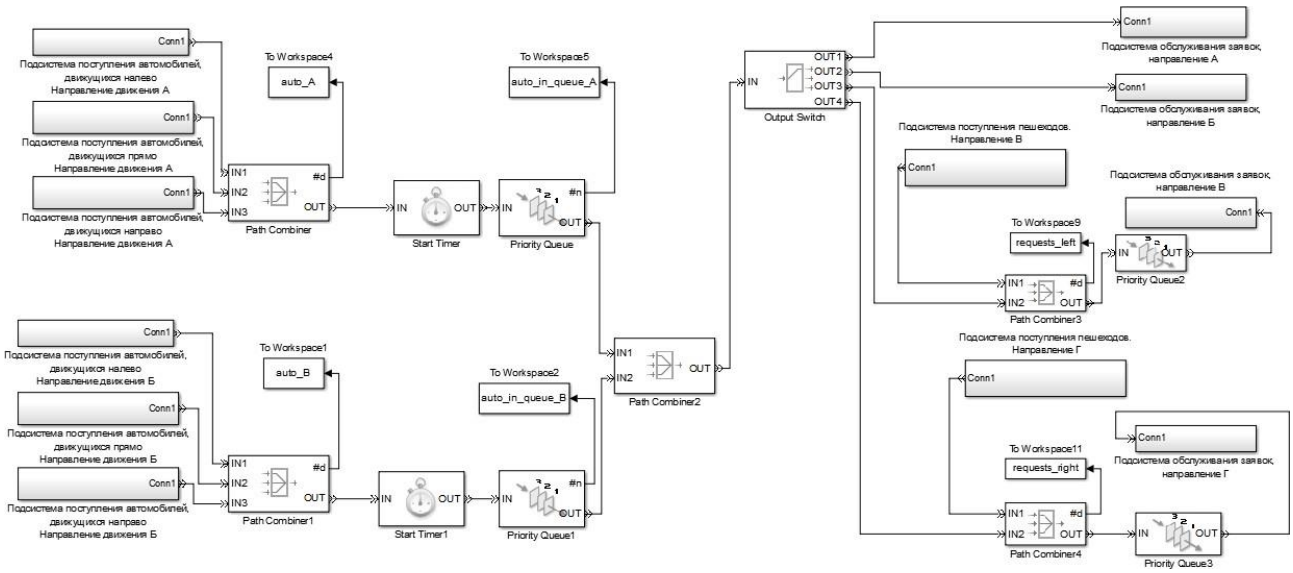


Рисунок 2.14 – Модель Т-подібного перехрестя з пішохідним переходом

2.5 Змоделювати роботу Х-перехрестя з урахуванням автомобілів з усіх напрямків та пішохідних переходів у всіх напрямках.

Необхідно змоделювати роботу Х-перетину (режим руху, показаний на рисунку 2.15) двосторонньої ділянки мережі руху, проаналізувати навантаження мережі та визначити перевантаження мережі.

Рисунок 2.15 – Принципова схема руху автомобіля та пішоходного перехрестя

2.5.1 Постановка задачі з точки зору теорії системи масового обслуговування

Запит на обслуговування: транспортні засоби в напрямках A, B, C і D (поверніть праворуч, поверніть ліворуч і їдьте прямо), а пішоходи перетинають напрямки A, B, C і D.

Сервісний вузол: Х-перехід, пішохідний перехід

Дисципліна обслуговування: Х-перехрестя - пріоритетна черга, пішохідний перехід - пріоритетна черга.

Порядок запитів на обслуговування: запити на обслуговування від автомобілів надходять частіше, ніж запити від пішоходів.

Час обслуговування: згідно з досвідом можна сказати, що час обслуговування пішохідного переходу довший, ніж час обслуговування автомобіля, а час обслуговування транспортного засобу на перехресті такий самий.

На рисунку 2.16 показана блок-схема моделі.

Рисунок 2.16 – Блок-схема моделі

2.6 Висновки до другого розділу

В другому розділі розповідається про моделювання різних випадків в транспортному русі. Також модулюються різні перехрестя, такі як: Т-подібне чи Х-перехрестя, а також розповідається про розробку моделі.

РОЗДІЛ 3. ОПИС ПРОГРАМНОГО ДОДАТКУ

3.1. Опис технічного завдання

Транспортна система – це поєднання транспортної інфраструктури, транспортних засобів і управління. Він забезпечує скоординований розвиток і функціонування всіх видів транспорту для задоволення транспортних потреб з найменшими витратами.

Транспортна інфраструктура включає транспортні мережі, які використовуються, шляхи сполучення, транспортні вузли та термінали, в яких перевантажуються товари або пересаджуються пасажири з одного виду транспорту на інший.

Транспортні засоби – це всі види транспорту, від велосипедів до поїздів.

Управління транспортною системою означає контроль системи. Він включає сигнали світлофора, стрілки на рейках та інші методи керування.

Транспортна система призначена для задоволення транспортних потреб людей і включає наступні компоненти:

- дорожньо-транспортний комплекс;
- учасники дорожнього руху;
- оточення.

Автомобільний транспортний комплекс - це сукупність різноманітних транспортних засобів, які з'єднуються і взаємодіють між собою при виконанні перевезень. [2]

Учасники дорожнього руху – це особи, які безпосередньо беруть участь у процесі дорожнього руху (водії транспортних засобів, пішоходи та пасажири транспортних засобів).

Навколишнє середовище є середовищем існування людини. У транспортній системі це все, що оточує людей під час дорожнього руху.

Одним із головних показників поточного стану міських транспортних систем є автомобілізація.

Автомобілізація є показником забезпеченості населення автомобілями. Це важлива частина суспільного прогресу. [3]

В даний час кількість автомобілів постійно зростає. Це мало великий вплив на соціально-економічний і соціальний розвиток. Як наслідок, зараз у світі активно обговорюють негативні витрати автомобілізації та їх вирішення.

Автомобілізація є частиною прогресу, але вона має як позитивні, так і негативні аспекти сучасного життя.

Автомобілі дають людям особисту мобільність для вільного пересування, що дуже важливо в сучасному житті.

Особистий автомобіль скорочує час в дорозі, дозволяє вибрати напрямок і час, позбавляє від необхідності коригувати розклад руху автобусів і маршрути, робить подорож більш комфортною. Все це позитивно впливає на настрій, що, в свою чергу, позитивно впливає на працездатність.

Але хоча транспортні засоби мають позитивні сторони, вони також мають чимало негативних показників.

Транспорт є одним із основних споживачів енергетичних ресурсів і джерелом забруднення навколишнього середовища. Автомобільний транспорт завдає великої шкоди навколишньому середовищу.

Зі збільшенням кількості транспортних засобів потрібно більше доріг, а отже, і землі. Утримувати дороги взимку, використовувати сіль та інші хімічні реактиви. Для кращої видимості при дорозі прибрали рослинність. Всі ці фактори негативно впливають на ґрунт, грунтові води та рослинність біля дороги.

Також під час руху автомобіль є джерелом шуму. Його повсякденний вплив негативно впливає на людину. Підвищена дратівливість, головний біль, швидка стомлюваність і зниження продуктивності праці.

Також автомобільний транспорт є найнебезпечнішим за кількістю аварій. Дорожньо-транспортні пригоди завдають матеріальних збитків транспортним засобам та завдають великої шкоди здоров’ю учасників.

Підвищений рівень автомобілізації призводить до проблем в організації дорожнього руху, ускладнень руху та збільшення заторів.

Пробки на дорогах – справжній крах для великих міст. Через збільшення кількості транспортних засобів у години пік транспортний потік перевищує пропускну здатність дорожньо-транспортної мережі, що може призвести до заторів.

Крім того, затори можуть бути викликані зниженням пропускної здатності з наступних причин:

- надзвичайні ситуації на дорогах;
- відсутність паркомісць;
- перевантажені вулиці з концентрацією торгових центрів та офісів;
- порушення правил дорожнього руху:
- дорожньо-транспортна пригода;
- погода;

Питання організації дорожнього руху:

- Неправильно відрегульовані світлофори;
- Вулиці закриті.

Затори призводять до збільшення затримок у русі пасажирів і збільшення часу в дорозі. Крім того, через затори на дорогах, шкідливі викиди, шум, зношеність транспортних засобів, марнотратне використання палива та збільшення кількості аварій.

Тому дуже важливо вирішити цю проблему та підвищити ефективність нагляду за транспортною системою.

Процес розробки імітаційної моделі поділяється на кілька основних етапів:

1. Збір статистичних даних;

2. Створення імітаційної моделі перехрестя;

3. Тестування моделі.

3.2 Збір базових даних

Метою етапу збору даних є підготовка набору високоякісних вхідних даних, необхідних для побудови моделі. Якість і точність вхідних даних відіграють вирішальну роль у достовірності результатів моделювання.

Вхідними даними для цього етапу є:

- необхідний тип початкових даних;
- обрані методики вимірювання;
- місце розташування місця вимірювання;
- період часу вимірювання;
- інші джерела даних.

На виході цього етапу повинен бути отриманий і зібраний повний набір необхідних вихідних даних відповідно до всіх вимог встановленої точності.

Якщо можливо, вимірювання досліджуваної області слід проводити одночасно, щоб отримати узгоджені значення. У разі невеликих ресурсів вимірювання можна проводити в різні дні, а також

Узгоджувати результати автоматичних вимірювань на основі щоденного динамічного контролю транспортного потоку.

Слід зазначити, що умови дорожнього руху в день опитування були однаковими і на них не вплинули такі фактори:

- погода;
- ремонтні роботи;
- дорожньо-транспортні пригоди;
- Публічні заходи.

Якщо опитування проводиться за наявності одного з вищезазначених факторів, дані, які згодом вводяться до SSS, будуть недостовірними, що негативно вплине на кінцеві результати моделювання.

Експертам з моделювання, які розроблятимуть і калібруватимуть моделі проблемної зони, рекомендується особисто відвідати сайт, щоб

охарактеризувати їхню функцію. Просте візуальне спостереження може виявити особливості та визначити додаткові параметри калібрування.

Основні початкові (вхідні) типи даних, необхідні для моделювання будь-якої існуючої ситуації при розробці проекту організації транспортування (ТМО), діляться на три категорії:

– транспортування;
– інфраструктура;
– допомога, специфічна для кожного програмного забезпечення.

Розглянемо кожен основний вид окремо.

1. Дані про трафік, які використовуються для побудови моделі, включають:

- інтенсивність руху по прольоту і напрямок руху на перехрестях розглянутих ділянок вулично-дорожньої мережі, отримані шляхом вимірювання, як правило, розбиті на інтервали по 5-15 хвилин;
- матриця відповідності транспортного потоку в цьому розділі;
- Склад руху за видами транспортних засобів;
- інтенсивність пішохідного руху;
- Сила циркуляційного потоку.

2. Вихідні (вхідні) дані інфраструктури включають у порядку:

- Геометричні параметри;
- Характеристики ОДД;
- Параметри налаштування, включаючи параметри підсистеми ІТС;
- Експлуатаційні параметри громадського транспорту (ТОП);
- Майбутні параметри.

Геометричні параметри можна отримати з існуючої проектної документації (наприклад, креслення в AutoCAD), супутникової чи аерофотозйомки та прямих вимірювань.

Параметри геометрії включають:

– кількість смуг (з урахуванням фактичного використання);
– ширина смуг;

- транспортна довжина;
- довжина розширеної частини;
- радіуси кривизни (з урахуванням їх впливу на швидкість руху);
- поздовжній ухил;
- параметри тротуарів та велодоріжок;
- конфігурація перехресть (включаючи пішохідні переходи та перехрестя з велосипедними доріжками).

Особливості ODD включають:

- особисті вправи заборонені;
- розподіляйте напрямки руху по смугах;
- наявність виділених смуг (для окремих видів транспорту);
- місцеве обмеження швидкості;
- є техногенне порушення;
- наявність або заборона паркування вздовж проїзної частини;
- обмежити операції з відновлення;
- певні види транспортних засобів заборонені. Параметри регулювання включають:
- розташування периферійних пристроїв (світлофорів, детекторів, камер відеоспостереження тощо);
- режими керування світлофором (кількість ступенів, почергова послідовність, тривалість циклу, основний і проміжний цикли);
- адаптивне керування параметрами;
- розташування та тип детекторів транспортних засобів;
- узгодження контрольних параметрів;
- алгоритм роботи системи управління знаками змінної інформації;
- алгоритми інформаційно-навігаційних систем;
- додаткові алгоритми роботи магістральних і мережевих ASUDD і підсистеми ITS.

ТОП робочих параметрів:

- ТОП маршрутів руху;
- місце розташування зупинки;
- рухомі інтервали або графіки;
- час трансферу на зупинці;
- тип і характеристики рухомого складу.

Основні статистичні дані, необхідні для побудови імітаційної моделі сегмента дорожньої мережі, включають:

- кількість транспортних засобів і пішоходів, що рухаються в усіх напрямках;
- швидкість руху.

Для отримання цієї інформації був обраний метод прямого локального спостереження.

При використанні цього методу отримання інформації зібрані дані є недостатньо детальними. Вимірювання зазвичай проводяться на кількох транспортних вузлах і протягом короткого періоду часу. Водночас точність і достовірність цього вимірювання неможливо визначити через використання несертифікованого методу підрахунку.

Однак цей підхід є найбільш поширеним у всьому світі через такі фактори:

- низька вартість (не потребує дорогого обладнання);
- мобільність (не потрібно чекати встановлення та налаштування
- обладнання);
- під час збору даних про дорожній рух інженери створюють особисте, професійне враження про досліджуване перехрестя.

Іншим методом збору даних є постійний моніторинг за допомогою детекторів транспортних засобів. Цей метод дозволяє отримувати інформацію з різних точок міської дорожньої мережі в один момент часу, що неможливо в оглядах трафіку через брак персоналу.

Детектори забезпечують більш широкий спектр даних і наразі вважаються сучасним та передовим обладнанням.

Для отримання більш достовірних даних у дослідженні розглядалися три основні часові періоди в будні та вихідні дні, які характеризуються різним рівнем навантаження мережі трафіком.

На рисунку 3.1 зображено графік середньої швидкості руху для обраної ділянки.

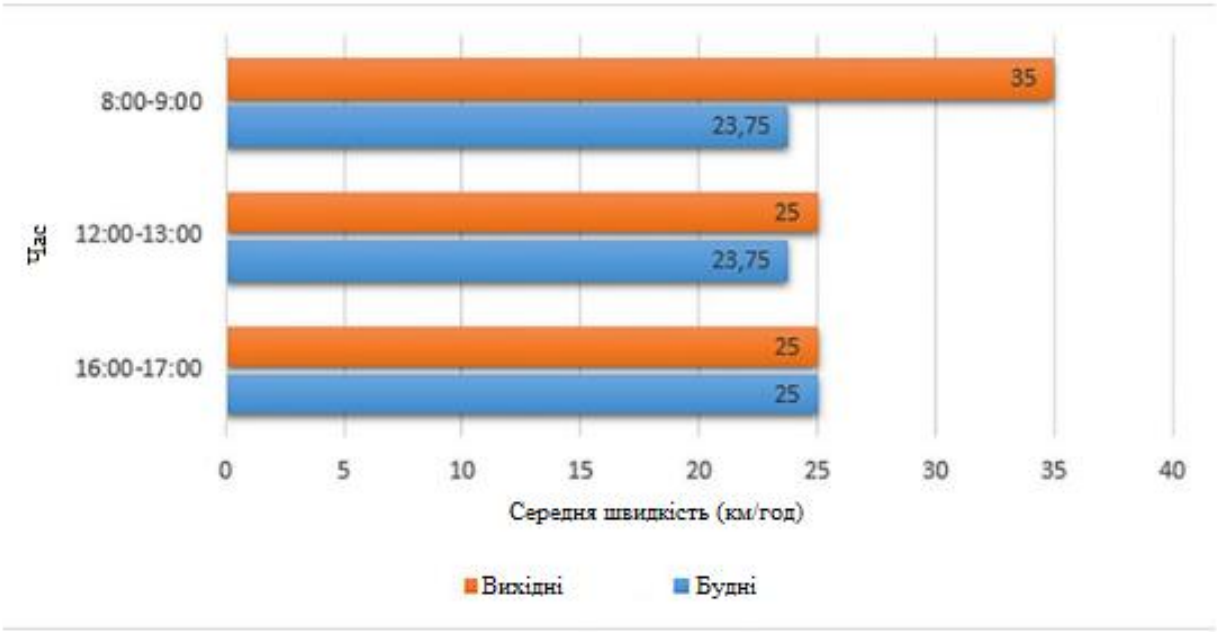


Рисунок 3.1 – Графік середньої швидкості

У таблиці 3.1 наведено дані щодо кількості транспортних засобів за напрямками руху.

Таблиця 3.1 - Отримані дані про кількість транспортних засобів за напрямком руху

Направлення руху	Кількість автотранспорту	
	Будні	Вихідні

	7.00-9.00	12.00-14.00	16.00-18.00	7.00-9.00	12.00-14.00	16.00-18.00
1	240	224	228	86	192	240
2	300	288	288	120	324	240
3	312	384	312	100	360	264
4	248	240	216	72	288	240

Крім того, при побудові імітаційної моделі перехрестя дорожньої мережі необхідно використовувати дані про цикл світлофора.

Світлофорний цикл — це періодично повторювана сукупність усіх фаз світлофорного регулювання.

Фаза світлофора - це комбінація основного та проміжного циклів світлофора.

Цикл світлофора - це час, протягом якого діє та чи інша комбінація сигналів світлофора.

Значення світлофорного циклу контрольованих ділянок дороги дивіться в таблиці 3.2.

Таблиця 3.2 — Значення для циклів світлофора

Такт	Кількість тактів
Червоний	240
Жовтий	20
Зелений	240

Параметри майбутньої інфраструктури включають очікувані зміни в структурі дорожньої мережі, схемах ODD, конфігурації естакади, системах керування тощо.

Крім даних, описаних вище, використовуються дані, що описують параметри автомобіля та поведінку водія, а також дані, характерні для кожного конкретного програмного продукту. Такі дані включають:

- довжина автомобіля (у вигляді розподілу або середнього значення та відхилення);
- розподіл необхідних швидкостей руху;
- середні і максимальні значення прискорення і уповільнення;
- параметри шкідливих викидів для типу автомобіля та режиму руху;
- спеціальні налаштування режимів поведінки (слідувати за лідером, змінювати смуги руху, вибирати проміжки).

Вимоги до точності вхідних даних для побудови моделі повинні визначатися необхідною техніко-економічною можливістю, яка залежить від допустимої похибки результатів моделювання та вартості підвищення точності вхідних даних. Перераховані вище вхідні параметри для побудови моделі можна розділити на детерміновані та випадкові. Детерміновані параметри включають незмінні, чітко визначені параметри, такі як геометрія HPS, параметри ODD, режими керування та робочі параметри. Точність таких параметрів не залежить від часу, витраченого на їх отримання, або кількості вимірювань, і тому повинна бути 100%. Будь-які невідповідності в параметрах цього типу класифікуються як помилки введення, а не помилки збору даних.

До стохастичних параметрів в основному відносяться такі параметри транспортного потоку, як інтенсивність, швидкість, склад, і отримані з них параметри умов руху - час у дорозі, довжина черги, фактична пропускна здатність тощо. Значення цих параметрів можуть змінюватися щодня. днів, що ускладнює їх точне визначення. Для якісної оцінки потрібне середнє значення таких параметрів. Рекомендується щонайменше 3 вимірювання параметрів випадкового типу. Далі слід оцінити розподіл отриманих значень. Додаткові вимірювання необхідно проводити, якщо відхилення від середнього перевищує 20% [34].

3.3 Побудова моделі

Після початкового введення вихідних даних правильну роботу моделі та введену інформацію необхідно перевірити перед тим, як перейти до етапу калібрування, процесу, який називається перевіркою. Якість введення та початкової настройки моделі може суттєво вплинути не тільки на складність подальшого калібрування та валідації, але й на ймовірність її реалізації в принципі.

Метою цього етапу є підтвердження правильності введення даних з таких аспектів:

- відсутність помилок при безпосередньому введенні числових параметрів;
- правильність основних установок і масштабів елементів моделі;
- враховуйте конкретні фактори.

З точки зору перевірки помилок введення, необхідно переконатися, що введені значення для допустимої швидкості, інтенсивності руху між відправленням/пунктом призначення, тривалості етапу, параметрів транспортного засобу та інших числових параметрів відповідають початковим зібраним даним. Типовими помилками на цьому етапі можуть бути прості орфографічні помилки, наприклад зайві нулі або неправильний вибір поля для вхідного значення.

Перевірку правильності основних налаштувань і масштабу елементів моделі необхідно проводити окремо для кожного елемента. Розглянемо ці елементи базового налаштування.

1. Вивчіть загальну структуру діаграм UDS та схем ODD.

Під час перевірки необхідно перевірити правильність введення елемента MAC та його основних параметрів, наприклад:

- фактична кількість смуг;

- дозволені напрямки;
- допустима швидкість;
- Геометрія перетину.

2. Параметри ТС.

Необхідно уточнити склад типів транспортних засобів, які використовуються в цьому проекті, а також параметри цих типів, такі як довжина, прискорення, уповільнення тощо.

3. Параметри поведінки водія.

На етапі перевірки початкові параметри поведінки водія повинні бути обґрунтовано обрані, такі як середня відстань у часі, агресивність водіння, налаштування моделі зміни смуги руху, частка порушників швидкості, частка незнайомих водіїв тощо. Розділи, що розглядаються, тощо.

4. Параметри перетину.

Для нерегульованих перехресть слід перевірити значення параметрів моделі вибору безпечного зазору. Вони залежатимуть від видимості, геометрії перехрестя, типу транспортного засобу, рівня навантаження. Якість налаштування моделі вибору пропуску може відігравати важливу роль у відтворенні пропускної здатності мережі.

Ви повинні перевірити, наскільки добре працює поворот, скільки смуг насправді потрібно для повороту, скільки автомобілів поміщається в зону нагромадження та як вони впливають на рух в інших напрямках. Слід також перевірити поведінку водіїв на перехрестях, які можуть блокувати другорядні напрямки через перевантаження.

На контрольованих з’єднаннях ретельно перевіряйте налаштування параметрів керування, таких як поділ фаз, час циклу, основні та проміжні цикли, адаптивні та узгоджені параметри керування. Контрольні параметри слід перевіряти за наявною документацією та висновками. Ключовий параметр, щоб бути максимально реалістичним, - зупинити насичений потік сегмента лінії.

Вплив траєкторії, нахилу, видимості, складу потоку, ширини смуги, джерел від пішоходів, велосипедистів, ТОС-станцій тощо на насичений потік слід ретельно вивчити. Наближаючись до перехрестя, слід перевірити ділянку зміни смуги на наявність подальших дій.

5. Особливості взаємодії різних учасників спорту.

Слід перевірити, наскільки достовірно відтворюється взаємодія транспортного потоку з пішоходами та громадським транспортом. Взаємодія з пішоходами в основному відбувається на перехрестях. Такі взаємодії слід явно моделювати, лише якщо вони мають значний вплив на дорожню ситуацію, наприклад, заблоковані повороти, неконтрольовані перехрестя з високою щільністю пішоходів, регульовані перехрестя під час фази виклику пішоходів.

Необхідно перевірити роботу громадського транспорту, особливо в районі зупинок. Слід звернути увагу на фактичний розклад, маневрені характеристики, час посадки та висадки, швидкість тощо. При наявності виділених смуг і пріоритетних пропусків необхідно перевірити їх роботу. Особливу увагу слід звернути на висадку пасажирів на трамвайних зупинках і відправлення автобусів з місцевих розширень.

6. Параметри навантаження.

Крім перевірки правильності введених значень параметрів транспортного навантаження, необхідно перевірити наступні моменти: склад трафіку за типом транспортного засобу, розподіл попиту в часі (з інтервалами 5-15 хвилин), коректність роботи з використанням відповідних матриць (наприклад, потрібно).

7. Параметри моделі маршрутизації.

Якщо розглянутий сегмент дорожньої мережі має можливість пересування автомобілів різними маршрутами, і такий розподіл впливає на результати, особливу увагу слід приділити вибору та коригування моделі маршруту. На етапі перевірки слід перевірити наступні аспекти:

- обґрунтованість обраної моделі розподілу потоку по лінії;
- вибрати основні налаштування для даної моделі;
- метод формування функції тарифу;
- параметри сегмента, які впливають на вибір маршруту.

Для виявлення друкарських помилок рекомендуються наступні методи:

1. Запустіть модель при невеликому навантаженні (менше 50% від пікового значення), щоб знайти можливі помилки. Якщо перевантаження відбувається при низькому навантаженні, то це може бути пов'язано з помилкою;

2. Слідкуйте за кількома траєкторіями транспортних засобів уздовж критичного маршруту. Відстежуйте несподівані гальмування та зміни смуги руху;

3. Запустіть модель із навантаженням 50% і вище, щоб проаналізувати здійсненність вимог. Слід перевірити, чи весь попит надходить у мережу і який відсоток виходить.

Висновок: на виході фази валідації модель має гарантовано не мати вхідних помилок і правильно встановлювати основні елементи та їх співвідношення, тобто вона повинна бути добре підготовлена до калібрування [34].

Побудова імітаційної моделі сегмента міської транспортної мережі складається з двох основних етапів:

- розробити графічну частину імітаційної моделі;
- розробити проксі-частину імітаційної моделі.

Розробка графічної частини імітаційної моделі передбачає побудову візуального відображення частин дорожньої мережі з використанням елементів вбудованої бібліотеки – доріг, світлофорів, пішохідних переходів, що потрапляють у імітаційну частину.

Елементи, використані для побудови моделі, перераховані в таблицях 3.3-3.4.

Таблиця 3.3 – Елементи бібліотеки трафіку, що використовуються для розробки графічної частини імітаційної моделі.

Бібліотека	Назва елементу	Опис	Основні параметри
Дорожнього руху	Дорога	Є графічним елементом розмітки простору, це безперервна дорога (тобто така дорога, яка не містить перехресть)	Напрямок руху, ширина смуги, колір дорожньої розмітки, кількість смуг основного руху
	Перехрестя	Є графічним елементом розмітки простору, використовується для з'єднання двох або більше доріг	З'єднувачі смуг (напрямки руху транспорту на перехресті)
	Роздільна полоса	Є графічним елементом розмітки простору, використовується для розділення направлення	Кількість смуг зустрічного руху, ширина розділової смуги, колір розділової смуги

Таблиця 3.4 – Елементи демонстраційної бібліотеки, що використовуються для розробки графічної частини імітаційної моделі.

Бібліотека	Назва елементу	Опис	Основні параметри
Фігури	Прямокутник	Геометрична фігура, для малювання презентації моделі автомобіля	Колір заливки, колір лінії, товщина лінії, стиль лінії
	Овал	Геометрична фігура для малювання презентації моделі світлофора	Колір заливки, колір лінії, товщина лінії, стиль лінії, радіус
	Лінія	Геометрична фігура для малювання презентації моделі доріг	Колір заливки, колір лінії, товщина лінії, стиль лінії

3.4 Тестування моделі

Під час тестування на моделі було проведено кілька експериментів, щоб налаштувати настроювані параметри. Оскільки на реальних ділянках вулично-дорожньої мережі регулювати кількість транспортних засобів і пішоходів неможливо, експерименти проводять на світлофорах і пішохідних переходах.

Результати випробувань наведені в таблиці 3.5.

Таблиця 3.5 - Модельні випробування

Випробування	Результат	Причина випробування
Збільшення циклу зеленого сигналу світлофора для пішоходів на переході в напрямках 2, 3	З обох боків пішохідного переходу утворюються затори	На даній ділянці проходить велика кількість пішоходів, тому вирішено було збільшити зелений цикл світлофора.
Збільшення зеленого світлофорного циклу для пішоходів на переході за напрямками 1,4	Утворюються пробки з обох боків від пішохідного переходу	На даній ділянці зелений сигнал працює не довго, тому вирішили збільшити час.

За результатами експерименту можна зробити висновок, що перший тест позитивно вплинув на модель, яка може враховувати зміни в реальній системі. Решта експерименту призвела до погіршення умов руху на перехресті. У 2 і 3 напрямках проїжджає велика кількість транспорту, тому світлофори знижують пропускну здатність і спричиняють затори.

3.5 Опис середовища та системи для розробки імітації руху на перехресті

Оскільки C# одна із найпоширеніших мов програмування, йому існує велика кількість інтегрованих середовищ розробки з різним функціоналом. Оскільки висока продуктивність є однією з основних вимог до розробки програми.

Важливим критерієм вибору середовища розробки програми є наявність профільника. Як об'єкти дослідження були обрані інтегровані середовища розробки MS Visual Studio, JetBrains Clion, Qt Creator, NetBeans та Eclipse SDK , як найбільш функціональні та поширені.

а) MS Visual Studio

Microsoft Visual Studio - це інтегроване середовище, призначене для розробки програмного забезпечення, що має низку додаткових інструментів.

Продукти MS Visual Studio дозволяють розробляти консольні програми, програми з графічним інтерфейсом, веб-сайти, веб-програми та веб-сервіси для платформ.

Рис. 3.1 –Microsoft Visual Studio

MS Visual Studio містить вбудований інструмент для редагування вихідного коду програми та дозволяє проводити рефакторинг коду. Відладчик, вбудований у MS Visual Studio, має два рівні роботи: рівень налагодження вихідного коду та рівень налагодження машинного коду. Додаткові вбудовані інструменти включають форми GUI та зручний редактор для створення програм з інтерфейсами, містять веб-редактор та редактор класів. Це середовище розробки підтримує можливість створення та підключення плагінів (надбудов), що використовуються для розширення функціональності. MS Visual Studio підтримує системи контролю версій вихідного коду (наприклад, Visual SourceSafe або Subversion), містить безліч інструментів, що дозволяють

редагувати та проектувати код предметно-орієнтованими мовами програмування.

б) JetBrains Clion

JetBrains CLion — інтелектуальне інтегроване середовище розробки програмного забезпечення, призначене для розробки додатків мовами C# і C++ [28]. JetBrains Clion дозволяє розробляти програмне забезпечення на платформах Windows, OS X та Linux.

Рис. 3.2 - JetBrains CLion

Багатофункціональне середовище розробки - JetBrains CLion включає вбудований налагоджувач, безліч шаблонів коду та інтерфейс для популярних систем управління версіями (таких як Subversion, Git, GitHub, Mercurial, CVS, Perforce та TFS). CLion надає можливості автодоповнення та автоформатування коду, виконує аналіз коду на стрічці з виділенням потенційних проблем та пропонує способи їх усунення, підтримує систему складання для кросплатформових проектів CMake, а також різні рефакторинги коду. JetBrains CLion має великий репозиторій модулів (доповнень) для розширення існуючої функціональності.

в) Creator Qt

Qt Creator – це повністю інтегроване середовище розробки програмного забезпечення, що містить інструменти для проектування та розробки складних програм у різних операційних системах.

Рис. 3.3 - Qt Creator

Qt Creator призначений для розробки програмного забезпечення мовою програмування C++ та C#. Також є PyQt для програмування на Python та PHP-Qt для PHP.

Qt Creator інтегрований з кросплатформовими системами автоматизації складання: qmake і CMake, містить редактор коду Qt Designer, що дозволяє проектувати і створювати графічні інтерфейси користувача (GUI) з віджетів Qt. Це середовище містить багато корисних інструментів, таких як підтримка систем контролю версій (Subversion, Git, GitHub, Mercurial та CVS) та емулятор Qt. Qt Creator містить внутрішній налагоджувач для налагодження звичайних програм C++, а також включає можливість підключення мобільних пристроїв до вашого комп'ютера та налагодження програм, що працюють на них.

г) NetBeans

IDE NetBeans – це безкоштовне середовище IDE з відкритим вихідним кодом для розробників програмного забезпечення [30]. NetBeans надає безліч різних інструментів для створення професійного програмного забезпечення мовами Java, C/C++, PHP, Python, JavaScript, Groovy, Ruby та інших.

Рис. 3.4 - IDE NetBeans

Середовище розробки NetBeans включає такі функції:

– рефакторинг коду;
– профільування;
– виділення кольором різних синтаксичних конструкцій;
– автоматичне завершення при вході до різних споруд;
– велика кількість шаблонів коду.

У середовищі IDE NetBeans спочатку має бути J2EE SDK або потрібна версія Sun JDK.

NetBeans надає засоби управління системами контролю версій з готовими інтеграціями для Subversion, Mercurial і Git. Редактори середовища IDE та функція перетягування також дозволяють швидко та ефективно розробляти графічні інтерфейси програм.

г) Eclipse SDK

Eclipse SDK – це безкоштовне інтегроване середовище розробки програмного забезпечення з відкритим кодом. Eclipse SDK має велику і розгалужену систему модулів (надбудов), що підключаються, яка забезпечує робоче середовище з такими мовами програмування, як C, C++, PHP, Perl, Python, Ruby, Ada або COBOL.

Рис. 3.5 - Eclipse SDK

Eclipse SDK дозволяє створювати кросплатформне програмне забезпечення для Microsoft Windows, Mac OS X, дистрибутиви Linux та навіть Solaris. Це середовище включає засоби розробки Eclipse Java (JDT), які містять компілятор Java. У програмі використовуються інструменти, що входять до системи Eclipse Rich Client Platform.

Така система допомагає розробникам створювати потужні функціональні користувацькі програми з гарним графічним інтерфейсом на основі CSS (Cascading Style Sheets – каскадні таблиці стилів). Інтегроване середовище розробки Eclipse SDK надає Java IDE з усіма інструментами, необхідні для розробки якісних програм.

Через війну вивчення розглянутих середовищ розробки програмування мовою C# було створено порівняльна таблиця 3.6.

Параметри	MS Visual Studio	JetBrains Clion	Qt Creator	Eclipse SDK	NetBeans
Редактор користувальницького інтерфейсу	+	-	+	+	+
Вбудований профільувальник	+	-	-	-	-
Вбудований відладчик	+	+	+	+	+
Підтримка C#	+	+	+	+	+
Встановлене ПЗ на підприємстві, наявність ліцензії	+	-	+	-	-
Опи використання	+	-	+	+	-
Сумісність з колишніми проектами	+	-	+	-	-

Для розробки програми вибрали середовище розробки Microsoft Visual Studio. Середовище розробки Microsoft Visual Studio було обрано тому, що він має ряд необхідних для розробки функцій, таких як підтримка профілювання коду мовою програмування C#, налагодження та сумісність з успадкованими проектами.

3.6 Вимоги до технічного забезпечення

Технічна підтримка системи має максимально ефективно використовувати існуючі технічні засоби.

До складу комплексу повинні входити такі технічні засоби:

1) сервер бази даних;

2) персональні комп'ютери (ПК) користувачів.

Мінімальні вимоги до характеристик апаратних компонентів, за яких значення тимчасових параметрів Системи повинні відповідати вимогам, представленим у ТЗ:

для сервера бази даних:

- Процесор - 2 x IntelXeon3 ГГц;
- обсяг оперативної пам'яті – 16 ГБ;
- дискова підсистема – 4 x 146 ГБ;
- Дисковод компакт-дисків (DVD-ROM);
- Мережевий адаптер - 100 Мбіт/с.

для ПК користувача:

- процесор - Intel Pentium 1,5 ГГц;
- обсяг оперативної пам'яті – 256 МБ;
- дискова пам'ять – 40 ГБ;
- мережевий адаптер - 100 Мбіт/с.

3.7 Опис методів

Є два шляхи вирішення вищезазначеної проблеми:

- Реконструкція існуючих та будівництво нових доріг транспортної мережі міста.
- Підвищення ефективності нагляду за міською транспортною мережею.

Реконструкція існуючих доріг і будівництво нових доріг не є рішенням, оскільки це фундаментальний характер і має багато недоліків:

- Висока капіталомісткість;
- Вимагає великих витрат часу;
- Не застосовується до всіх ділянок транспортної мережі (вузькі вулиці, ділянки історичної цінності).

А в перспективі така політика може призвести до зворотного. Дослідження у Великій Британії підтвердило, що збільшення пропускної спроможності доріг призвело до збільшення кількості поїздок на 20%, а в районах з високою щільністю населення – до збільшення кількості поїздок на 40%.

Крім того, ці заходи, оскільки кількість транспортних засобів продовжує зростати, не повністю зменшують різницю між пропускною здатністю та інтенсивністю руху.

Підвищення нормативної ефективності міських транспортних мереж є менш витратним рішенням. Він включає перенастроювання параметрів налаштування мережі передачі, таких як фаза світлофора.

Сучасні технології дозволяють автоматизувати процес перенастроювання параметрів контролю. Автоматизація означає використання інтелектуальних транспортних систем.

Інтелектуальна транспортна система — це інтелектуальна система, яка використовує інноваційні розробки в моделюванні транспортної системи та регулюванні транспортного потоку, щоб надати кінцевим користувачам більше інформаційного наповнення та безпеки, а також якісно покращити участь у транспортуванні порівняно з традиційною системою взаємодії з користувачем транспорту.

Вирішіть задачу використання ITC для підвищення ефективності шляхом побудови інтегрованої системи:

Люди – транспортна інфраструктура – транспортні засоби, максимально використовуючи новітні інформаційні та керуючі технології.

ІТС означає використання джерел інформації — засобів зв’язку та засобів управління та контролю, вбудованих у транспортні засоби та інфраструктуру (світлофори, камери спостереження, розумні покажчики, інформаційні табло), а також багаторівневих систем контролю. Відповідно до інформації, отриманої в режимі реального часу, система управління керує мережею передачі.

Імітаційне моделювання є спрощеним, але настільки ж ефективним варіантом для інтелектуальних транспортних систем.

У цій роботі змодельовано двостороннє перехрестя з двома напрямками руху, де з’єднані чотири смуги руху L_1, L_2, L_3 і L_4, де на кожному розі встановлено світлофор (T_1, T_2, T_3 і T_4). рисунок 3.6.

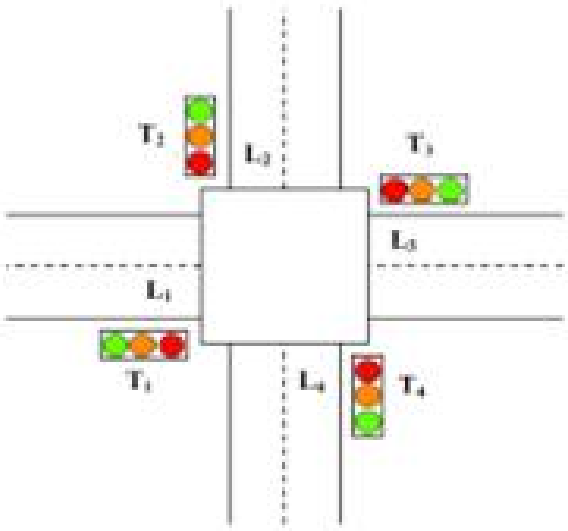


Рис. 3.6 – модель перехрестя

Просте перехрестя дорожніх світлофорів для водіїв. Для кожного світлофора і кожного циклу є три послідовні стани: зелений, жовтий і червоний. Частка автомобілів, що прибули на смугу L_i в момент часу t - $\lambda_i(t)$, i=1,2,3,4. Коли світлофор стає зеленим (жовтим відповідно), у момент часу t $\mu i(t)$ (відповідно $k_i(t)$), частка автомобілів, що виїжджають на смугу L_i, i=1,2,3,4. Припустимо, що жовте світло горить протягом фіксованого часу, що дорівнює δyellow (цей час визначається відповідно до встановлених стандартів, стандартів багато:

американських, європейських і т.д.; зазвичай коливається від 3 до 4 секунд). Зелене світло горить на змінний проміжок часу, розрахунок якого залежить від: швидкості, що дотримується на дорозі, кількості смуг, потоку транспортних засобів тощо, і коливається від 20 до 30 секунд. Вважається, що час увімкненого світлофора дорівнює сумі часу, коли світлофор на зустрічній смузі горить жовтим світлом (див. табл. 3.7).

Таблиця 3.7. Схема перемикання сигналів світлофора

Цикл	T_1	T_2	T_3	T_4
$[t_0, t_1 – δжов)$	Червоний	Зелений	Червоний	Зелений
$[t_1 – δжов, t_1)$	Червоний	Жовтий	Червоний	Жовтий
$[t_1, t_2 - δжов)$	Зелений	Червоний	Зелений	Червоний
$[t_2 - δжов, t_2)$	Жовтий	Червоний	Жовтий	Червоний
$[t_2, t_3 - δжов)$	Червоний	Зелений	Червоний	Зелений
$[t_3 - δжов, t_3)$	Червоний	Жовтий	Червоний	Жовтий

У цій роботі ми також пропонуємо модель простої задачі керування рухом перехрестя. Є перехрестя, де зустрічаються дві вулиці, обидві смуги йдуть в одному напрямку, і повороти ліворуч заборонені. На кожному розі перехрестя є світлофори. Нам потрібно з’ясувати період, протягом якого світлофори T1 і T3 повинні залишатися зеленими (червоні T2 і T4), а T1 і T3 – червоними (зелені T2 і T4), щоб уникнути заторів, які описуються кількістю транспортних засобів, що очікують на кожній смузі (довжина черги).

Дослідження також було використано для оцінки заторів на смугах руху, враховуючи різні критерії: суму середньої довжини черги на смугу, середній час очікування, довжину найдовшої черги або комбінацію двох. Таким чином, мета полягає в тому, щоб обчислити оптимальний часовий ряд світлофорів, щоб мінімізувати всі вищевказані критерії. Це дозволяє нам моделювати та

розв’язувати проблеми, які мають два етапи в кожному циклі: рисунок 3.7 і рисунок 3.8.

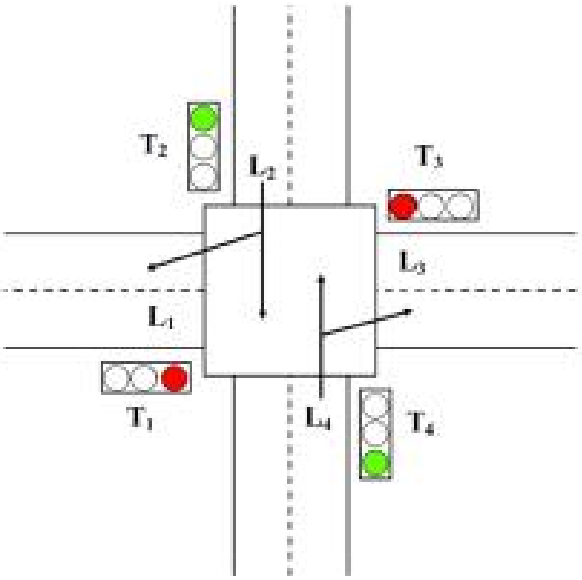


Рисунок 3.7 – перша фаза

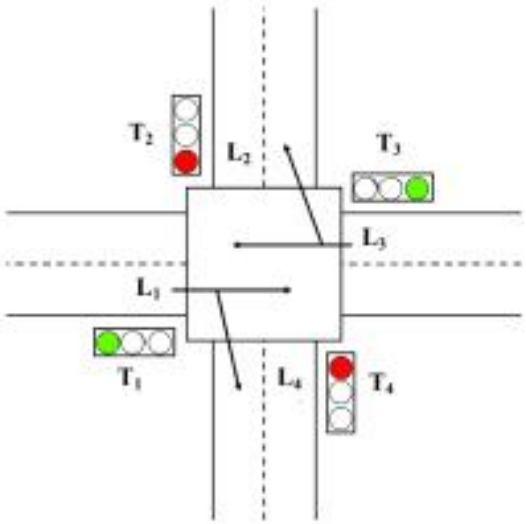


Рисунок 3.8 – друга фаза

На основі поставленої задачі розроблено інформаційну систему імітаційного моделювання. Система може імітувати рух автомобілів по дорозі, і аналізувати машини, яким потрібно виїхати на перехрестя під час руху, тобто машина зупиниться на червоне світло і не заїде на перехрестя, на зелене – так.

Бажану швидкість у системі можна змінити, щоб краще контролювати процес моделювання. Для встановлення затримки на світлофорі встановлені спеціальні елементи керування, які дозволяють задавати затримку між перемиканнями сигналів світлофора.

3.8 Висновки до розділу 3

За результатами роботи можна судити, що модель може бути використана для аналізу та оптимізації міської транспортної системи. Модель зазнає потрійного ускладнення шляхом додавання напрямків руху та додавання порядків вищого пріоритету.

Однак останні результати моделювання показують, що відсутність світлофорного контролю негативно впливає на умови руху, велике перевантаження транспортних вузлів, і навіть зниження інтенсивності прибуття автомобілів не може зменшити навантаження на транспортні вузли.

РОЗДІЛ 4. ЕТАПИ РОЗРОБКИ ТА СТВОРЕННЯ ПРОГРАМНОГО ДОДАТКУ

4.1 Опис розроблювального алгоритму

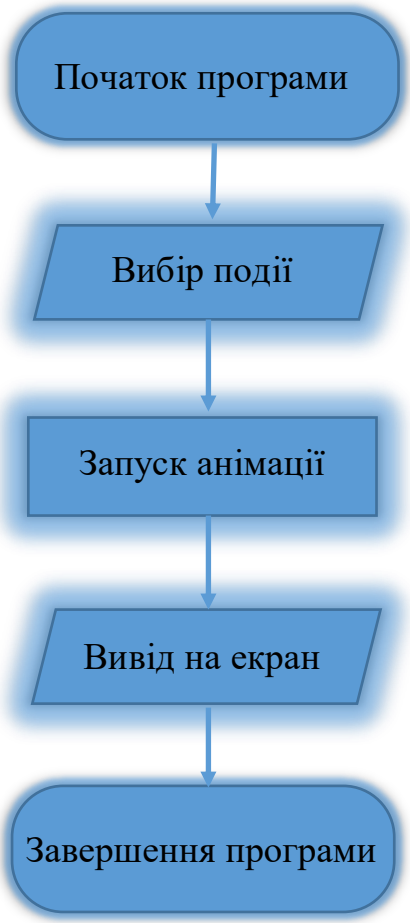
Завдання, представлені в цій роботі, охоплюють невелику підмножину підзадач, кожна з яких вирішується окремо. Основним завданням є створення моделі (системи), яка дозволяє наочно продемонструвати роботу різних алгоритмів, які використовуються в теорії моделювання транспортних потоків. Теорія моделювання транспортного потоку надає інформацію про можливі затори або надзвичайні ситуації на маршруті перевезення, які можуть бути результатом необережного керування транспортним засобом і заторів на самому маршруті перевезення. Уникнення транспортних засобів під час руху по дорозі є важливою частиною правильного функціонування системи, оскільки процес моделювання повинен відповідати фактичному завданню, в якому кожен водій повинен вирішити, як безпечно об'їхати інший автомобіль під час руху по дорозі, за умови, що виконуються такі умови: Швидкість цього автомобіля мала. Щоб вирішити цю ситуацію під час симуляції, загальним рішенням є пошук усіх транспортних засобів, що знаходяться поруч із транспортним засобом, що рухається, у кожній петлі, і якщо наступний транспортний засіб може спричинити зіткнення, просто знизити швидкість до безпечної швидкості, якщо його неможливо уникнути, або обійти транспортний засіб. Для такого алгоритму завжди існує багато критеріїв порівняння, тому можна створювати шаблони. Часто алгоритми управління світлофором є доповненням до таких завдань і не завжди потрібні. Такий алгоритм має створити певні умови, за яких транспортний засіб не повинен проїжджати через зону, обмежену світлофором, до зміни статусу світлофора. Цей алгоритм часто поєднується з моделюванням пішохідного переходу. У цьому випадку завдання ускладнюється ще й необхідністю імітувати поведінку пішоходів, що стоять в черзі перед переходом і під час переходу. Алгоритм руху пішоходів, що переходять дорогу, є окремим

випадком алгоритму, який об’їжджає перешкоди або інші транспортні засоби під час руху по дорозі, і його можна ігнорувати. Окрім регулювання дорожнього руху, світлофори також створюють умови для перевірки правильності алгоритму об’їзду транспортних засобів та скупчення транспортних засобів перед світлофором. Алгоритм моделювання маршрутизації є найпростішим для реалізації, і не завжди потребує великої кількості вхідних параметрів. Під час руху по дорозі водій самостійно вирішує, якою дорогою їхати відповідно до даних правил дорожнього руху та відповідно до власних потреб. Водій може навіть самостійно вибрати інший шлях, але, знову ж таки, він вибере лише напрямок повороту, виходячи з дозволеного зачеплення. Виходячи з цих міркувань, самовибіркове моделювання може бути виконано за допомогою генератора псевдовипадкових значень, який дозволить вам отримати псевдовипадкове значення та, на основі цього значення, повернути в певному напрямку або продовжити прямо.

Необхідно розробити алгоритми об’їзду транспортних засобів під час руху по дорозі, регулювання дорожнього руху за допомогою світлофора, імітації вибору водієм шляху. Для виконання цієї роботи були розглянуті різні моделі, включаючи гідродинамічні моделі, моделі Лайтхілла-Вітема, моделі Гріншилдса та Грінберга, а також моделі клітинних автоматів. Модель клітинних автоматів виявилася найбільш прийнятною реалізацією, оскільки її можна використовувати для легкої візуалізації результатів алгоритму.

Світлофори можна налаштувати, ввівши спеціальні значення в матрицю, і алгоритм об'їзду автомобіля сприймає їх як червоний або зелений сигнал світлофора.

На основі вище сказаного, було розроблено наступний алгоритм програми:



4.2 Написання програмного продукту

Для початку роботи програми були підключені наступні бібліотеки

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

Далі було намальовано перехрестя, за допомогою ліній

```
//road (top-left)
```

```
g.DrawLine(new Pen(Brushes.Red, 4), new Point(0, 300), new
Point(300, 300));
g.DrawLine(new Pen(Brushes.Red, 4), new Point(300, 0), new
Point(300, 300));

//road (bottom-left)
g.DrawLine(new Pen(Brushes.Red, 4), new Point(0, 400), new
Point(300, 400));
g.DrawLine(new Pen(Brushes.Red, 4), new Point(300, 700), new
Point(300, 400));

//road (top-right)
g.DrawLine(new Pen(Brushes.Red, 4), new Point(400, 0), new
Point(400, 300));
g.DrawLine(new Pen(Brushes.Red, 4), new Point(700, 300), new
Point(400, 300));

//road (bottom-right)
g.DrawLine(new Pen(Brushes.Red, 4), new Point(700, 400), new
Point(400, 400));
g.DrawLine(new Pen(Brushes.Red, 4), new Point(400, 400), new
Point(400, 700));
```

Для розділення дороги на полоси, біло намальовано штриховані лінії

```
//dash (top)
g.DrawLine(new Pen(Brushes.Red, 4), new Point(350, 0), new
Point(350, 25));
g.DrawLine(new Pen(Brushes.Red, 4), new Point(350, 50), new
Point(350, 75));
g.DrawLine(new Pen(Brushes.Red, 4), new Point(350, 100), new
Point(350, 125));
g.DrawLine(new Pen(Brushes.Red, 4), new Point(350, 150), new
Point(350, 175));
g.DrawLine(new Pen(Brushes.Red, 4), new Point(350, 200), new
Point(350, 225));
g.DrawLine(new Pen(Brushes.Red, 4), new Point(350, 250), new
Point(350, 275));

//dash (bottom)
g.DrawLine(new Pen(Brushes.Red, 4), new Point(350, 700), new
Point(350, 675));
g.DrawLine(new Pen(Brushes.Red, 4), new Point(350, 650), new
Point(350, 625));
g.DrawLine(new Pen(Brushes.Red, 4), new Point(350, 600), new
Point(350, 575));
g.DrawLine(new Pen(Brushes.Red, 4), new Point(350, 550), new
Point(350, 525));
```

```
g.DrawLine(new Pen(Brushes.Red, 4), new Point(350, 500), new
Point(350, 475));
g.DrawLine(new Pen(Brushes.Red, 4), new Point(350, 450), new
Point(350, 425));

//dash (left)
g.DrawLine(new Pen(Brushes.Red, 4), new Point(0, 350), new
Point(25, 350));
g.DrawLine(new Pen(Brushes.Red, 4), new Point(50, 350), new
Point(75, 350));
g.DrawLine(new Pen(Brushes.Red, 4), new Point(100, 350), new
Point(125, 350));
g.DrawLine(new Pen(Brushes.Red, 4), new Point(150, 350), new
Point(175, 350));
g.DrawLine(new Pen(Brushes.Red, 4), new Point(200, 350), new
Point(225, 350));
g.DrawLine(new Pen(Brushes.Red, 4), new Point(250, 350), new
Point(275, 350));

//dash (right)
g.DrawLine(new Pen(Brushes.Red, 4), new Point(675, 350), new
Point(700, 350));
g.DrawLine(new Pen(Brushes.Red, 4), new Point(625, 350), new
Point(650, 350));
g.DrawLine(new Pen(Brushes.Red, 4), new Point(575, 350), new
Point(600, 350));
g.DrawLine(new Pen(Brushes.Red, 4), new Point(525, 350), new
Point(550, 350));
g.DrawLine(new Pen(Brushes.Red, 4), new Point(475, 350), new
Point(500, 350));
g.DrawLine(new Pen(Brushes.Red, 4), new Point(425, 350), new
Point(450, 350));
```

А також було намальовано світлофори, у вигляді коло

```
//light (top-left)
g.FillEllipse(new SolidBrush(Color.Green), 275, 275, 20, 20);

//light (bottom-right)
g.FillEllipse(new SolidBrush(Color.Green), 405, 405, 20, 20);

//light (top-right)
g.FillEllipse(new SolidBrush(Color.Red), 405, 275, 20, 20);

//light (bottom-left)
g.FillEllipse(new SolidBrush(Color.Red), 275, 405, 20, 20);
```

Для зміни кольору світлофора було написана подія таймеру, в якій відповідно від значення змінюються кольори світлофорів

```
if (10<timer && timer <= 240 )
{
    //light (top-left)
    g.FillEllipse(new SolidBrush(Color.Green), 275, 275, 20,
20);

    //light (bottom-right)
    g.FillEllipse(new SolidBrush(Color.Green), 405, 405, 20,
20);

    //light (top-right)
    g.FillEllipse(new SolidBrush(Color.Red), 405, 275, 20,
20);

    //light (bottom-left)
    g.FillEllipse(new SolidBrush(Color.Red), 275, 405, 20,
20);
}
else if ((240<timer && timer<260) || timer<=10 || timer>=490)
{
    //light (top-left)
    g.FillEllipse(new SolidBrush(Color.Orange), 275, 275, 20,
20);

    //light (bottom-right)
    g.FillEllipse(new SolidBrush(Color.Orange), 405, 405, 20,
20);

    //light (top-right)
    g.FillEllipse(new SolidBrush(Color.Orange), 405, 275, 20,
20);

    //light (bottom-left)
    g.FillEllipse(new SolidBrush(Color.Orange), 275, 405, 20,
20);
}
else
{
    //light (top-left)
    g.FillEllipse(new SolidBrush(Color.Red), 275, 275, 20,
20);

    //light (bottom-right)
    g.FillEllipse(new SolidBrush(Color.Red), 405, 405, 20,
20);
```

```
    //light (top-right)
    g.FillEllipse(new SolidBrush(Color.Green), 405, 275, 20,
20);

    //light (bottom-left)
    g.FillEllipse(new SolidBrush(Color.Green), 275, 405, 20,
20);
}
```

В самому кінці, відповідно від вибраного варіанту події, відбуваються відповідні анімаційні події.

4.3 Опис функціоналу

Для запуску програми необхідно перейти за наступним посиланням (рис. 4.1) та запустити файл

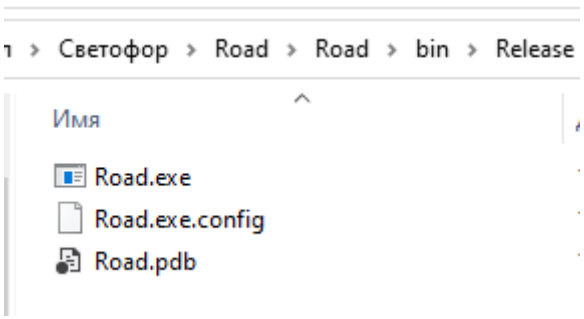


Рисунок 4.1 – шлях до файлу

Запустивши файл перед вами з’явиться стартове вікно програми(рис. 4.2)

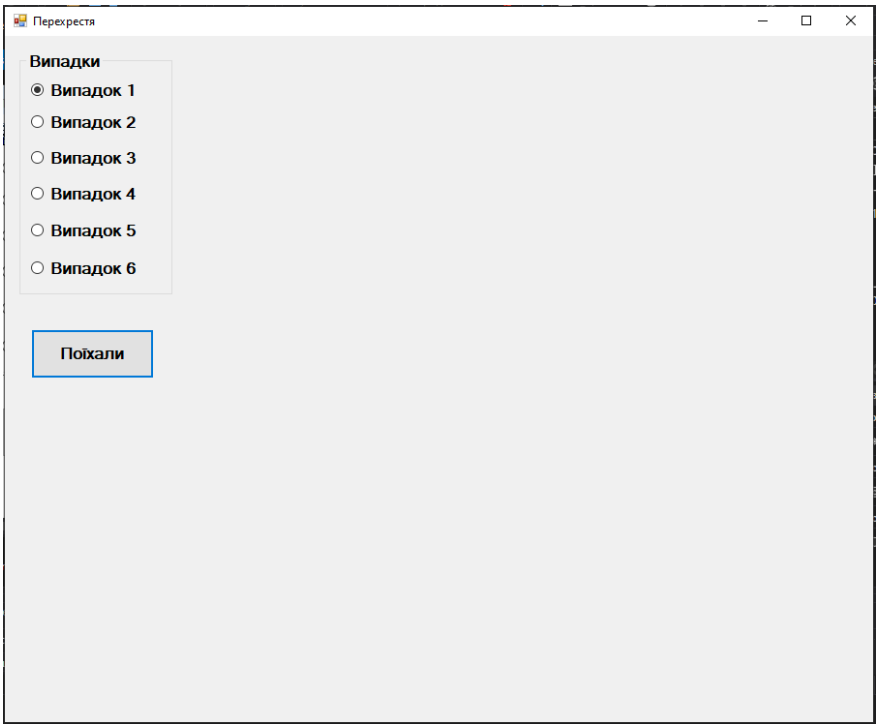


Рисунок 4.2 – Стартове вікно програми

Далі можна обрати різні випадки які нас цікавлять (рис 4.3-4.8). Перший випадок (рис. 4.3)це пусте перехрестя, на якому працює світлофори, і ведеться таймер до зміни кольору світлофора.

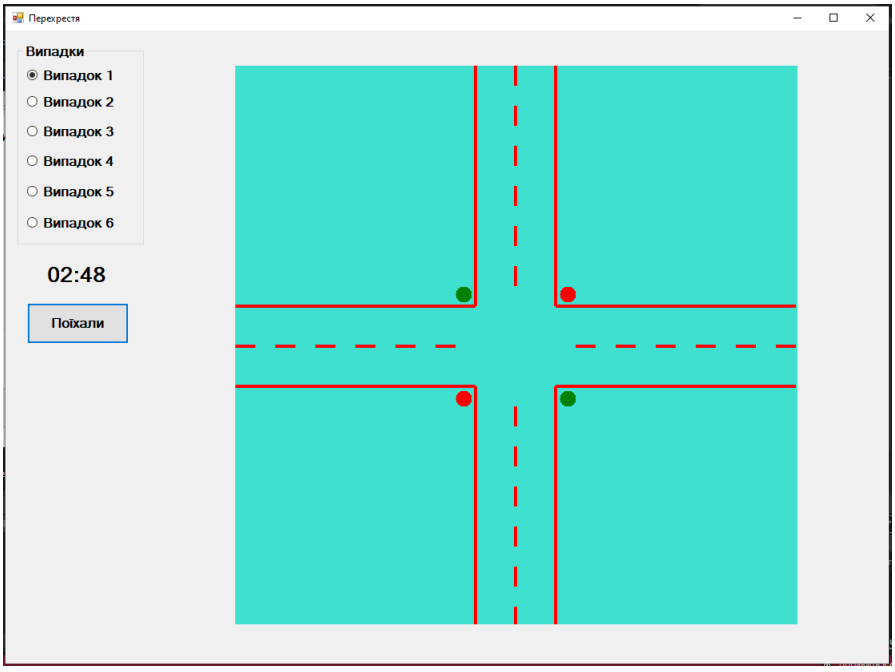


Рисунок 4.3 – Випадок №1

У другому випадку реалізовано поява машин з лівої сторони. Також реалізована можливість додавання авто (рис. 4.4).

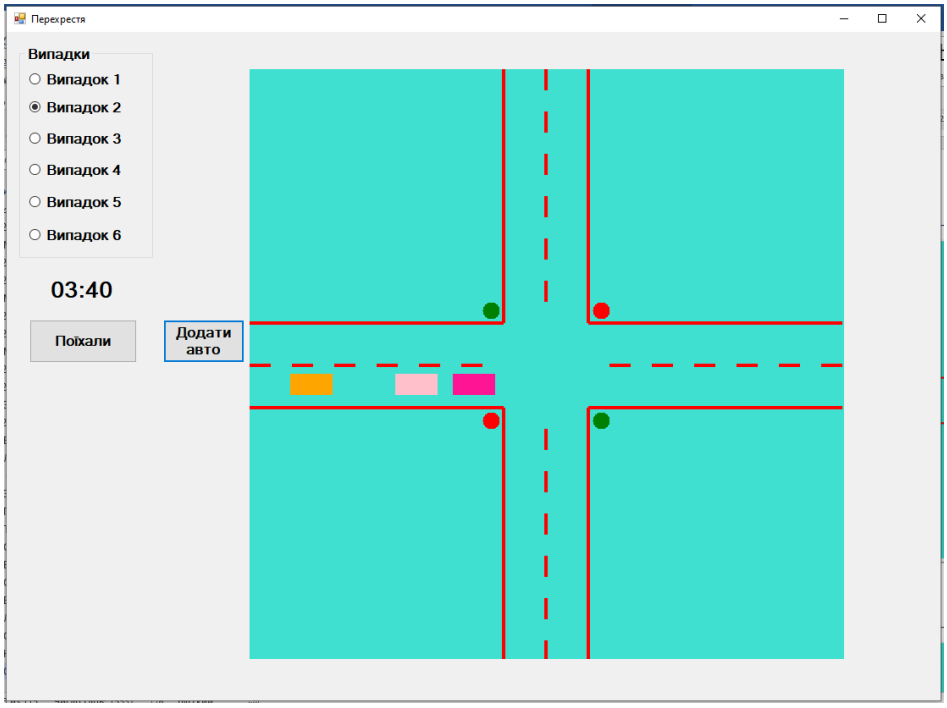


Рисунок 4.4 – Випадок №2

У третьому випадку реалізована поява машин з двох сторін. Також є можливість додавати авто з різних сторін (рис. 4.5).

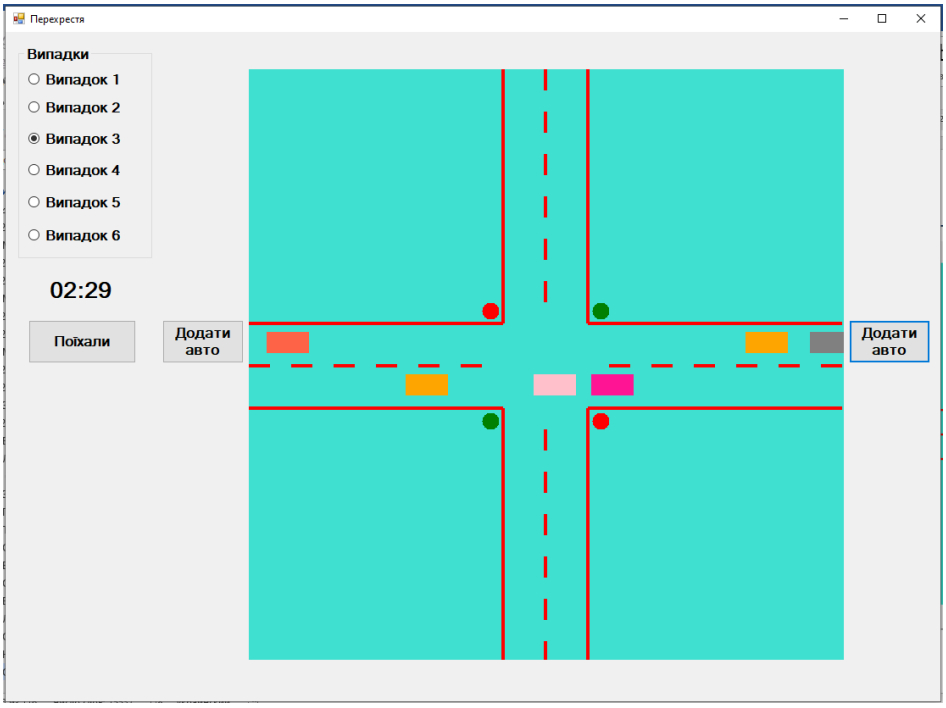


Рисунок 4.5 – Випадок №3

У четвертому випадку, ще додана можливість додавати авто зверху (рис. 4.6).

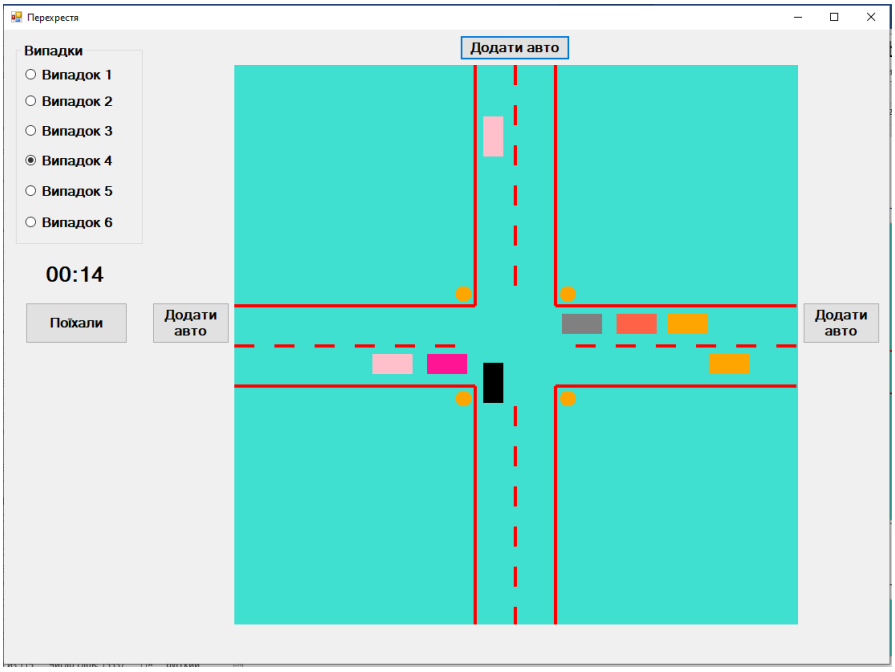


Рисунок 4.6 – Випадок №4

У п’ятому варіанті авто їздять з усіх чотирьох сторін. А також є можливість додавати авто(рис. 4.7). В залежності в кількості авто з якої сторони, час світлофора змінюється.

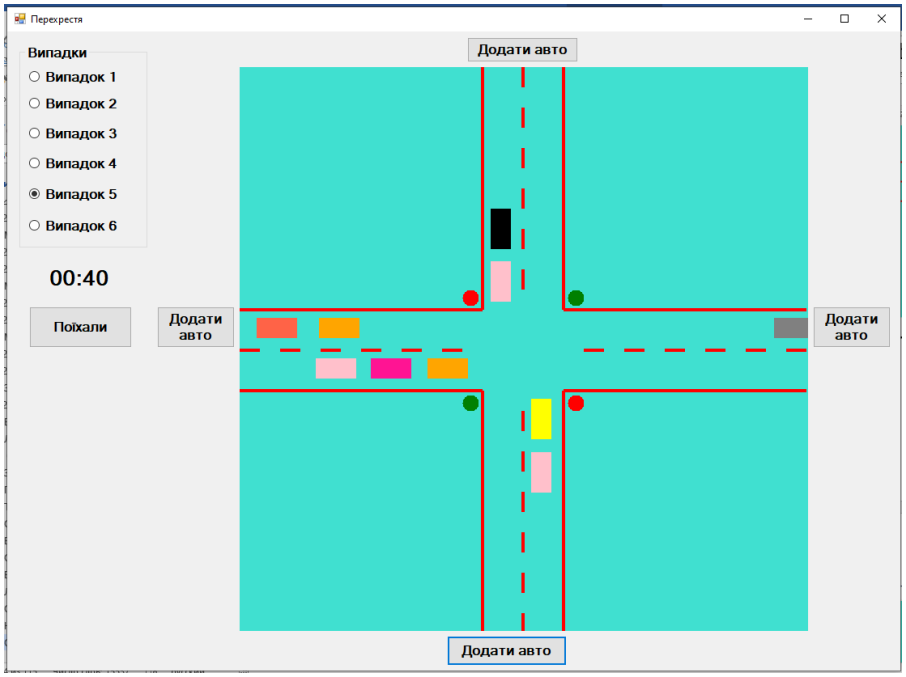


Рисунок 4.7 – Випадок №5

В останньому випадку, реалізовано рух автівки з поворотами.

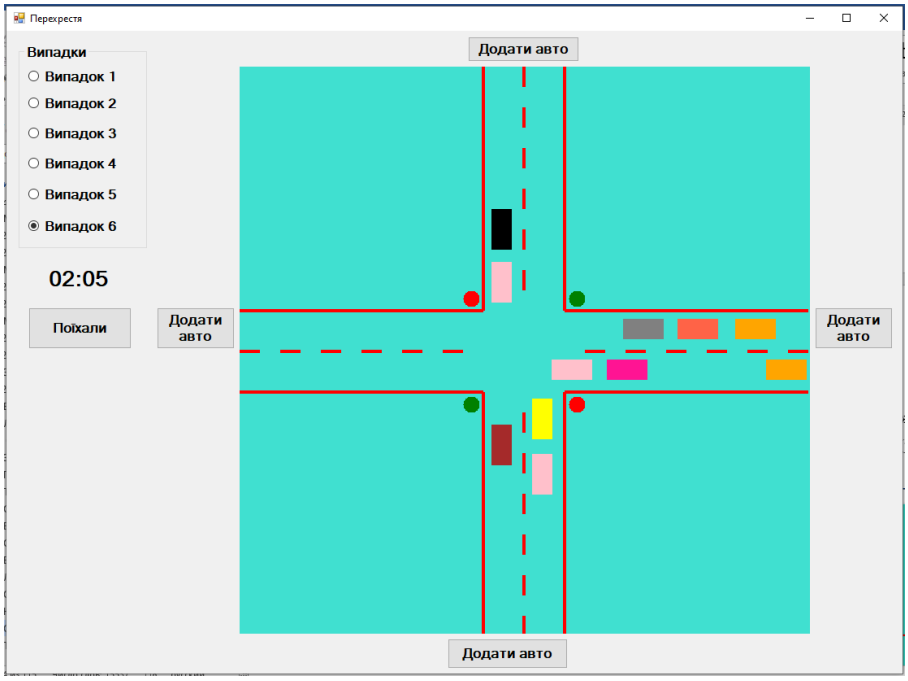


Рисунок 4.8 – Випадок №6

4.4 Тестування

Під час тестування усі виникаючі помилки одразу ж виправлялися. Нижче наведено декілька тестових прикладів роботи програми (рис. 4.9 - 4.13).

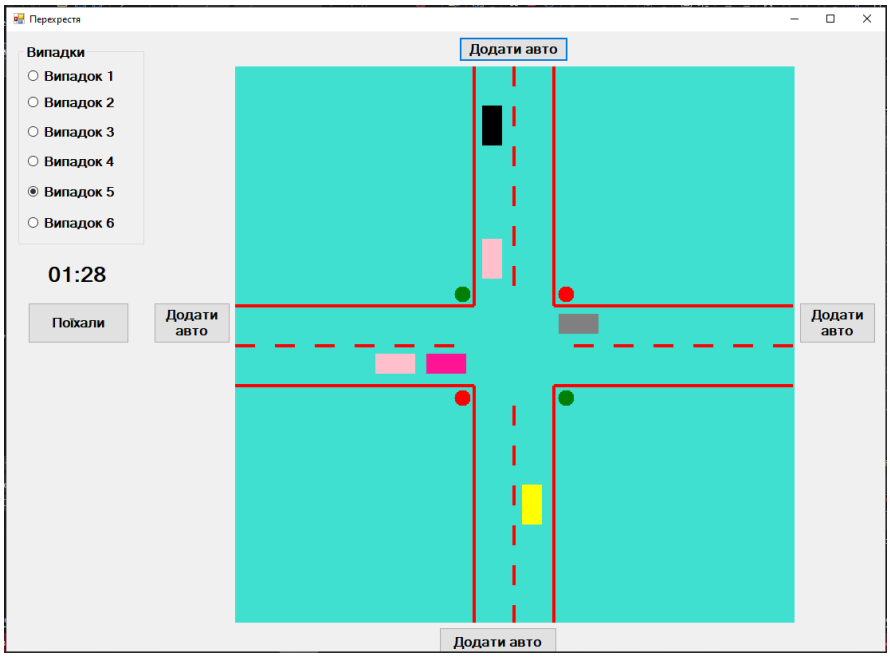


Рисунок 4.9 – тестовий приклад роботи програми

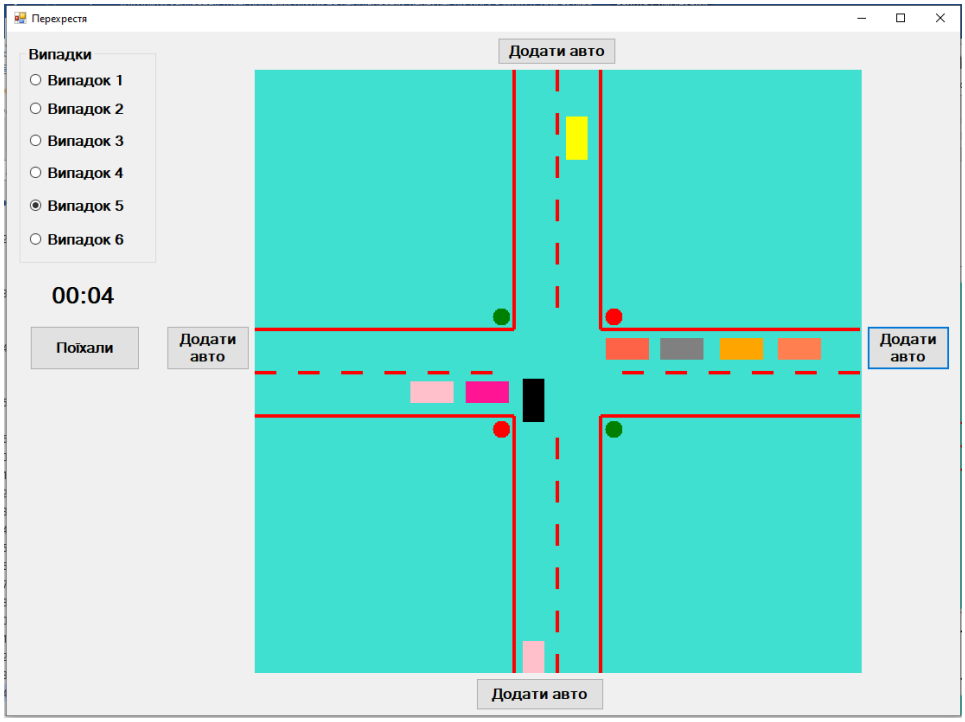


Рисунок 4.10 – тестовий приклад роботи програми

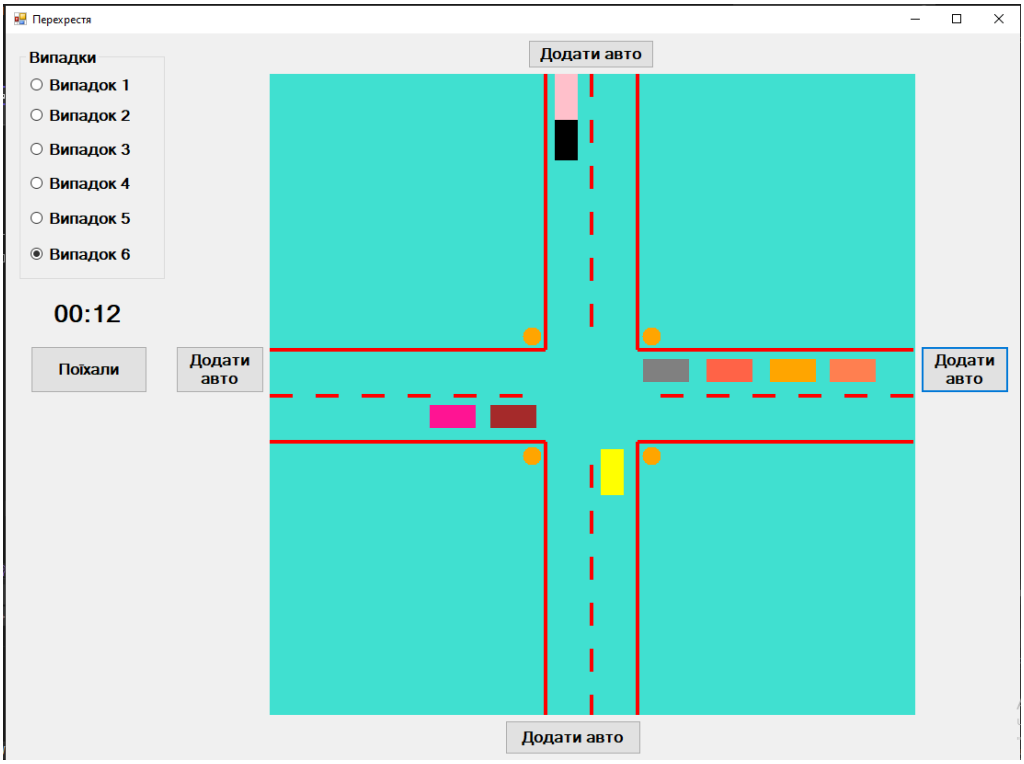


Рисунок 4.11 – тестовий приклад роботи програми

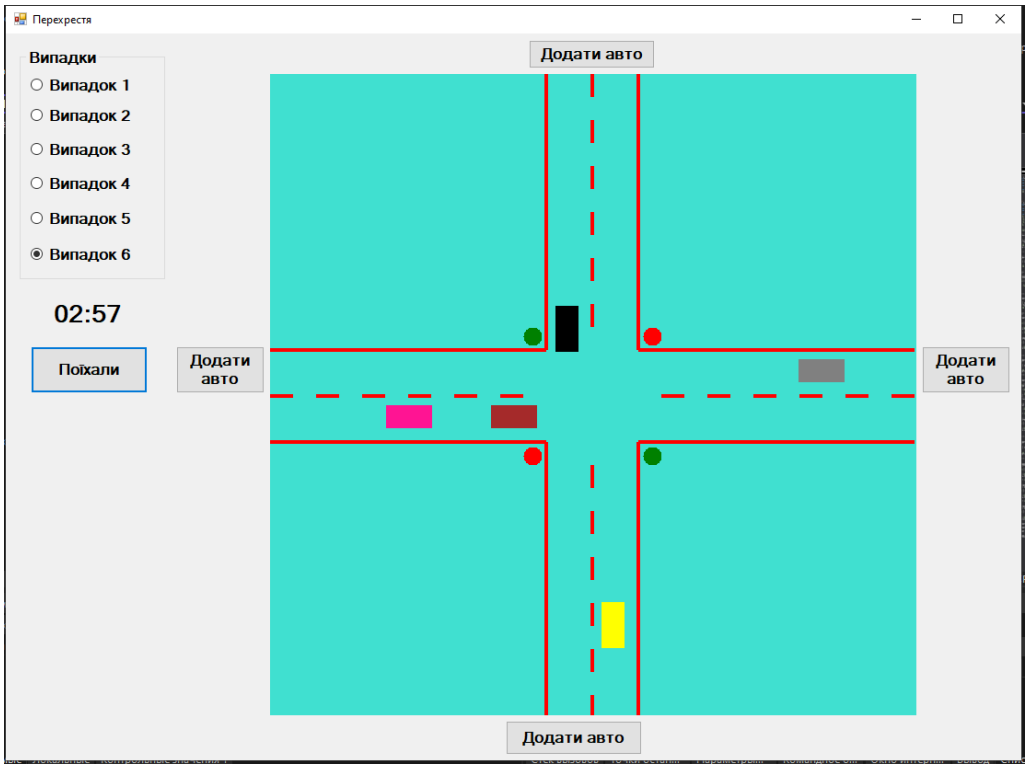


Рисунок 4.12 – тестовий приклад роботи програми

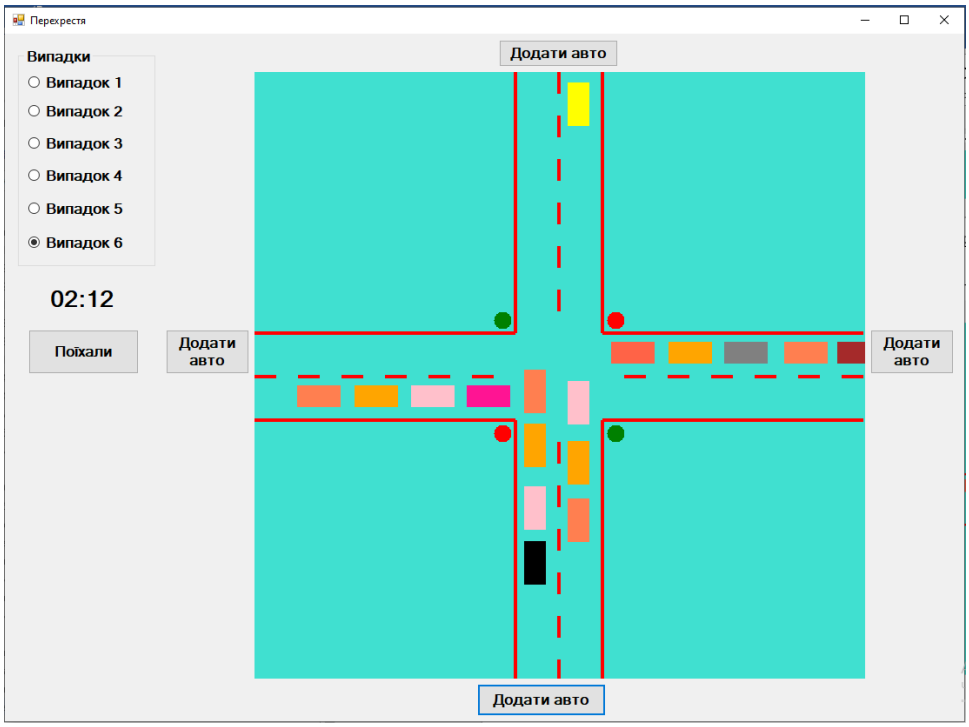


Рисунок 4.13 – тестовий приклад роботи програми

4.5 Висновки до розділу 4

В четвертому розділі розповідається про етапи розробки та створення програмного додатку. Описується алгоритм роботи програми. Також описується функціонал програми та тестування програми.

РОЗДІЛ 5. ЕКОНОМЧНІ РОЗРАХУНКИ

5.1 Організація та планування робіт

Таблиця 5.1 – Перелік робіт та тривалість їх виконання

Етапи роботи	Виконавці	Завантаження виконавців
Постановка цілей та завдань	НК	НК – 100%
Складання та затвердження ТЗ	НК, С	НК – 100% С – 50%
Підбір та вивчення матеріалів за тематикою	НК, С	НК – 20% С – 100%
Розробка календарного плану	НК, С	НК – 100% С – 20%
Пошук аналогів та обговорення їх переваг та недоліків	НК, С	НК – 10% С – 100%
Вибір бібліотеки для створення моделі	НК, С	НК – 30% С – 80%
Виявлення та класифікація факторів, що впливають на пропускну спроможність транспортних вузлів.	С	С – 100%
Розробка моделі	С	С – 100%
Оптимізація моделі	НК, С	НК – 30% С – 100%
Оформлення розрахунково-пояснювальної записки	С	С – 100%
Оформлення графічного матеріалу	С	С – 100%
Підбиття підсумків	НК, С	НК – 60% С – 100%

Для виконання перелічених у таблиці 5.1 робіт потрібні спеціалісти:

- Студент – у його ролі діють виконавець ВКР;
- Науковий керівник

Тривалість етапів робіт

Розрахунок тривалості етапів робіт здійснюється дослідно-статистичним методом експертним способом за формулою:

$$t_{оч} = \frac{3t_{min} + 2t_{max}}{5} \quad (5.1)$$

де t_{min} - Мінімальна тривалість роботи, дн.;

t_{max} - Максимальна тривалість роботи, дн.

5.2 Розрахунок кошторису витрат за виконання проекту

Розробка та оптимізація моделі транспортної системи міста велася з використанням програмного забезпечення, ліцензія на яке надається безкоштовно студентам та співробітникам університету, на власному ноутбуці чи комп'ютерах, без оренди приміщення: вдома чи в університеті. Тому розрахунок кошторисної вартості виконання проекту проводитиметься за такими статтями витрат:

- сировина, матеріали та покупні вироби;
- заробітна плата;
- соціальний податок;
- витрати на електроенергію (без освітлення);
- амортизаційні нарахування;
- оплата послуг зв'язку;
- інші (накладні витрати) витрати.

5.2.1 Розрахунок витрат на матеріали

До цієї роботи витрат відноситимемо вартість матеріалів, які були використані під час розробки проекту. Розрахунок вартості матеріальних витрат провадиться за діючими прейскурантами або договірними цінами. У вартість

матеріальних витрат включають транспортно-заготівельні витрати (3 – 5% ціни). До цієї статті включаються витрати на оформлення документації (канцелярське приладдя, тиражування матеріалів).

Таблиця 5.2 – Розрахунок витрат за матеріали

Назва матеріалів	Ціна за од., грн.	Кількість	Сума, грн.
Папір для принтера формату А4	290	1 уп.	290
Тонер для принтера	250	1 шт.	250
Папір для принтера формату А3	600	1 шт.	600
Кольоровий тонер	250	1 шт.	250
Разом:			1390

Транспортно-заготівельні витрати (ТЗВ) становлять 5 % від

відпускної ціни матеріалів, тоді витрати на матеріали з урахуванням ТЗВ дорівнюють

$$C_{мат} = 1390 * 1.05 = 1459.5 грн.$$

5.2.2 Розрахунок заробітної плати

Так як над проектом працювали тільки науковий керівник та студент, то розрахунок заробітних плат проводитиметься лише за статтею основних заробітних плат. Заробітна плата розраховується на основі суми заробітної плати виконавця та наукового керівника. Величина витрат із заробітної плати визначається виходячи з трудомісткості виконуваних робіт та чинної системи оплати праці. До складу основної заробітної плати включається премія, що

виплачується щомісяця із фонду заробітної плати (розмір визначається Положенням про оплату праці).

Розмір місячного окладу наукового керівника отримано з відкритих даних, розміщених на офіційному сайті. Розмір місячного окладу інженерів береться як місячний оклад інженера кафедри.

Середньоденна тарифна заробітна плата розраховується за формулі:

$$ЗП_{дн-т} = МО / N \quad (5.2)$$

де МО - місячний оклад, грн.;

N – кількість робочих днів на місяць, при шестиденному робочому тижні.

Середньоденна тарифна заробітна плата наукового керівника дорівнює

$$ЗП_{дн-т} = \frac{26300}{24.91} = 1055.8 грн./доб$$

А середньоденна тарифна заробітна плата студента дорівнює

$$ЗП_{дн-т} = \frac{7864.11}{20.58} = 382.12 грн./доб$$

Витрати часу за кожним виконавцем у робочих днях взято з таблиці 5.1. Для переходу від тарифної суми заробітку виконавця, пов'язаної з участю у проекті, до повного заробітку необхідно буде тарифну суму заробітку виконавця, пов'язану з участю у проекті помножити на інтегральний коефіцієнт. Інтегральний коефіцієнт знаходиться за формулою:

$$К_и = К_{пр} К_{доп.ЗП} К_р \quad (5.3)$$

де Кпр - коефіцієнт премій, Кпр = 1,1;

Кдоп.ЗП – коефіцієнт додаткової зарплати, при шестиденному робочому тижні Кдоп.ЗП = 1,188, а при п'ятиденному робочому тижні Кдоп.ЗП = 1,113;

Кр – коефіцієнт районної надбавки, Кр = 1,3.

Результати обчислень представлені у таблиці 5.3.

Таблиця 5.3 – Витрати зарплатню

Виконавець	Оклад, грн./міс.	ЗП, грн./роб. день	Витрати часу, роб. дні	Коефіцієнт	Фонд з/плати, грн.
НК	26300	1055,8	22	1,699	39 463,69
С	7 864,11	382,12	95	1,62	50 497,67
Разом:					89 961,36

5.2.3 Розрахунок внеску до соціальних фондів

Стаття включає внески до позабюджетних фондів. Внесок до соціальних фондів встановлено у розмірі 30,2% від заробітної плати. Розмір внеску розраховуються за такою формулою:

$$C_{соц} = C_{зп} \cdot 0{,}302 \quad (5.4)$$

де $C_{зп}$ – розмір зарплати.

Підставивши необхідні значення у формулу 5.4 отримаємо:

$$C_{соц} = 89961{,}36 \cdot 0{,}302 = 27168{,}33 грн.$$

5.2.4 Розрахунок витрат за електроенергію

Витрати на електроенергію розраховуються за такою формулою:

$$C_{ел.об} = P_{об} \cdot t_{об} \cdot Ц_{ел} \quad (5.5)$$

де

$P_{об}$ - потужність, що споживається обладнанням, кВт;

$t_{об}$ - час роботи обладнання, година;

$Ц_{ел}$ – тариф на 1 кВт/годину.

Час роботи устаткування обчислюється з урахуванням підсумкових даних таблиці 5.2 для студента з розрахунку, що тривалість робочого дня дорівнює 8 годин.

$$t_{об} = T_{РД} \cdot K_t \quad (5.6)$$

де K_t - Коефіцієнт використання обладнання за часом, K_t=0.9.

Потужність, що споживається обладнанням, визначається за такою формулою:

$$P_{об} = P_{ном} \cdot К_С \quad (5.7)$$

де K_C – коефіцієнт завантаження;

$P_{ном}$ - Номінальна потужність обладнання, кВт. Для технологічного обладнання малої потужності.

Таблиця 5.4 - Витрати на електроенергію технологічну

Найменування обладнання	Час роботи обладнання $t_{ОБ}$, годину	Споживана потужність $P_{ОБ}$, кВт	Витрати $Е_{ОБ}$, грн.
Персональний комп'ютер студента	684	0,09	101,08
Разом:			101,08

5.2.5 Розрахунок амортизаційних витрат

Для розрахунку амортизаційних витрат використовується формула:

$$C_{АМ} = \frac{Н_А \cdot Ц_{ОБ} \cdot t_{рф} \cdot n}{F_д} \quad (5.7)$$

Де $Н_А$ - Річна норма амортизації одиниці обладнання;

$Ц_{ОБ}$- балансова вартість одиниці обладнання з обліків ТЗР, вартість ПК інженера - 19290 грн.;

$t_{рф}$ – фактичний час роботи обладнання під час виконання проекту;

n - число задіяних однотипних одиниць обладнання;

$F_Д$ – дійсний річний фонд часу роботи відповідного обладнання,

понад 2 до 3 років включно.

У цій роботі приймемо срок амортизації =2,5 року.

Таким чином,

$$C_{АМ}(ПК) = \frac{0{,}4 \cdot 19290 \cdot 760 \cdot 1}{2384} = 2459.76грн$$

Разом нараховано амортизації 2459,76 грн.

5.2.6 Розрахунок витрат на послуги зв'язку

Витрати на послуги зв'язку визначені наявністю підключення до Інтернету на комп'ютері, використаному в даній роботі.

Щомісячна оплата, згідно з тарифом, становить 400 гривень. Відповідно до таблиці 5.2, трудомісткість виконуваної задачі становить чотири календарні місяці.

Отже, сума витрат послуги зв'язку становить 4*400 = 1600 грн.

Загальна сума видатків $C_{СВ} = 1600$

5.2.7 Розрахунок інших витрат

Інші витрати слід прийняти рівними 10% суми всіх попередніх витрат. Вони знаходяться за формулою:

$$C_{проч} = (C_{мат} + C_{ЗП} + C_{соц} + C_{ел.об.} + C_{АМ} + C_{св}) \cdot 0.1 \qquad (5.8)$$

де $C_{мат}$ - Витрати на матеріали, грн.;

$C_{ЗП}$ - Основна заробітна плата, грн.;

$C_{соц}$ - Витрати єдиний соціальний податок, грн.;

$C_{ел.об}$ - Витрати на електроенергію, грн.;

$C_{АМ}$ - Амортизаційні витрати, грн.;

$C_{св}$ - Витрати послуги зв'язку, грн.

Підставивши отримані вище результати, отримаємо:

$C_{проч}$ = (1459,5 + 89 961,36 + 27 168,33 + 101,08 + 2459,76 + 1600) ·0,1 =

= 12252,4 грн.

5.2.8 Розрахунок загальної собівартості розробки

Провівши розрахунок за всіма статтями кошторису витрат за розробку, можна визначити загальну собівартість проекту.

Таблиця 5.5 – Кошторис витрат за розробку проекту

Стаття витрат	Умовне позначення	сума, грн.
Матеріали та покупні вироби	$C_{мат}$	1459,50
Основна заробітна плата	$C_{ЗП}$	89 961,36
Відрахування до соціальних фондів	$C_{соц}$	27168,33

Витрати на електроенергію	$С_{ел.об.}$	126,20
Амортизаційні відрахування	$С_{АМ}$	2459,76
Витрати на послуги зв'язку	$С_{св}$	1600
Інші витрати	$С_{проч}$	12 252,4
Разом:		135 027,55

Таким чином, витрати на розробку склали С = 135027,55 грн.

5.2.9 Розрахунок прибутку

Прибуток приймемо у розмірі 20% повної собівартості проекту. У цьому проекті вона становить 27 005,51 грн. від видатків на розробку проекту.

5.2.10 Розрахунок ПДВ

ПДВ 20% сплачується платником податків до бюджету, якщо його діяльність не потрапляє до переліків, де діє знижена пільгова ставка.

ПДВ зверху становить 20% від суми витрат на розробку та прибутку.

У нашому випадку це (135027,55 + 27005,51) 0,2 = 32 406,61 грн.

5.2.11 Ціна розробки НДР

Ціна дорівнює сумі повної собівартості, прибутку та ПДВ, у нашому випадку

ЦНДР (КР) = 135 027,55 + 27 005,51 + 32 406,61 = 194 439,67 грн.

5.3 Оцінка економічної ефективності проекту

Впровадження цього проекту дозволить покращити пропускну спроможність транспортних мереж, оптимізувати витрати на паливо та покращити екологічну обстановку. Завдяки відкритому коду, надалі модель може використовуватись у будь-якій геоінформаційній системі чи системі підтримки прийняття рішень.

5.3.1 Визначення терміну окупаємості

Термін окупності використовується як показник ефективності проекту. Чим менший термін окупності, тим ефективніший проект. Для розрахунку використовується формула:

$$РР = \frac{С}{ПР_ч} \tag{5.9}$$

де С - Витрати на розробку, грн.;

$ПР_ч$ - Річний чистий прибуток, грн.

Підставивши отримані вище результати, отримаємо:

$$РР = \frac{135027.55}{27005.51} = 5 років$$

5.3.2 Оцінка науково-технічного рівня НДР

Науково-технічний рівень характеризує вплив проекту на рівень та динаміку забезпечення науково-технічного прогресу в даній галузі. Для оцінки наукової цінності, технічної значущості та ефективності, що плануються та виконуються НДР, використовується метод бальних оцінок. Бальна оцінка

полягає в тому, що кожному фактору за прийнятою шкалою надається певна кількість балів.

Узагальнену оцінку проводять за сумою балів за всіма показниками. На її основі робиться висновок про доцільність НДР.

Сутність методу полягає в тому, що на основі оцінок ознак роботи визначається інтегральний показник (індекс) її науково-технічного рівня за формулою:

$$I_{НТУ} = \sum_{i=1}^{3} R_i \cdot n_i \qquad (5.10)$$

де $I_{НТУ}$ – інтегральний індекс науково-технічного рівня;

R_i – ваговий коефіцієнт і ознаки науково-технічного ефекту;

n_i - Кількісна оцінка і-го ознаки науково-технічного ефекту, в балах.

Приватні оцінки рівня ni та їх коротке обґрунтування наведено в таблиці 5.6.

Таблиця 5.6 - Оцінки науково-технічного рівня НДР

Значимість	Чинник НТУ	Рівень фактора	Вибраний бал	Обґрунтування обраного бала
0,4	Рівень новизни	Нова	8	Надає можливість потенційному користувачеві змоделювати будь-який транспортний вузол з будь-яким навантаженням, що значно перевищує ефективність статистичні дані, т.к. Крім простого прогнозування, ця модель може

				використовуватися в системах підтримки прийняття рішень.
0,1	Теоретичний рівень	Розробка моделі	6	Вивчення внутрішнього коду, використання бібліотеки.
0,5	Можливість реалізації	Протягом перших років	10	Складні методи та опрацювання дисципліни обслуговування кожного типу перехрестя та переходу.

Інтегральний показник науково-технічного рівня для цього проекту становить:

$$I_{НТУ} = 0.4 \cdot 8 + 0.1 \cdot 6 + 0.5 \cdot 10 = 8.8$$

За отриманими даними можна зробити висновок, що проект має високий рівень науково-технічного ефекту.

ВИСНОВКИ

В результаті дослідницької роботи розроблено модель транспортного вузла, на основі якої побудовано модель системи управління транспортними потоками. Після аналізу та огляду було вирішено оптимізувати модель системи управління транспортним потоком шляхом імітації роботи вузлових світлофорів та подальшого налаштування режимів їх роботи.

Перевага такої оптимізації полягає не тільки в розвантаженні транспортного вузла з попередньо встановленими параметрами розподілу вхідних запитів типу «Пішохід» і «Автомобіль», але і в збереженні стабільної роботи вузла при переході цих параметрів до збільшення навантаження.

Отримана модель перехрестя є дуже гнучкою і може бути легко перетворена на будь-який тип перехрестя з будь-яким навантаженням на нього.

Перспективи успішного використання моделі включають її використання в поєднанні з системою збору статистики перехресть на основі розпізнавання образів. Визначаючи частоту прибуття автомобілів або пішоходів, ви можете налаштувати світлофори залежно від дня тижня, сезону або часу доби.

Успішне впровадження такої системи управління дорожнім рухом дозволить не тільки скоротити витрати на паливо та час у дорозі, але й надалі може бути використано для побудови систем підтримки прийняття рішень для автомобільних та пасажирських маршрутів.

У майбутньому ця модель може бути розширена шляхом додавання інших частин мережі. Це дозволить аналізувати транспортні ситуації, виявляти проблемні зони та приймати рішення щодо їх усунення.

Результати експерименту показують, що контроль розглянутих перехресть в цілому є ефективним, з деякими незначними недоліками, які можна виправити для покращення умов руху.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Макарова І.В., Хабабуллін Р.Г., Шубенкова К.А. Удосконалення управління транспортними потоками міста з використанням імітаційного моделювання // 4-а Всеросійська науково-практична конференція з імітаційного моделювання ІММОД 2009/С.-П., 2009

2. Швецов В.І., Алієв А.С. Математичне моделювання завантаження транспортних мереж. -М.: М'яка обкладинка, 2003. - 62с.

3. Поняття моделі та моделювання [Електронний ресурс], режим доступу URL: http://studopedia.ru/3_36006_ponyatiya-modeli-i-modelirovaniya.html (дата звернення: 13.10.2022)

4. Швецов В.І. Математичне моделювання транспортних потоків. -М.: Інститут системного аналізу РАН, 2003. - 52 с.

5. Антипробочна терапія [Електронний ресурс], режим доступу URL: http://www.arterylite.ru/library/page/3 (дата звернення: 13.10.2022)

6. Інтелектуальна транспортна система для великих міст [Електронний ресурс], режим доступу URL: http://studik.net/intelektualnaya- transportnaya-sistema-dlya-krupnyx-gorodov (дата звернення: 13.10.2022)

7. Розпорядження Федеральної дорожньої агенції від 27 лютого 2013р. №236-р «Про видання та застосування ОДМ 218.6.003-2011 «Методичні рекомендації щодо проектування світлофорних об'єктів на автомобільних дорогах»: [Електронний ресурс], режим доступу URL: http://www.garant.ru/products/ipo/prime/doc/70226672/ (дата звернення: 13.10.2022)

8. Якимів М.Р. Транспортне планування: створення транспортних моделей міст: Автореф. дис. Канд. наук. - Москва: 2013

9. Теорія масового обслуговування [Електронний ресурс], режим доступу URL: http://window.edu.ru/resource/124/47124/files/sssu068/ (дата звернення: 13.10.2022)

10. Теорія масового обслуговування [Електронний ресурс], режим доступу до URL: http://e-lib.kemtipp.ru/uploads/08 (дата звернення: 13.10.2022)

11. 3.Марківські процеси [Електронний ресурс], режим доступу URL: http://e-biblio.ru/book/bib/06_management/teor_mass_obslug/158.9.13.html вільний (дата звернення: 13.10.2022)

12. MATLAB [Електронний ресурс], режим доступу URL: http://matlab.ru/products/matlab (дата звернення: 13.10.2022)

13. Білохвостіков І.В. Системні дослідження та оптимізація функціонування Інтернет систем з використанням мереж Петрі : Дис. ...канд. техн. наук : 05.13.01 : Краснодар, 2004 168 с.

14. Simulink [Електронний ресурс] режим доступу до URL: http://matlab.ru/products/simulink/(дата звернення: 13.10.2022)

15. SimEvents [Електронний ресурс] режим доступу до URL: http://matlab.ru/products/simevents (дата звернення: 13.10.2022)

16. Про AnyLogic [Електронний ресурс], режим доступу URL: https://www.anylogic.ru/overview (дата звернення: 13.10.2022)

17. Лагерєв Р.Ю. Сучасні принципи керування транспортними потоками на міських магістралях // Вісник ІРДТУ / №1(48) – 2011р.

18. Охорона праці. Основи безпеки життєдіяльності // www.Grandars.ru. URL: http://www.grandars.ru/shkola/bezopasnost- zhiznedeyatelnosti/ohrana-truda.html (дата звернення: 13.10.2022);

19. СНиП 23-05-95 Будівельні норми та правила Російської Федерації. Природне та штучне освітлення // URL: http://www.vashdom.ru/snip/2305-95/ (дата звернення: 13.10.2022);

20. ГОСТ ИСО 8995-2002. Висвітлення робочих систем усередині приміщень // URL: http://www.internet-law.ru/gosts/gost/5955/ (дата звернення 11.03.2017);

21. ГОСТ 12.1.038 – 82 система стандартів безпеки праці. Електробезпека, гранично допустимі значення напруги дотику та струмів // URL: http://vsegost.com/Catalog/21/21681.shtml (дата звернення: 13.10.2022);

22. Кормен Т., Лейзерсон Ч., Рівест Р. Алгоритми: побудова та аналіз, 2001.

23. Окулов З. М. Програмування алгоритмах. - М: БІНОМ. 2002 р.

24. Д.Е. Хлист. Мистецтво програмування, том 3. Пров. З англійської. М.: Видавництво Вільямс, 2007. (та ін видання)

25. Бутч Г. Об'єктно-орієнтований аналіз та проектування з прикладами додатків. Видавництво Вільямс, Санкт-Петербург, 2008

26. Бертран Мейєр. Об'єктно-орієнтоване проектування програмних систем. -М: Російське видання, 2005 р.

27. Лафор Р. Об'єктно-орієнтоване програмування на C++. СПб: Пітер, 2006.

28. Синтес А. Освойте власне об'єктно-орієнтоване програмування за 21 день. Москва4 СПб: Вільямс, 2002.

29. Річард Барас: До теорії інновацій у сфері послуг, -1986, с. -183.

30. Хазієва А. М. Сучасна державна статистика // Тенденції та перспективи розвитку статистичної науки та інформаційних технологій. Збірник наукових статей, -2013, -С. 94-98.

31. Стеценко І.В. Системи моделювання: Навч. поб. [Електронний ресурс, текст]/І.В. Стеценка; Міністерство освіти та науки України, Черкаси. Холдинг техн. ун. – Черкаси: ЧДТУ, 2010. – 399 с.

32. Навроцький, А.А. Основи алгоритмізації та програмування у середовищі Visual C++: метод дослідження. посібник/А. А. Навроцький. - Мінськ: БДУІР, 2014. - 160 с.

33. Шільдт, Г. Базовий курс C++, 3-тє вид. / Г. Шільдт. пров. з англійської мови. - М.: Видавництво Вільямс, 2015. - 624 с.

34. МакКоннелл С. Ідеальний код. Майстер-клас/С. Макконнелл. пров. з англійської мови. - М: Видавництво «Російська редакція», 2010. - 896 с.

35. Документація Visual Studio [електронний ресурс]. Режим доступу: https://docs.microsoft.com/en-us/visualstudio/ide/?view=vs-2017.

36. Стандарт кодування C++ Тодда Хоффа [електронний ресурс]. Режим доступу: http://www.possibility.com/Cpp/c++_coding_standards.pdf

37. Посібник із стилю Google C++ [електронний ресурс]. Режим доступу: https://google.github.io/styleguide/cppguide.html 7. ГОСТ 19.701-90 Схеми алгоритмів, програм, даних та систем. 8. Доманов А.Т. Стандарт підприємства СТП 01-2017/А.Т. Доманов, Н.І. Сорока. - Мінськ: БДУІР, 2017. - 169 с.

38. Введення в мови програмування С та С++ [електронний ресурс] / режим доступу: http://www.intuit.ru/studies/courses/1039/231/info

39. Верифікація ПЗ [Електронний ресурс]/режим доступу: http://www.intuit.ru/studies/courses/1040/209/info

40. Інструменти, алгоритми та структури даних [електронний ресурс]/режим доступу: http://www.intuit.ru/studies/courses/683/539/info

41. Мюссер, Д. С++ і STL: довідник / Д. Мюссер, Дж. Дерг, А. Сейні. - М.: Ред. Будинок "Вільямс", 2010 р.

42. Саттер, Г. Стандарти програмування на С++: серія "C++ In-Depth"/Г. Саттер, А. Александреску. - М.: Ред. Будинок "Вільямс", 2008 рік.

43. Седжвік, Р. Алгоритми в С ++ / Р. Седжвік. - М.: Ред. Будинок "Вільямс", 2010 р.

44. Шільдт, Г. С++: Методи програмування Шільда / Г. Шільдт. - М.: Ред. Будинок "Вільямс", 2008 рік.

45. Девіс, С. С ++ для чайників / С. Девіс. - 6-те вид. - М.: Ред. Будинок "Вільямс", 2010 р.

46. Ліберті, Дж. Вивчіть С++ самостійно за 21 день / Дж. Ліберті, Дж. Бредлі. - 5-те вид. - М.: Ред. Будинок "Вільямс", 2010 р.

47. Страуструп, Б. Програмування на прикладах на C++: принципи та практика / Б. Страуструп. - М.: Ред. Будинок "Вільямс", 2010 р.

48. Шильдт, Г. Повний посібник з C++ / Г. Шильдт. - М.: Ред. Будинок "Вільямс", 2010 р.

49. Транспортна система [Електронний ресурс]. URL: https://ua.wikipedia.org/wiki/Транспортна_система, вільний. Дата звернення: 17.10.2022р.

50. Дорожньо-транспортний комплекс [Електронний ресурс]. URL: http://www.transportall.ru/info/perevozki/291/2125.html, вільний. Дата звернення: 16.10.2022р.

51. Автомобілізація [Електронний ресурс]. URL: http://knowledge.allbest.ru/transport/2c0a65635b2ad69a5c43a89421216c37_0.html, Дата звернення: 15.10.2022р.

52. Правила проїзду перехресть [Електронний ресурс]. URL: http://pddmaster.ru/pdd/pravila-proezda-perekrestkov-chast-1-chto-takoeperekrestok.html, Дата звернення: 11.10.2022р.

53. Моделювання проїзду перехрестя [Електронний ресурс]. URL: https://domashke.com/referati/referaty-po-transportu/referat-modelirovanie-proezdaperekrestka, Дата звернення: 11.10.2022р.

54. Імітаційне моделювання [Електронний ресурс]. URL: http://www.anylogic.ru/use-of-simulation, Дата звернення: 10.10.2022р.

55. Клінковштейн, Г.І. Організація дорожнього руху: Підручник для автомобільно-дорожніх вузів та факультетів. - 2-ге вид., перероб. та дод. - М.: Транспорт, 2004. 240с.

56. Кременець, Ю.А., Печерський М.П., Афанасьєв М.Б. Технічні засоби регулювання дорожнього руху: Навч. для вузів. - М.: ІКЦ «Академкнига», 2005. - 279с.

57. Лобанов Є. М. Транспортне планування міст: Підручник для студентів вузів/Є. М. Лобанов. - М: Транспорт, 1990. 240 с.

58. Коноплянко В. І. Організація та безпека дорожнього руху / В. І. Коноплянко. МАДИ.- М.: 2007. - 240 с.

59. Ільїна, Н. В. Економічне обґрунтування заходів щодо підвищення безпеки руху: Метод. вказівка / Н. В. Ільїна. - Красноярськ: ІСЦ КДТУ, 2003. - 27 с.

60. ДЕРЖСТАНДАРТ Р 52290-2004 Технічні засоби організації дорожнього руху. Дорожні знаки. Загальні вимоги. - 125 с.

61. ГОСТ Р 52289-2004. Технічні засоби організації дорожнього руху. Правила застосування дорожніх знаків, розмітки, світлофорів, дорожніх огорож та напрямних пристроїв

62. СНіП 2.05.02-85. Будівельні норми і правила. Конструктивні параметри дороги. Правила дорожнього руху. Науково-видавниче підприємство. 2-Р - М.: 1994. - 63 с.

63. Babicheva T.S. Numerical methods for modeling of traffic flows at research and optimization of traffic on the signal-controlled road intersections. MIPT Proceedings. 2015. V. 7, № 1. P. 132–144.

64. Afanasieva L.G. Rudenko I.V. Service systems $GI|G|\infty$ and applications to transport models analisys. Probability theory and applications. 2012. V. 57, № 2. P. 1–27.

65. Afanasieva L.G. Bulinskaya E.V. Matemathical models of transport systems based on queuing theory. MIPT Proceedings. 2010. V. 2, № 4, P. 6–21.

66. Nagui Rouphail, Andrzej Tarko, Jing Li Traffic flow at signalized intersections. TRB Special Report 165: Traffic Flow Theory. 1975.

67. Gasnikov A.V., Dorn Y.V., Nesterov Y.E., Shpirko S.V. About three-stage version of the model the transient dynamics traffic. Mathematical modelling. 2014. in the publication process.

68. Trapeznikova M.A., Furmanov I.R., Churmanova N.G., Lipp R. Modelling of multilane traffic on the basis of cellular automati. Mathematical modelling. 2011. V. 23, № 6. P. 133–146.

69. Modeling and forecasting traffic flow at the intersection of st. Uralskaya–st. Krupskoy. [HTML] (http://road.perm.ru/index.php?id=894).

70. Математическое моделирование динамики транспортных потоков мегаполиса, Семенов В.В., стаття

71. Daganzo C.F. Remarks on Traffic Flow Modeling and its Applications // Dept. of Civil and Environmental Engeneering University of California, Berkeley, стаття

72. Studying the ebb and flow of stop-and-go; Los Alamos Lab using cold war tools to scrutinize traffic patterns alan sipress washington post staff writer" Thursday, August 5, 1999, http://www.science.com, стаття

73. Static and Dynamic Traffic Assignment with Recurrent Neural Networks, Paul Mathias, Siemens AG, ATD SV PSM, Minich, and Department of Computer Sciense 4, Aachen University of Technology

74. Prigogine, I., Herman R. Kinetic theory of vehicular traffic. – New York: Elsevier, 1971.

75. Helbing, D. Gas-kinetic derivation of Navier-Stokes-like traffic equations // Phys. Rev. E. 1996. Vol.53. № 3. P.2366 – 2381.

76. Payne, H. J. Models of Freeway Traffic and Control // Mathematical Models of Public Systems. La Jolla: Simulation Council. CA. 1971. Vol.1. P. 51.

77. Kühne, R. D., Beckschulte, R. In Proceedings of 12th International Symposium on the Theory of Traffic Flow and Transportation / edited by C.F. Daganzo. Amsterdam: Elsevier, 1993. P. 367.Kühne R. D. Phys. Bl. 1991. P. 201.

78. Sick, B. Dynamishe Effekte be in ichtlinearer Wellengleichungenmit Anwendungen in der Verkehrsfkub theorie: Master Thesis. University of Ulm, 1989.

79. Rödiger, M. Chaotische Lösungennichtlinearer Wellengleichungenmit Anwendungen in der Verkehrsfkub theorie: Master Thesis. University of Münster, 1990.

80. Musha, T., Higuchi, H. The 1/f fluctuation of traffic current on an

expressway // Journal of Applied Physics. 1978. № 15.

81. Аналіз сучасних програмних засобів транспортного моделювання М. М. Бекмагамбетов, д.т.н., проф. / О. В. Кочетков, д.т.н., проф. автомобільний журнал випуск №12 від 2022 року.

ДОДАТКИ

Додаток А(Технічне завдання)

Основними проблемами міської транспортної системи: зростання рівня автомобілізації населення, збільшення інтенсивності використання індивідуального транспорту, диспропорція між рівнем автомобілізації та темпами дорожнього будівництва, невідповідність пропускної спроможності вулично-дорожньої мережі реальному попиту на транспортні послуги тощо, з кожним днем усі актуальнішим стає вирішення завдань, що відповідають на такі питання: який ефект дасть та чи інша модернізація елемента вулично-дорожньої мережі; як зміна в організації дорожнього руху може вплинути на аварійність та пропускну здатність вузла або групи вузлів. Складання об'єктивних прогнозів транспортної ситуації залежно від зовнішніх та внутрішніх змін, аналіз та підготовка рекомендацій для інвестиційних проектів у галузі інфраструктури на результатах імітаційного моделювання параметрів транспортного потоку. Аналіз наукових досліджень та літературних джерел показав, що існує досить великий спектр дій щодо вирішення транспортних проблем. До них відносяться: удосконалення техніко експлуатаційних показників окремих елементів вулично-дорожньої мережі, систем управління дорожнім рухом та окремих систем транспорту тощо.

Однак ці дії не дозволяють ефективно використовувати дані моніторингу дорожнього руху, приймати обґрунтовані рішення щодо реалізації різних заходів щодо організації дорожнього руху, а також приймати науково обґрунтовані рішення на етапі проектування. Моделювання транспортних процесів дозволить значно підвищити ефективність діяльності з управління дорожнім рухом, а отже зменшити негативні аспекти автомобілізації.

У зв'язку з вище сказаним були поставлені наступні технічні завдання:

1. Розробити візуалізацію перехрестя.

2. Додати світлофор та таймер зміни світлофора на перехресті.

3. Розробити імітаційну систему руху автомобілів на перехресті.

4. Зробити можливим додавати автомобілі з різних сторін.

5. Підрахувати пропускну спроможність автомобілів з кожної сторони перехрестя.

Додаток Б(Form1.cs)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;


namespace Road
{
    public partial class Form1 : Form
    {
        int Vx1 = 0;
        int Vy1 = 360;
        int Vx2 = 700;
        int Vy2 = 310;
        int Vx3 = 310;
        int Vy3 = 0;
        int Vx4 = 360;
        int Vy4 = 700;
        int Vx5 = 100;
        int Vy5 = 360;


        int Vx6 = 0;
        int Vy6 = 360;
        int Vx7 = 0;
        int Vy7 = 360;
        int Vx8 = 0;
        int Vy8 = 360;


        int Vx9 = 700;
```

```
int Vy9 = 310;
int Vx10 = 700;
int Vy10 = 310;
int Vx11 = 700;
int Vy11 = 310;

int Vx12 = 310;
int Vy12 = 0;
int Vx13 = 310;
int Vy13 = 0;
int Vx14 = 310;
int Vy14 = 0;

int Vx15 = 360;
int Vy15 = 700;
int Vx16 = 360;
int Vy16 = 700;
int Vx17 = 360;
int Vy17 = 700;


int timer = 0;
Random rnd = new Random();

int carTop = 0;
int carLeft = 0;
int carRight = 0;
int carBottom = 0;


int r1;
int r2;
int r3;
int r4;
int r6;
int r7;
int r8;
int r9;
int r10;
int r11;
int r12;
int r13;
int r14;
int r15;
int r16;
int r17;

public Form1()
{
    InitializeComponent();
```

```
        }

        private void button1_Click(object sender, EventArgs e)
        {
            timer = 0;
            carTop = 0;
            carLeft = 0;
            carRight = 0;
            carBottom = 0;

            r1 = rnd.Next(1, 10);
            r2 = rnd.Next(1, 10);
            r3 = rnd.Next(1, 10);
            r4 = rnd.Next(1, 10);
            r6 = rnd.Next(1, 10);
            r7 = rnd.Next(1, 10);
            r8 = rnd.Next(1, 10);
            r9 = rnd.Next(1, 10);
            r10 = rnd.Next(1, 10);
            r11 = rnd.Next(1, 10);
            r12 = rnd.Next(1, 10);
            r13 = rnd.Next(1, 10);
            r14 = rnd.Next(1, 10);
            r15 = rnd.Next(1, 10);
            r16 = rnd.Next(1, 10);
            r17 = rnd.Next(1, 10);


            Graphics g = pictureBox1.CreateGraphics();
            g.Clear(Color.Turquoise);

            //road (top-left)
            g.DrawLine(new Pen(Brushes.Red, 4), new Point(0,
300), new Point(300, 300));
            g.DrawLine(new Pen(Brushes.Red, 4), new Point(300,
0), new Point(300, 300));

            //road (bottom-left)
            g.DrawLine(new Pen(Brushes.Red, 4), new Point(0,
400), new Point(300, 400));
            g.DrawLine(new Pen(Brushes.Red, 4), new Point(300,
700), new Point(300, 400));

            //road (top-right)
            g.DrawLine(new Pen(Brushes.Red, 4), new Point(400,
0), new Point(400, 300));
            g.DrawLine(new Pen(Brushes.Red, 4), new Point(700,
300), new Point(400, 300));

            //road (bottom-right)
```

```
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(700,
400), new Point(400, 400));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(400,
400), new Point(400, 700));

                //dash (top)
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(350,
0), new Point(350, 25));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(350,
50), new Point(350, 75));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(350,
100), new Point(350, 125));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(350,
150), new Point(350, 175));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(350,
200), new Point(350, 225));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(350,
250), new Point(350, 275));

                //dash (bottom)
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(350,
700), new Point(350, 675));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(350,
650), new Point(350, 625));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(350,
600), new Point(350, 575));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(350,
550), new Point(350, 525));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(350,
500), new Point(350, 475));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(350,
450), new Point(350, 425));

                //dash (left)
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(0,
350), new Point(25, 350));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(50,
350), new Point(75, 350));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(100,
350), new Point(125, 350));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(150,
350), new Point(175, 350));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(200,
350), new Point(225, 350));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(250,
350), new Point(275, 350));

                //dash (right)
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(675,
350), new Point(700, 350));
```

```
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(625,
350), new Point(650, 350));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(575,
350), new Point(600, 350));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(525,
350), new Point(550, 350));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(475,
350), new Point(500, 350));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(425,
350), new Point(450, 350));

                //light (top-left)
                g.FillEllipse(new   SolidBrush(Color.Green),   275,
275, 20, 20);

                //light (bottom-right)
                g.FillEllipse(new   SolidBrush(Color.Green),   405,
405, 20, 20);

                //light (top-right)
                g.FillEllipse(new SolidBrush(Color.Red), 405, 275,
20, 20);

                //light (bottom-left)
                g.FillEllipse(new SolidBrush(Color.Red), 275, 405,
20, 20);



                Vx1 = 0;
                Vx2 = 0;
                Vy3 = 0;
                Vy4 = 0;
                Vx5 = 100;
                Vy5 = 360;
                Vx6 = 0;


                timer1.Enabled = true;
            }

            private void timer1_Tick(object sender, EventArgs e)
            {
                timer++;
                int time=0;
                if (timer > 500)
                    timer = 0;
                if (timer < 10)
                    time = 10 - timer;
                if (10<timer && timer<=240)
                    time = 240 - timer;
```

```
                if (240 < timer && timer<=260 )
                    time = 260 - timer;
                if (260 < timer && timer<=490)
                    time = 490 - timer;
                if (490 < timer && timer<=500)
                    time = 510 - timer;
                if (time%60>=10)
                    label1.Text = "0" + Convert.ToString(time / 60)
+ ":" + Convert.ToString(time % 60);
                else
                    label1.Text = "0" + Convert.ToString(time / 60)
+ ":0" + Convert.ToString(time % 60);


                Graphics g = pictureBox1.CreateGraphics();
                g.Clear(Color.Turquoise);

                //road (top-left)
                g.DrawLine(new  Pen(Brushes.Red,  4),  new  Point(0,
300), new Point(300, 300));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(300,
0), new Point(300, 300));

                //road (bottom-left)
                g.DrawLine(new  Pen(Brushes.Red,  4),  new  Point(0,
400), new Point(300, 400));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(300,
700), new Point(300, 400));

                //road (top-right)
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(400,
0), new Point(400, 300));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(700,
300), new Point(400, 300));

                //road (bottom-right)
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(700,
400), new Point(400, 400));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(400,
400), new Point(400, 700));

                //dash (top)
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(350,
0), new Point(350, 25));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(350,
50), new Point(350, 75));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(350,
100), new Point(350, 125));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(350,
150), new Point(350, 175));
```

```
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(350,
200), new Point(350, 225));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(350,
250), new Point(350, 275));

                //dash (bottom)
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(350,
700), new Point(350, 675));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(350,
650), new Point(350, 625));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(350,
600), new Point(350, 575));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(350,
550), new Point(350, 525));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(350,
500), new Point(350, 475));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(350,
450), new Point(350, 425));

                //dash (left)
                g.DrawLine(new  Pen(Brushes.Red,  4),  new  Point(0,
350), new Point(25, 350));
                g.DrawLine(new  Pen(Brushes.Red,  4),  new  Point(50,
350), new Point(75, 350));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(100,
350), new Point(125, 350));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(150,
350), new Point(175, 350));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(200,
350), new Point(225, 350));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(250,
350), new Point(275, 350));

                //dash (right)
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(675,
350), new Point(700, 350));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(625,
350), new Point(650, 350));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(575,
350), new Point(600, 350));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(525,
350), new Point(550, 350));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(475,
350), new Point(500, 350));
                g.DrawLine(new Pen(Brushes.Red, 4), new Point(425,
350), new Point(450, 350));

                if (10<timer && timer <= 240 )
                {
                    //light (top-left)
```

```
                    g.FillEllipse(new SolidBrush(Color.Green), 275,
275, 20, 20);

                    //light (bottom-right)
                    g.FillEllipse(new SolidBrush(Color.Green), 405,
405, 20, 20);

                    //light (top-right)
                    g.FillEllipse(new  SolidBrush(Color.Red),  405,
275, 20, 20);

                    //light (bottom-left)
                    g.FillEllipse(new  SolidBrush(Color.Red),  275,
405, 20, 20);
                }
                else if ((240<timer && timer<260) || timer<=10 ||
timer>=490)
                {
                    //light (top-left)
                    g.FillEllipse(new     SolidBrush(Color.Orange),
275, 275, 20, 20);

                    //light (bottom-right)
                    g.FillEllipse(new     SolidBrush(Color.Orange),
405, 405, 20, 20);

                    //light (top-right)
                    g.FillEllipse(new     SolidBrush(Color.Orange),
405, 275, 20, 20);

                    //light (bottom-left)
                    g.FillEllipse(new     SolidBrush(Color.Orange),
275, 405, 20, 20);
                }
                else
                {
                    //light (top-left)
                    g.FillEllipse(new  SolidBrush(Color.Red),  275,
275, 20, 20);

                    //light (bottom-right)
                    g.FillEllipse(new  SolidBrush(Color.Red),  405,
405, 20, 20);

                    //light (top-right)
                    g.FillEllipse(new SolidBrush(Color.Green), 405,
275, 20, 20);

                    //light (bottom-left)
                    g.FillEllipse(new SolidBrush(Color.Green), 275,
405, 20, 20);
```

```
                }


                if (radioButton1.Checked==true)
                {
                    button2.Visible = false;
                    button3.Visible = false;
                    button4.Visible = false;
                    button5.Visible = false;
                }

                if (radioButton2.Checked==true)
                {
                    button2.Visible = true;
                    button3.Visible = false;
                    button4.Visible = false;
                    button5.Visible = false;

                    Brush c1 = new SolidBrush(Color.DeepPink);
                    if (Vx1 >= 240 && Vx1 <= 250 && (timer<260 ||
timer>=490))
                        Vx1 += 0;
                    else if (Vx1 > Vx6 - 70 && Vx1 < Vx6 && Vy6 ==
Vy1)
                        Vx1 += 0;
                    else if (Vx1 > Vx7 - 70 && Vx1 < Vx7 && Vy7 ==
Vy1)
                        Vx1 += 0;
                    else if (Vx1 > Vx8 - 70 && Vx1 < Vx8 && Vy8 ==
Vy1)
                        Vx1 += 0;
                    else
                        Vx1 += r1;

                    if (Vx1 >= 700)
                        Vx1 -= 700;
                    g.FillRectangle(c1, Vx1, Vy1, 50, 25);

                    if (carLeft >= 1)
                    {
                        Brush c6 = new SolidBrush(Color.Pink);

                        if (Vx6 >= 240 && Vx6 <= 250 && (timer < 260
|| timer >= 490))
                            Vx6 += 0;
                        else if (Vx6 > Vx1 - 70 && Vx6 < Vx1 && Vy6
== Vy1)
                            Vx6 += 0;
                        else if (Vx6 > Vx7 - 70 && Vx6 < Vx7 && Vy6
== Vy7)
                            Vx6 += 0;
```

```
                        else if (Vx6 > Vx8 - 70 && Vx6 < Vx8 && Vy6
== Vy8)
                            Vx6 += 0;
                        else
                            Vx6 += r6;

                        if (Vx6 >= 700)
                            Vx6 -= 700;
                        g.FillRectangle(c6, Vx6, Vy6, 50, 25);

                        if (carLeft >= 2)
                        {
                            Brush          c7          =          new
SolidBrush(Color.Orange);

                            if (Vx7 >= 240 && Vx7 <= 250 && (timer
< 260 || timer >= 490))
                                Vx7 += 0;
                            else if (Vx7 > Vx1 - 70 && Vx7 < Vx1 &&
Vy7 == Vy1)
                                Vx7 += 0;
                            else if (Vx7 > Vx6 - 70 && Vx7 < Vx6 &&
Vy7 == Vy6)
                                Vx7 += 0;
                            else if (Vx7 > Vx8 - 70 && Vx7 < Vx8 &&
Vy7 == Vy8)
                                Vx7 += 0;
                            else
                                Vx7 += r7;

                            if (Vx7 >= 700)
                                Vx7 -= 700;
                            g.FillRectangle(c7, Vx7, Vy7, 50, 25);

                            if (carLeft >= 3)
                            {
                                Brush        c8        =        new
SolidBrush(Color.Coral);

                                if (Vx8 >= 240 && Vx8 <= 250 &&
(timer < 260 || timer >= 490))
                                    Vx8 += 0;
                                else if (Vx8 > Vx1 - 70 && Vx8 < Vx1
&& Vy8 == Vy1)
                                    Vx8 += 0;
                                else if (Vx8 > Vx6 - 70 && Vx8 < Vx6
&& Vy8 == Vy6)
                                    Vx8 += 0;
                                else if (Vx8 > Vx7 - 70 && Vx8 < Vx7
&& Vy8 == Vy7)
                                    Vx8 += 0;
```

```
                                else
                                    Vx8 += r8;

                                if (Vx8 >= 700)
                                    Vx8 -= 700;
                                g.FillRectangle(c8, Vx8, Vy8, 50,
25);
                            }
                        }
                    }
                }

                if (radioButton3.Checked == true)
                {
                    button2.Visible = true;
                    button3.Visible = false;
                    button4.Visible = true;
                    button5.Visible = false;
                    Brush c1 = new SolidBrush(Color.DeepPink);
                    Brush c2 = new SolidBrush(Color.Gray);
                    if (Vx1 >= 240 && Vx1 <= 250 && (timer < 260 ||
timer >= 490))
                        Vx1 += 0;
                    else if (Vx1 > Vx6 - 70 && Vx1 < Vx6 && Vy6 ==
Vy1)
                        Vx1 += 0;
                    else if (Vx1 > Vx7 - 70 && Vx1 < Vx7 && Vy7 ==
Vy1)
                        Vx1 += 0;
                    else if (Vx1 > Vx8 - 70 && Vx1 < Vx8 && Vy8 ==
Vy1)
                        Vx1 += 0;
                    else
                        Vx1 += r1;

                    if (Vx1 >= 700)
                        Vx1 -= 700;
                    g.FillRectangle(c1, Vx1, Vy1, 50, 25);

                    if (carLeft >= 1)
                    {
                        Brush c6 = new SolidBrush(Color.Pink);

                        if (Vx6 >= 240 && Vx6 <= 250 && (timer < 260
|| timer >= 490))
                            Vx6 += 0;
                        else if (Vx6 > Vx1 - 70 && Vx6 < Vx1 && Vy6
== Vy1)
                            Vx6 += 0;
                        else if (Vx6 > Vx7 - 70 && Vx6 < Vx7 && Vy6
== Vy7)
```

```
                    Vx6 += 0;
                else if (Vx6 > Vx8 - 70 && Vx6 < Vx8 && Vy6
== Vy8)
                    Vx6 += 0;
                else
                    Vx6 += r6;

                if (Vx6 >= 700)
                    Vx6 -= 700;
                g.FillRectangle(c6, Vx6, Vy6, 50, 25);

                if (carLeft >= 2)
                {
                    Brush          c7          =          new
SolidBrush(Color.Orange);

                    if (Vx7 >= 240 && Vx7 <= 250 && (timer
< 260 || timer >= 490))
                        Vx7 += 0;
                    else if (Vx7 > Vx1 - 70 && Vx7 < Vx1 &&
Vy7 == Vy1)
                        Vx7 += 0;
                    else if (Vx7 > Vx6 - 70 && Vx7 < Vx6 &&
Vy7 == Vy6)
                        Vx7 += 0;
                    else if (Vx7 > Vx8 - 70 && Vx7 < Vx8 &&
Vy7 == Vy8)
                        Vx7 += 0;
                    else
                        Vx7 += r7;

                    if (Vx7 >= 700)
                        Vx7 -= 700;
                    g.FillRectangle(c7, Vx7, Vy7, 50, 25);

                    if (carLeft >= 3)
                    {
                        Brush        c8        =        new
SolidBrush(Color.Coral);

                        if (Vx8 >= 240 && Vx8 <= 250 &&
(timer < 260 || timer >= 490))
                            Vx8 += 0;
                        else if (Vx8 > Vx1 - 70 && Vx8 < Vx1
&& Vy8 == Vy1)
                            Vx8 += 0;
                        else if (Vx8 > Vx6 - 70 && Vx8 < Vx6
&& Vy8 == Vy6)
                            Vx8 += 0;
                        else if (Vx8 > Vx7 - 70 && Vx8 < Vx7
&& Vy8 == Vy7)
```

```
                                    Vx8 += 0;
                                else
                                    Vx8 += r8;

                                if (Vx8 >= 700)
                                    Vx8 -= 700;
                                g.FillRectangle(c8, Vx8, Vy8, 50,
25);
                            }
                        }
                    }



                    if (Vx2 >= 400 && Vx2 <= 410 && (timer < 260 ||
timer >= 490))
                        Vx2 += 0;
                    else if(Vx2 < Vx9 + 70 && Vx2 > Vx9 && Vy2 ==
Vy9)
                        Vx2 += 0;
                    else if (Vx2 < Vx10 + 70 && Vx2 > Vx10 && Vy2 ==
Vy10)
                        Vx2 += 0;
                    else if (Vx2 < Vx11 + 70 && Vx2 > Vx11 && Vy2 ==
Vy11)
                        Vx2 += 0;
                    else
                        Vx2 -= r2;

                    if (Vx2 <= 0)
                        Vx2 += 700;
                    g.FillRectangle(c2, Vx2, Vy2, 50, 25);

                    if (carRight >= 1)
                    {
                        Brush c9 = new SolidBrush(Color.Tomato);

                        if (Vx9 >= 400 && Vx9 <= 410 && (timer < 260
|| timer >= 490))
                            Vx9 += 0;
                        else if (Vx9 < Vx2 + 70 && Vx9 > Vx2 && Vy9
== Vy2)
                            Vx9 += 0;
                        else if (Vx9 < Vx10 + 70 && Vx9 > Vx10 &&
Vy9 == Vy10)
                            Vx9 += 0;
                        else if (Vx9 < Vx11 + 70 && Vx9 > Vx11 &&
Vy9 == Vy11)
                            Vx9 += 0;
                        else
                            Vx9 -= r9;
```

```
                        if (Vx9 <= 0)
                            Vx9 += 700;
                        g.FillRectangle(c9, Vx9, Vy9, 50, 25);

                        if (carRight >= 2)
                        {
                            Brush         c10         =         new
SolidBrush(Color.Orange);

                            if (Vx10 >= 400 && Vx10 <= 410 && (timer
< 260 || timer >= 490))
                                Vx10 += 0;
                            else if (Vx10 < Vx2 + 70 && Vx10 > Vx2
&& Vy10 == Vy2)
                                Vx10 += 0;
                            else if (Vx10 < Vx9 + 70 && Vx10 > Vx9
&& Vy10 == Vy9)
                                Vx10 += 0;
                            else if (Vx10 < Vx11 + 70 && Vx10 > Vx11
&& Vy10 == Vy11)
                                Vx10 += 0;
                            else
                                Vx10 -= r10;

                            if (Vx10 <= 0)
                                Vx10 += 700;
                            g.FillRectangle(c10,  Vx10,  Vy10,  50,
25);

                            if (carRight >= 3)
                            {
                                Brush        c11        =        new
SolidBrush(Color.Coral);

                                if (Vx11 >= 400 && Vx11 <= 410 &&
(timer < 260 || timer >= 490))
                                    Vx11 += 0;
                                else if (Vx11 < Vx2 + 70 && Vx11 >
Vx2 && Vy11 == Vy2)
                                    Vx11 += 0;
                                else if (Vx11 < Vx9 + 70 && Vx11 >
Vx9 && Vy11 == Vy9)
                                    Vx11 += 0;
                                else if (Vx11 < Vx10 + 70 && Vx11 >
Vx10 && Vy11 == Vy10)
                                    Vx11 += 0;
                                else
                                    Vx11 -= r11;

                                if (Vx11 <= 0)
```

```
                                    Vx11 += 700;
                                g.FillRectangle(c11,  Vx11,  Vy11,
50, 25);
                            }
                        }
                    }
                }


                if (radioButton4.Checked == true)
                {
                    button2.Visible = true;
                    button3.Visible = true;
                    button4.Visible = true;
                    button5.Visible = false;

                    Brush c1 = new SolidBrush(Color.DeepPink);
                    Brush c2 = new SolidBrush(Color.Gray);
                    Brush c3 = new SolidBrush(Color.Black);
                    if (Vx1 >= 240 && Vx1 <= 250 && (timer < 260 ||
timer >= 490))
                        Vx1 += 0;
                    else if (Vx1 > Vx6 - 70 && Vx1 < Vx6 && Vy6 ==
Vy1)
                        Vx1 += 0;
                    else if (Vx1 > Vx7 - 70 && Vx1 < Vx7 && Vy7 ==
Vy1)
                        Vx1 += 0;
                    else if (Vx1 > Vx8 - 70 && Vx1 < Vx8 && Vy8 ==
Vy1)
                        Vx1 += 0;
                    else
                        Vx1 += r1;

                    if (Vx1 >= 700)
                        Vx1 -= 700;
                    g.FillRectangle(c1, Vx1, Vy1, 50, 25);

                    if (carLeft >= 1)
                    {
                        Brush c6 = new SolidBrush(Color.Pink);

                        if (Vx6 >= 240 && Vx6 <= 250 && (timer < 260
|| timer >= 490))
                            Vx6 += 0;
                        else if (Vx6 > Vx1 - 70 && Vx6 < Vx1 && Vy6
== Vy1)
                            Vx6 += 0;
                        else if (Vx6 > Vx7 - 70 && Vx6 < Vx7 && Vy6
== Vy7)
                            Vx6 += 0;
```

```
                        else if (Vx6 > Vx8 - 70 && Vx6 < Vx8 && Vy6
== Vy8)
                            Vx6 += 0;
                        else
                            Vx6 += r6;

                        if (Vx6 >= 700)
                            Vx6 -= 700;
                        g.FillRectangle(c6, Vx6, Vy6, 50, 25);

                        if (carLeft >= 2)
                        {
                            Brush          c7          =          new
SolidBrush(Color.Orange);

                            if (Vx7 >= 240 && Vx7 <= 250 && (timer
< 260 || timer >= 490))
                                Vx7 += 0;
                            else if (Vx7 > Vx1 - 70 && Vx7 < Vx1 &&
Vy7 == Vy1)
                                Vx7 += 0;
                            else if (Vx7 > Vx6 - 70 && Vx7 < Vx6 &&
Vy7 == Vy6)
                                Vx7 += 0;
                            else if (Vx7 > Vx8 - 70 && Vx7 < Vx8 &&
Vy7 == Vy8)
                                Vx7 += 0;
                            else
                                Vx7 += r7;

                            if (Vx7 >= 700)
                                Vx7 -= 700;
                            g.FillRectangle(c7, Vx7, Vy7, 50, 25);

                            if (carLeft >= 3)
                            {
                                Brush        c8        =        new
SolidBrush(Color.Coral);

                                if (Vx8 >= 240 && Vx8 <= 250 &&
(timer < 260 || timer >= 490))
                                    Vx8 += 0;
                                else if (Vx8 > Vx1 - 70 && Vx8 < Vx1
&& Vy8 == Vy1)
                                    Vx8 += 0;
                                else if (Vx8 > Vx6 - 70 && Vx8 < Vx6
&& Vy8 == Vy6)
                                    Vx8 += 0;
                                else if (Vx8 > Vx7 - 70 && Vx8 < Vx7
&& Vy8 == Vy7)
                                    Vx8 += 0;
```

```
                                else
                                    Vx8 += r8;

                                if (Vx8 >= 700)
                                    Vx8 -= 700;
                                g.FillRectangle(c8, Vx8, Vy8, 50,
25);
                            }
                        }
                    }



                    if (Vx2 >= 400 && Vx2 <= 410 && (timer < 260 ||
timer >= 490))
                        Vx2 += 0;
                    else if (Vx2 < Vx9 + 70 && Vx2 > Vx9 && Vy2 ==
Vy9)
                        Vx2 += 0;
                    else if (Vx2 < Vx10 + 70 && Vx2 > Vx10 && Vy2 ==
Vy10)
                        Vx2 += 0;
                    else if (Vx2 < Vx11 + 70 && Vx2 > Vx11 && Vy2 ==
Vy11)
                        Vx2 += 0;
                    else
                        Vx2 -= r2;

                    if (Vx2 <= 0)
                        Vx2 += 700;
                    g.FillRectangle(c2, Vx2, Vy2, 50, 25);

                    if (carRight >= 1)
                    {
                        Brush c9 = new SolidBrush(Color.Tomato);

                        if (Vx9 >= 400 && Vx9 <= 410 && (timer < 260
|| timer >= 490))
                            Vx9 += 0;
                        else if (Vx9 < Vx2 + 70 && Vx9 > Vx2 && Vy9
== Vy2)
                            Vx9 += 0;
                        else if (Vx9 < Vx10 + 70 && Vx9 > Vx10 &&
Vy9 == Vy10)
                            Vx9 += 0;
                        else if (Vx9 < Vx11 + 70 && Vx9 > Vx11 &&
Vy9 == Vy11)
                            Vx9 += 0;
                        else
                            Vx9 -= r9;
```

```
                        if (Vx9 <= 0)
                            Vx9 += 700;
                        g.FillRectangle(c9, Vx9, Vy9, 50, 25);

                        if (carRight >= 2)
                        {
                            Brush         c10         =         new
SolidBrush(Color.Orange);

                            if (Vx10 >= 400 && Vx10 <= 410 && (timer
< 260 || timer >= 490))
                                Vx10 += 0;
                            else if (Vx10 < Vx2 + 70 && Vx10 > Vx2
&& Vy10 == Vy2)
                                Vx10 += 0;
                            else if (Vx10 < Vx9 + 70 && Vx10 > Vx9
&& Vy10 == Vy9)
                                Vx10 += 0;
                            else if (Vx10 < Vx11 + 70 && Vx10 > Vx11
&& Vy10 == Vy11)
                                Vx10 += 0;
                            else
                                Vx10 -= r10;

                            if (Vx10 <= 0)
                                Vx10 += 700;
                            g.FillRectangle(c10,  Vx10,  Vy10,  50,
25);

                            if (carRight >= 3)
                            {
                                Brush        c11        =        new
SolidBrush(Color.Coral);

                                if (Vx11 >= 400 && Vx11 <= 410 &&
(timer < 260 || timer >= 490))
                                    Vx11 += 0;
                                else if (Vx11 < Vx2 + 70 && Vx11 >
Vx2 && Vy11 == Vy2)
                                    Vx11 += 0;
                                else if (Vx11 < Vx9 + 70 && Vx11 >
Vx9 && Vy11 == Vy9)
                                    Vx11 += 0;
                                else if (Vx11 < Vx10 + 70 && Vx11 >
Vx10 && Vy11 == Vy10)
                                    Vx11 += 0;
                                else
                                    Vx11 -= r11;

                                if (Vx11 <= 0)
```

```
                                    Vx11 += 700;
                                g.FillRectangle(c11,  Vx11,  Vy11,
50, 25);
                            }
                        }
                    }



                    if (Vy3 >= 240 && Vy3 <= 250 && (timer < 10 ||
timer >= 240))
                        Vy3 += 0;
                    else if (Vy3 > Vy12 - 70 && Vy3 < Vy12 && Vx3 ==
Vx12)
                        Vy3 += 0;
                    else if (Vy3 > Vy13 - 70 && Vy3 < Vy13 && Vx3 ==
Vx13)
                        Vy3 += 0;
                    else if (Vy3 > Vy14 - 70 && Vy3 < Vy14 && Vx3 ==
Vx14)
                        Vy3 += 0;
                    else
                        Vy3 += r3;

                    if (Vy3 >= 700)
                        Vy3 -= 700;

                    g.FillRectangle(c3, Vx3, Vy3, 25, 50);

                    if (carTop >= 1)
                    {
                        Brush c12 = new SolidBrush(Color.Pink);

                        if (Vy12 >= 240 && Vy12 <= 250 && (timer <
10 || timer >= 240))
                            Vy12 += 0;
                        else if (Vy12 > Vy3 - 70 && Vy12 < Vy3 &&
Vx12 == Vx3)
                            Vy12 += 0;
                        else if (Vy12 > Vy13 - 70 && Vy12 < Vy13 &&
Vx12 == Vx13)
                            Vy12 += 0;
                        else if (Vy12 > Vy14 - 70 && Vy12 < Vy14 &&
Vx12 == Vx14)
                            Vy12 += 0;
                        else
                            Vy12 += r12;

                        if (Vy12 >= 700)
                            Vy12 -= 700;
                        g.FillRectangle(c12, Vx12, Vy12, 25, 50);
```

```
                        if (carTop >= 2)
                        {
                            Brush         c13         =         new
SolidBrush(Color.Orange);

                            if (Vy13 >= 240 && Vy13 <= 250 && (timer
< 10 || timer >= 240))
                                Vy13 += 0;
                            else if (Vy13 > Vy3 - 70 && Vy13 < Vy3
&& Vx13 == Vx3)
                                Vy13 += 0;
                            else if (Vy13 > Vy12 - 70 && Vy13 < Vy12
&& Vx13 == Vx12)
                                Vy13 += 0;
                            else if (Vy13 > Vy14 - 70 && Vy13 < Vy14
&& Vy13 == Vy14)
                                Vy13 += 0;
                            else
                                Vy13 += r13;

                            if (Vy13 >= 700)
                                Vy13 -= 700;
                            g.FillRectangle(c13,  Vx13,  Vy13,  25,
50);

                            if (carTop >= 3)
                            {
                                Brush        c14        =        new
SolidBrush(Color.Coral);

                                if (Vy14 >= 240 && Vy14 <= 250 &&
(timer < 10 || timer >= 240))
                                    Vy14 += 0;
                                else if (Vy14 > Vy3 - 70 && Vy14 <
Vy3 && Vx14 == Vx3)
                                    Vy14 += 0;
                                else if (Vy14 > Vy12 - 70 && Vy14 <
Vy12 && Vx14 == Vx12)
                                    Vy14 += 0;
                                else if (Vy14 > Vy13 - 70 && Vy14 <
Vy13 && Vx14 == Vx13)
                                    Vy14 += 0;
                                else
                                    Vy14 += r14;

                                if (Vy14 >= 700)
                                    Vy14 -= 700;
                                g.FillRectangle(c14,  Vx14,  Vy14,
25, 50);
                            }
```

```
                    }
                }

            }


            if (radioButton5.Checked == true)
            {
                button2.Visible = true;
                button3.Visible = true;
                button4.Visible = true;
                button5.Visible = true;

                Brush c1 = new SolidBrush(Color.DeepPink);
                Brush c2 = new SolidBrush(Color.Gray);
                Brush c3 = new SolidBrush(Color.Black);
                Brush c4 = new SolidBrush(Color.Yellow);
                if (Vx1 >= 240 && Vx1 <= 250 && (timer < 260 ||
timer >= 490))
                    Vx1 += 0;
                else if (Vx1 > Vx6 - 70 && Vx1 < Vx6 && Vy6 ==
Vy1)
                    Vx1 += 0;
                else if (Vx1 > Vx7 - 70 && Vx1 < Vx7 && Vy7 ==
Vy1)
                    Vx1 += 0;
                else if (Vx1 > Vx8 - 70 && Vx1 < Vx8 && Vy8 ==
Vy1)
                    Vx1 += 0;
                else
                    Vx1 += r1;

                if (Vx1 >= 700)
                    Vx1 -= 700;
                g.FillRectangle(c1, Vx1, Vy1, 50, 25);

                if (carLeft >= 1)
                {
                    Brush c6 = new SolidBrush(Color.Pink);

                    if (Vx6 >= 240 && Vx6 <= 250 && (timer < 260
|| timer >= 490))
                        Vx6 += 0;
                    else if (Vx6 > Vx1 - 70 && Vx6 < Vx1 && Vy6
== Vy1)
                        Vx6 += 0;
                    else if (Vx6 > Vx7 - 70 && Vx6 < Vx7 && Vy6
== Vy7)
                        Vx6 += 0;
                    else if (Vx6 > Vx8 - 70 && Vx6 < Vx8 && Vy6
== Vy8)
```

```
                    Vx6 += 0;
                else
                    Vx6 += r6;

                if (Vx6 >= 700)
                    Vx6 -= 700;
                g.FillRectangle(c6, Vx6, Vy6, 50, 25);

                if (carLeft >= 2)
                {
                    Brush          c7          =          new
SolidBrush(Color.Orange);

                    if (Vx7 >= 240 && Vx7 <= 250 && (timer
< 260 || timer >= 490))
                        Vx7 += 0;
                    else if (Vx7 > Vx1 - 70 && Vx7 < Vx1 &&
Vy7 == Vy1)
                        Vx7 += 0;
                    else if (Vx7 > Vx6 - 70 && Vx7 < Vx6 &&
Vy7 == Vy6)
                        Vx7 += 0;
                    else if (Vx7 > Vx8 - 70 && Vx7 < Vx8 &&
Vy7 == Vy8)
                        Vx7 += 0;
                    else
                        Vx7 += r7;

                    if (Vx7 >= 700)
                        Vx7 -= 700;
                    g.FillRectangle(c7, Vx7, Vy7, 50, 25);

                    if (carLeft >= 3)
                    {
                        Brush        c8        =        new
SolidBrush(Color.Coral);

                        if (Vx8 >= 240 && Vx8 <= 250 &&
(timer < 260 || timer >= 490))
                            Vx8 += 0;
                        else if (Vx8 > Vx1 - 70 && Vx8 < Vx1
&& Vy8 == Vy1)
                            Vx8 += 0;
                        else if (Vx8 > Vx6 - 70 && Vx8 < Vx6
&& Vy8 == Vy6)
                            Vx8 += 0;
                        else if (Vx8 > Vx7 - 70 && Vx8 < Vx7
&& Vy8 == Vy7)
                            Vx8 += 0;
                        else
                            Vx8 += r8;
```

```
                                if (Vx8 >= 700)
                                    Vx8 -= 700;
                                g.FillRectangle(c8, Vx8, Vy8, 50,
25);
                            }
                        }
                    }



                    if (Vx2 >= 400 && Vx2 <= 410 && (timer < 260 ||
timer >= 490))
                        Vx2 += 0;
                    else if (Vx2 < Vx9 + 70 && Vx2 > Vx9 && Vy2 ==
Vy9)
                        Vx2 += 0;
                    else if (Vx2 < Vx10 + 70 && Vx2 > Vx10 && Vy2 ==
Vy10)
                        Vx2 += 0;
                    else if (Vx2 < Vx11 + 70 && Vx2 > Vx11 && Vy2 ==
Vy11)
                        Vx2 += 0;
                    else
                        Vx2 -= r2;

                    if (Vx2 <= 0)
                        Vx2 += 700;
                    g.FillRectangle(c2, Vx2, Vy2, 50, 25);

                    if (carRight >= 1)
                    {
                        Brush c9 = new SolidBrush(Color.Tomato);

                        if (Vx9 >= 400 && Vx9 <= 410 && (timer < 260
|| timer >= 490))
                            Vx9 += 0;
                        else if (Vx9 < Vx2 + 70 && Vx9 > Vx2 && Vy9
== Vy2)
                            Vx9 += 0;
                        else if (Vx9 < Vx10 + 70 && Vx9 > Vx10 &&
Vy9 == Vy10)
                            Vx9 += 0;
                        else if (Vx9 < Vx11 + 70 && Vx9 > Vx11 &&
Vy9 == Vy11)
                            Vx9 += 0;
                        else
                            Vx9 -= r9;

                        if (Vx9 <= 0)
                            Vx9 += 700;
```

```
                        g.FillRectangle(c9, Vx9, Vy9, 50, 25);

                        if (carRight >= 2)
                        {
                            Brush         c10         =         new
SolidBrush(Color.Orange);

                            if (Vx10 >= 400 && Vx10 <= 410 && (timer
< 260 || timer >= 490))
                                Vx10 += 0;
                            else if (Vx10 < Vx2 + 70 && Vx10 > Vx2
&& Vy10 == Vy2)
                                Vx10 += 0;
                            else if (Vx10 < Vx9 + 70 && Vx10 > Vx9
&& Vy10 == Vy9)
                                Vx10 += 0;
                            else if (Vx10 < Vx11 + 70 && Vx10 > Vx11
&& Vy10 == Vy11)
                                Vx10 += 0;
                            else
                                Vx10 -= r10;

                            if (Vx10 <= 0)
                                Vx10 += 700;
                            g.FillRectangle(c10,  Vx10,  Vy10,  50,
25);

                            if (carRight >= 3)
                            {
                                Brush        c11        =        new
SolidBrush(Color.Coral);

                                if (Vx11 >= 400 && Vx11 <= 410 &&
(timer < 260 || timer >= 490))
                                    Vx11 += 0;
                                else if (Vx11 < Vx2 + 70 && Vx11 >
Vx2 && Vy11 == Vy2)
                                    Vx11 += 0;
                                else if (Vx11 < Vx9 + 70 && Vx11 >
Vx9 && Vy11 == Vy9)
                                    Vx11 += 0;
                                else if (Vx11 < Vx10 + 70 && Vx11 >
Vx10 && Vy11 == Vy10)
                                    Vx11 += 0;
                                else
                                    Vx11 -= r11;

                                if (Vx11 <= 0)
                                    Vx11 += 700;
                                g.FillRectangle(c11,   Vx11,   Vy11,
50, 25);
```

```
                        }
                    }
                }

                if (Vy3 >= 240 && Vy3 <= 250 && (timer < 10 ||
timer >= 240))
                    Vy3 += 0;
                else if (Vy3 > Vy12 - 70 && Vy3 < Vy12 && Vx3 ==
Vx12)
                    Vy3 += 0;
                else if (Vy3 > Vy13 - 70 && Vy3 < Vy13 && Vx3 ==
Vx13)
                    Vy3 += 0;
                else if (Vy3 > Vy14 - 70 && Vy3 < Vy14 && Vx3 ==
Vx14)
                    Vy3 += 0;
                else
                    Vy3 += r3;

                if (Vy3 >= 700)
                    Vy3 -= 700;

                g.FillRectangle(c3, Vx3, Vy3, 25, 50);

                if (carTop >= 1)
                {
                    Brush c12 = new SolidBrush(Color.Pink);

                    if (Vy12 >= 240 && Vy12 <= 250 && (timer <
10 || timer >= 240))
                        Vy12 += 0;
                    else if (Vy12 > Vy3 - 70 && Vy12 < Vy3 &&
Vx12 == Vx3)
                        Vy12 += 0;
                    else if (Vy12 > Vy13 - 70 && Vy12 < Vy13 &&
Vx12 == Vx13)
                        Vy12 += 0;
                    else if (Vy12 > Vy14 - 70 && Vy12 < Vy14 &&
Vx12 == Vx14)
                        Vy12 += 0;
                    else
                        Vy12 += r12;

                    if (Vy12 >= 700)
                        Vy12 -= 700;
                    g.FillRectangle(c12, Vx12, Vy12, 25, 50);

                    if (carTop >= 2)
                    {
                        Brush         c13         =         new
SolidBrush(Color.Orange);
```

```
                        if (Vy13 >= 240 && Vy13 <= 250 && (timer
< 10 || timer >= 240))
                            Vy13 += 0;
                        else if (Vy13 > Vy3 - 70 && Vy13 < Vy3
&& Vx13 == Vx3)
                            Vy13 += 0;
                        else if (Vy13 > Vy12 - 70 && Vy13 < Vy12
&& Vx13 == Vx12)
                            Vy13 += 0;
                        else if (Vy13 > Vy14 - 70 && Vy13 < Vy14
&& Vx13 == Vx14)
                            Vy13 += 0;
                        else
                            Vy13 += r13;

                        if (Vy13 >= 700)
                            Vy13 -= 700;
                        g.FillRectangle(c13,  Vx13,  Vy13,  25,
50);

                        if (carTop >= 3)
                        {
                            Brush        c14        =        new
SolidBrush(Color.Coral);

                            if (Vy14 >= 240 && Vy14 <= 250 &&
(timer < 10 || timer >= 240))
                                Vy14 += 0;
                            else if (Vy14 > Vy3 - 70 && Vy14 <
Vy3 && Vx14 == Vx3)
                                Vy14 += 0;
                            else if (Vy14 > Vy12 - 70 && Vy14 <
Vy12 && Vx14 == Vx12)
                                Vy14 += 0;
                            else if (Vy14 > Vy13 - 70 && Vy14 <
Vy13 && Vx14 == Vx13)
                                Vy14 += 0;
                            else
                                Vy14 += r14;

                            if (Vy14 >= 700)
                                Vy14 -= 700;
                            g.FillRectangle(c14,  Vx14,  Vy14,
25, 50);
                        }
                    }
                }
```

```
                    if (Vy4 >= 400 && Vy4 <= 410 && (timer < 10 ||
timer >= 240))
                        Vy4 += 0;
                    else if (Vy4 < Vy15 + 70 && Vy4 > Vy15 && Vx4 ==
Vx15)
                        Vy4 += 0;
                    else if (Vy4 < Vy16 + 70 && Vy4 > Vy16 && Vx4 ==
Vx16)
                        Vy4 += 0;
                    else if (Vy4 < Vy17 + 70 && Vy4 > Vy17 && Vx4 ==
Vx17)
                        Vy4 += 0;
                    else
                        Vy4 -= r4;

                    if (Vy4 <= 0)
                        Vy4 += 700;

                    g.FillRectangle(c4, Vx4, Vy4, 25, 50);

                    if (carBottom >= 1)
                    {
                        Brush c15 = new SolidBrush(Color.Pink);

                        if (Vy15 >= 400 && Vy15 <= 410 && (timer <
10 || timer >= 240))
                            Vy15 += 0;
                        else if (Vy15 < Vy4 + 70 && Vy15 > Vy4 &&
Vx15 == Vx4)
                            Vy15 += 0;
                        else if (Vy15 < Vy16 + 70 && Vy15 > Vy16 &&
Vx15 == Vx16)
                            Vy15 += 0;
                        else if (Vy15 < Vy17 + 70 && Vy15 > Vy17 &&
Vx15 == Vx17)
                            Vy15 += 0;
                        else
                            Vy15 -= r15;

                        if (Vy15 <= 0)
                            Vy15 += 700;
                        g.FillRectangle(c15, Vx15, Vy15, 25, 50);

                        if (carBottom >= 2)
                        {
                            Brush         c16         =         new
SolidBrush(Color.Orange);

                            if (Vy16 >= 400 && Vy16 <= 410 && (timer
< 10 || timer >= 240))
                                Vy16 += 0;
```

```
                                else if (Vy16 < Vy4 + 70 && Vy16 > Vy4
&& Vx16 == Vx4)
                                    Vy16 += 0;
                                else if (Vy16 < Vy15 + 70 && Vy16 > Vy15
&& Vx16 == Vx15)
                                    Vy16 += 0;
                                else if (Vy16 < Vy17 + 70 && Vy16 > Vy17
&& Vx16 == Vx17)
                                    Vy16 += 0;
                                else
                                    Vy16 -= r16;

                                if (Vy16 <= 0)
                                    Vy16 += 700;
                                g.FillRectangle(c16,  Vx16,  Vy16,  25,
50);

                                if (carBottom >= 3)
                                {
                                    Brush       c17       =       new
SolidBrush(Color.Coral);

                                    if (Vy17 >= 400 && Vy17 <= 410 &&
(timer < 10 || timer >= 240))
                                        Vy17 += 0;
                                    else if (Vy17 < Vy4 + 70 && Vy17 >
Vy4 && Vx17 == Vx4)
                                        Vy17 += 0;
                                    else if (Vy17 < Vy15 + 70 && Vy17 >
Vy15 && Vx17 == Vx15)
                                        Vy17 += 0;
                                    else if (Vy17 < Vy16 + 70 && Vy17 >
Vy16 && Vx17 == Vx16)
                                        Vy17 += 0;
                                    else
                                        Vy17 -= r17;

                                    if (Vy17 <= 0)
                                        Vy17 += 700;
                                    g.FillRectangle(c17,  Vx17,  Vy17,
25, 50);
                                }
                            }
                        }
                    }

                    if (radioButton6.Checked == true)
                    {
                        button2.Visible = true;
                        button3.Visible = true;
```

```
                    button4.Visible = true;
                    button5.Visible = true;

                    Brush c1 = new SolidBrush(Color.DeepPink);
                    Brush c2 = new SolidBrush(Color.Gray);
                    Brush c3 = new SolidBrush(Color.Black);
                    Brush c4 = new SolidBrush(Color.Yellow);
                    Brush c5 = new SolidBrush(Color.Brown);

                    if (Vx1 >= 240 && Vx1 <= 250 && (timer < 260 ||
timer >= 490))
                        Vx1 += 0;
                    else if (Vx1 > Vx6 - 70 && Vx1 < Vx6 && Vy1 ==
Vy6)
                        Vx1 += 0;
                    else if (Vx1 > Vx7 - 70 && Vx1 < Vx7 && Vy1 ==
Vy7)
                        Vx1 += 0;
                    else if (Vx1 > Vx8 - 70 && Vx1 < Vx8 && Vy1 ==
Vy8)
                        Vx1 += 0;
                    else if (Vx1 > Vx5 - 70 && Vx1 < Vx5 && Vy1 ==
Vy5)
                        Vx1 += 0;
                    else
                        Vx1 += r1;

                    if (Vx1 >= 700)
                        Vx1 -= 700;
                    g.FillRectangle(c1, Vx1, Vy1, 50, 25);

                    if (carLeft >= 1)
                    {
                        Brush c6 = new SolidBrush(Color.Pink);

                        if (Vx6 >= 240 && Vx6 <= 250 && (timer < 260
|| timer >= 490))
                            Vx6 += 0;
                        else if (Vx6 > Vx1 - 70 && Vx6 < Vx1 && Vy6
== Vy1)
                            Vx6 += 0;
                        else if (Vx6 > Vx7 - 70 && Vx6 < Vx7 && Vy6
== Vy7)
                            Vx6 += 0;
                        else if (Vx6 > Vx8 - 70 && Vx6 < Vx8 && Vy6
== Vy8)
                            Vx6 += 0;
                        else if (Vx6 > Vx5 - 70 && Vx6 < Vx5 && Vy6
== Vy5)
                            Vx6 += 0;
                        else
```

```
                    Vx6 += r6;

                if (Vx6 >= 700)
                    Vx6 -= 700;
                g.FillRectangle(c6, Vx6, Vy6, 50, 25);

                if (carLeft >= 2)
                {
                    Brush          c7          =          new
SolidBrush(Color.Orange);

                    if (Vx7 >= 240 && Vx7 <= 250 && (timer
< 260 || timer >= 490))
                        Vx7 += 0;
                    else if (Vx7 > Vx1 - 70 && Vx7 < Vx1 &&
Vy7 == Vy1)
                        Vx7 += 0;
                    else if (Vx7 > Vx6 - 70 && Vx7 < Vx6 &&
Vy7 == Vy6)
                        Vx7 += 0;
                    else if (Vx7 > Vx8 - 70 && Vx7 < Vx8 &&
Vy7 == Vy8)
                        Vx7 += 0;
                    else if (Vx7 > Vx5 - 70 && Vx7 < Vx5 &&
Vy7 == Vy5)
                        Vx7 += 0;
                    else
                        Vx7 += r7;

                    if (Vx7 >= 700)
                        Vx7 -= 700;
                    g.FillRectangle(c7, Vx7, Vy7, 50, 25);

                    if (carLeft >= 3)
                    {
                        Brush        c8        =        new
SolidBrush(Color.Coral);

                        if (Vx8 >= 240 && Vx8 <= 250 &&
(timer < 260 || timer >= 490))
                            Vx8 += 0;
                        else if (Vx8 > Vx1 - 70 && Vx8 < Vx1
&& Vy8 == Vy1)
                            Vx8 += 0;
                        else if (Vx8 > Vx6 - 70 && Vx8 < Vx6
&& Vy8 == Vy6)
                            Vx8 += 0;
                        else if (Vx8 > Vx7 - 70 && Vx8 < Vx7
&& Vy8 == Vy7)
                            Vx8 += 0;
```

```
                                else if (Vx8 > Vx5 - 70 && Vx8 < Vx5
&& Vy8 == Vy5)
                                    Vx8 += 0;
                                else
                                    Vx8 += r8;

                                if (Vx8 >= 700)
                                    Vx8 -= 700;
                                g.FillRectangle(c8, Vx8, Vy8, 50,
25);
                            }
                        }
                    }



                    if (Vx2 >= 400 && Vx2 <= 410 && (timer < 260 ||
timer >= 490))
                        Vx2 += 0;
                    else if (Vx2 < Vx9 + 70 && Vx2 > Vx9 && Vy2 ==
Vy9)
                        Vx2 += 0;
                    else if (Vx2 < Vx10 + 70 && Vx2 > Vx10 && Vy2 ==
Vy10)
                        Vx2 += 0;
                    else if (Vx2 < Vx11 + 70 && Vx2 > Vx11 && Vy2 ==
Vy11)
                        Vx2 += 0;
                    else if (Vx2 < Vx5 + 70 && Vx2 > Vx5 && Vy2 ==
Vy5)
                        Vx2 += 0;
                    else
                        Vx2 -= r2;

                    if (Vx2 <= 0)
                        Vx2 += 700;
                    g.FillRectangle(c2, Vx2, Vy2, 50, 25);

                    if (carRight >= 1)
                    {
                        Brush c9 = new SolidBrush(Color.Tomato);

                        if (Vx9 >= 400 && Vx9 <= 410 && (timer < 260
|| timer >= 490))
                            Vx9 += 0;
                        else if (Vx9 < Vx2 + 70 && Vx9 > Vx2 && Vy9
== Vy2)
                            Vx9 += 0;
                        else if (Vx9 < Vx10 + 70 && Vx9 > Vx10 &&
Vy9 == Vy10)
```

```
                        Vx9 += 0;
                    else if (Vx9 < Vx11 + 70 && Vx9 > Vx11 &&
Vy9 == Vy11)
                        Vx9 += 0;
                    else if (Vx9 < Vx5 + 70 && Vx9 > Vx5 && Vy9
== Vy5)
                        Vx9 += 0;
                    else
                        Vx9 -= r9;

                    if (Vx9 <= 0)
                        Vx9 += 700;
                    g.FillRectangle(c9, Vx9, Vy9, 50, 25);

                    if (carRight >= 2)
                    {
                        Brush         c10         =         new
SolidBrush(Color.Orange);

                        if (Vx10 >= 400 && Vx10 <= 410 && (timer
< 260 || timer >= 490))
                            Vx10 += 0;
                        else if (Vx10 < Vx2 + 70 && Vx10 > Vx2
&& Vy10 == Vy2)
                            Vx10 += 0;
                        else if (Vx10 < Vx9 + 70 && Vx10 > Vx9
&& Vy10 == Vy9)
                            Vx10 += 0;
                        else if (Vx10 < Vx11 + 70 && Vx10 > Vx11
&& Vy10 == Vy11)
                            Vx10 += 0;
                        else if (Vx10 < Vx5 + 70 && Vx10 > Vx5
&& Vy10 == Vy5)
                            Vx10 += 0;
                        else
                            Vx10 -= r10;

                        if (Vx10 <= 0)
                            Vx10 += 700;
                        g.FillRectangle(c10,  Vx10,  Vy10,  50,
25);

                        if (carRight >= 3)
                        {
                            Brush        c11        =        new
SolidBrush(Color.Coral);

                            if (Vx11 >= 400 && Vx11 <= 410 &&
(timer < 260 || timer >= 490))
                                Vx11 += 0;
```

```
                                    else if (Vx11 < Vx2 + 70 && Vx11 >
Vx2 && Vy11 == Vy2)
                                        Vx11 += 0;
                                    else if (Vx11 < Vx9 + 70 && Vx11 >
Vx9 && Vy11 == Vy9)
                                        Vx11 += 0;
                                    else if (Vx11 < Vx10 + 70 && Vx11 >
Vx10 && Vy11 == Vy10)
                                        Vx11 += 0;
                                    else if (Vx11 < Vx5 + 70 && Vx11 >
Vx5 && Vy11 == Vy5)
                                        Vx11 += 0;
                                    else
                                        Vx11 -= r11;

                                    if (Vx11 <= 0)
                                        Vx11 += 700;
                                    g.FillRectangle(c11,  Vx11,  Vy11,
50, 25);
                                }
                            }
                        }

                        if (Vy3 >= 240 && Vy3 <= 250 && (timer < 10 ||
timer >= 240))
                            Vy3 += 0;
                        else if (Vy3 > Vy12 - 70 && Vy3 < Vy12 && Vx3 ==
Vx12)
                            Vy3 += 0;
                        else if (Vy3 > Vy13 - 70 && Vy3 < Vy13 && Vx3 ==
Vx13)
                            Vy3 += 0;
                        else if (Vy3 > Vy14 - 70 && Vy3 < Vy14 && Vx3 ==
Vx14)
                            Vy3 += 0;
                        else if (Vy3 > Vy5 - 70 && Vy3 < Vy5 && Vx3 ==
Vx5)
                            Vy3 += 0;
                        else
                            Vy3 += r3;

                        if (Vy3 >= 700)
                            Vy3 -= 700;

                        g.FillRectangle(c3, Vx3, Vy3, 25, 50);

                        if (carTop >= 1)
                        {
                            Brush c12 = new SolidBrush(Color.Pink);
```

```
                        if (Vy12 >= 240 && Vy12 <= 250 && (timer <
10 || timer >= 240))
                            Vy12 += 0;
                        else if (Vy12 > Vy3 - 70 && Vy12 < Vy3 &&
Vx12 == Vx3)
                            Vy12 += 0;
                        else if (Vy12 > Vy13 - 70 && Vy12 < Vy13 &&
Vx12 == Vx13)
                            Vy12 += 0;
                        else if (Vy12 > Vy14 - 70 && Vy12 < Vy14 &&
Vx12 == Vx14)
                            Vy12 += 0;
                        else if (Vy12 > Vy5 - 70 && Vy12 < Vy5 &&
Vx12 == Vx5)
                            Vy12 += 0;
                        else
                            Vy12 += r12;

                        if (Vy12 >= 700)
                            Vy12 -= 700;
                        g.FillRectangle(c12, Vx12, Vy12, 25, 50);

                        if (carTop >= 2)
                        {
                            Brush         c13         =         new
SolidBrush(Color.Orange);

                            if (Vy13 >= 240 && Vy13 <= 250 && (timer
< 10 || timer >= 240))
                                Vy13 += 0;
                            else if (Vy13 > Vy3 - 70 && Vy13 < Vy3
&& Vx13 == Vx3)
                                Vy13 += 0;
                            else if (Vy13 > Vy12 - 70 && Vy13 < Vy12
&& Vx13 == Vx12)
                                Vy13 += 0;
                            else if (Vy13 > Vy14 - 70 && Vy13 < Vy14
&& Vx13 == Vx14)
                                Vy13 += 0;
                            else if (Vy13 > Vy5 - 70 && Vy13 < Vy5
&& Vx13 == Vx5)
                                Vy13 += 0;
                            else
                                Vy13 += r13;

                            if (Vy13 >= 700)
                                Vy13 -= 700;
                            g.FillRectangle(c13,  Vx13,  Vy13,  25,
50);

                            if (carTop >= 3)
```

```
                        {
                            Brush        c14        =        new
SolidBrush(Color.Coral);

                            if (Vy14 >= 240 && Vy14 <= 250 &&
(timer < 10 || timer >= 240))
                                Vy14 += 0;
                            else if (Vy14 > Vy3 - 70 && Vy14 <
Vy3 && Vx14 == Vx3)
                                Vy14 += 0;
                            else if (Vy14 > Vy12 - 70 && Vy14 <
Vy12 && Vx14 == Vx12)
                                Vy14 += 0;
                            else if (Vy14 > Vy13 - 70 && Vy14 <
Vy13 && Vx14 == Vx13)
                                Vy14 += 0;
                            else if (Vy14 > Vy5 - 70 && Vy14 <
Vy5 && Vx14 == Vx5)
                                Vy14 += 0;
                            else
                                Vy14 += r14;

                            if (Vy14 >= 700)
                                Vy14 -= 700;
                            g.FillRectangle(c14,  Vx14,  Vy14,
25, 50);
                        }
                    }
                }


                if (Vy4 >= 400 && Vy4 <= 410 && (timer < 10 ||
timer >= 240))
                    Vy4 += 0;
                else if (Vy4 < Vy15 + 70 && Vy4 > Vy15 && Vx4 ==
Vx15)
                    Vy4 += 0;
                else if (Vy4 < Vy16 + 70 && Vy4 > Vy16 && Vx4 ==
Vx16)
                    Vy4 += 0;
                else if (Vy4 < Vy17 + 70 && Vy4 > Vy17 && Vx4 ==
Vx17)
                    Vy4 += 0;
                else if (Vy4 < Vy5 + 70 && Vy4 > Vy5 && Vx4 ==
Vx5)
                    Vy4 += 0;
                else
                    Vy4 -= r4;

                if (Vy4 <= 0)
                    Vy4 += 700;
```

```
                    g.FillRectangle(c4, Vx4, Vy4, 25, 50);

                    if (carBottom >= 1)
                    {
                        Brush c15 = new SolidBrush(Color.Pink);

                        if (Vy15 >= 400 && Vy15 <= 410 && (timer <
10 || timer >= 240))
                            Vy15 += 0;
                        else if (Vy15 < Vy4 + 70 && Vy15 > Vy4 &&
Vx15 == Vx4)
                            Vy15 += 0;
                        else if (Vy15 < Vy16 + 70 && Vy15 > Vy16 &&
Vx15 == Vx16)
                            Vy15 += 0;
                        else if (Vy15 < Vy17 + 70 && Vy15 > Vy17 &&
Vx15 == Vx17)
                            Vy15 += 0;
                        else if (Vy15 < Vy5 + 70 && Vy15 > Vy5 &&
Vx15 == Vx5)
                            Vy15 += 0;
                        else
                            Vy15 -= r15;

                        if (Vy15 <= 0)
                            Vy15 += 700;
                        g.FillRectangle(c15, Vx15, Vy15, 25, 50);

                        if (carBottom >= 2)
                        {
                            Brush         c16         =         new
SolidBrush(Color.Orange);

                            if (Vy16 >= 400 && Vy16 <= 410 && (timer
< 10 || timer >= 240))
                                Vy16 += 0;
                            else if (Vy16 < Vy4 + 70 && Vy16 > Vy4
&& Vx16 == Vx4)
                                Vy16 += 0;
                            else if (Vy16 < Vy15 + 70 && Vy16 > Vy15
&& Vx16 == Vx15)
                                Vy16 += 0;
                            else if (Vy16 < Vy17 + 70 && Vy16 > Vy17
&& Vx16 == Vx17)
                                Vy16 += 0;
                            else if (Vy16 < Vy5 + 70 && Vy16 > Vy5
&& Vx16 == Vx5)
                                Vy16 += 0;
                            else
                                Vy16 -= r16;
```

```
                            if (Vy16 <= 0)
                                Vy16 += 700;
                            g.FillRectangle(c16,  Vx16,  Vy16,  25,
50);

                            if (carBottom >= 3)
                            {
                                Brush       c17       =       new
SolidBrush(Color.Coral);

                                if (Vy17 >= 400 && Vy17 <= 410 &&
(timer < 10 || timer >= 240))
                                    Vy17 += 0;
                                else if (Vy17 < Vy4 + 70 && Vy17 >
Vy4 && Vx17 == Vx4)
                                    Vy17 += 0;
                                else if (Vy17 < Vy15 + 70 && Vy17 >
Vy15 && Vx17 == Vx15)
                                    Vy17 += 0;
                                else if (Vy17 < Vy16 + 70 && Vy17 >
Vy16 && Vx17 == Vx16)
                                    Vy17 += 0;
                                else if (Vy17 < Vy5 + 70 && Vy17 >
Vy5 && Vx17 == Vx5)
                                    Vy17 += 0;
                                else
                                    Vy17 -= r17;

                                if (Vy17 <= 0)
                                    Vy17 += 700;
                                g.FillRectangle(c17,  Vx17,  Vy17,
25, 50);
                            }
                        }
                    }


                    if (Vy5<=365 && Vy5>=355)
                    {
                        if (Vx5 >= 240 && Vx5 <= 250 && (timer < 260
|| timer >= 490))
                            Vx5 += 0;
                        else  if  (Vx5  >  Vx1  -  70  &&  Vx5<Vx1  &&
Vy5==Vy1)
                            Vx5 += 0;
                        else if (Vx5 > Vx6 - 70 && Vx5 < Vx6 && Vy5
== Vy6)
                            Vx5 += 0;
```

```
                    else if (Vx5 > Vx7 - 70 && Vx5 < Vx7 && Vy5
== Vy7)
                        Vx5 += 0;
                    else if (Vx5 > Vx8 - 70 && Vx5 < Vx8 && Vy5
== Vy8)
                        Vx5 += 0;
                    else
                        Vx5 += 10;
                    g.FillRectangle(c5, Vx5, Vy5, 50, 25);
                }
                if (Vy5 <= 315 && Vy5 >= 305)
                {
                    if (Vx5 >= 400 && Vx5 <= 410 && (timer < 260
|| timer >= 490))
                        Vx5 += 0;
                    else if (Vx5 < Vx2 + 70 && Vx5 > Vx2 && Vy5
== Vy2)
                        Vx5 += 0;
                    else if (Vx5 < Vx9 + 70 && Vx5 > Vx9 && Vy5
== Vy9)
                        Vx5 += 0;
                    else if (Vx5 < Vx10 + 70 && Vx5 > Vx10 &&
Vy5 == Vy10)
                        Vx5 += 0;
                    else if (Vx5 < Vx11 + 70 && Vx5 > Vx11 &&
Vy5 == Vy11)
                        Vx5 += 0;
                    else
                        Vx5 -= 10;
                    g.FillRectangle(c5, Vx5, Vy5, 50, 25);
                }
                if (Vx5 >= 305 && Vx5 <= 315)
                {
                    if (Vy5 >= 240 && Vy5 <= 250 && (timer < 10
|| timer >= 240))
                        Vy5 += 0;
                    else if (Vy5 > Vy3 - 70 && Vy5 < Vy3 && Vx5
== Vx3)
                        Vy5 += 0;
                    else if (Vy5 > Vy12 - 70 && Vy5 < Vy12 &&
Vx5 == Vx12)
                        Vy5 += 0;
                    else if (Vy5 > Vy13 - 70 && Vy5 < Vy13 &&
Vx5 == Vx13)
                        Vy5 += 0;
                    else if (Vy5 > Vy14 - 70 && Vy5 < Vy14 &&
Vx5 == Vx14)
                        Vy5 += 0;
                    else
                        Vy5 += 10;
                    g.FillRectangle(c5, Vx5, Vy5, 25, 50);
```

```
                    }
                    if (Vx5 >= 355 && Vx5 <= 365)
                    {
                        if (Vy4 >= 410 && Vy4 <= 420 && (timer < 10
|| timer >= 240))
                            Vy5 -= 0;
                        else if (Vy5 < Vy4 + 70 && Vy5 > Vy4 && Vx5
== Vx4)
                            Vy5 += 0;
                        else if (Vy5 < Vy15 + 70 && Vy5 > Vy15 &&
Vx5 == Vx15)
                            Vy5 += 0;
                        else if (Vy5 < Vy16 + 70 && Vy5 > Vy16 &&
Vx5 == Vx16)
                            Vy5 += 0;
                        else if(Vy5 < Vy17 + 70 && Vy5 > Vy17 && Vx5
== Vx17)
                            Vy5 += 0;
                        else
                            Vy5 -= 10;
                        g.FillRectangle(c5, Vx5, Vy5, 25, 50);
                    }

                    if (Vy5>=700)
                        Vx5 += 50;
                    if (Vx5 >= 700)
                        Vy5 -= 50;
                    if (Vy5 <= 0)
                        Vx5 -= 50;
                    if (Vx5 <= 0)
                        Vy5 += 50;

                }



            }

            private void button2_Click(object sender, EventArgs e)
            {
                if (carLeft<3)
                carLeft++;

            }

            private void button3_Click(object sender, EventArgs e)
            {
                if (carTop<3)
                carTop++;
            }
```

```
        private void button4_Click(object sender, EventArgs e)
        {
            if (carRight < 3)
                carRight++;
        }

        private void button5_Click(object sender, EventArgs e)
        {
            if (carBottom < 3)
                carBottom++;
        }
    }
}
```