НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

«До захисту допущено»
Завідувач кафедри
_____ Олег Чертов
«___» _____ 2023 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Наука про дані та математичне моделювання»
спеціальності 113 «Прикладна математика»
на тему: «Система моделювання епідемічних, ендемічних та ерадикаційних процесів з урахуванням моделі людської мобільності»

Виконав:
студент IV курсу, групи КМ-92
Маляренко Микита Артемович

_____

Керівник:
асистент
Громова Вікторія Вікторівна

_____

Консультант з нормоконтролю:
Старший викладач,
Мальчиков Володимир Вікторович

_____

Рецензент:
Старший викладач каф. СПіСКС, канд. техн. наук
Наливайчук Микола Васильович

_____

Засвідчую, що в цій дипломній роботі
немає запозичень із праць інших авторів
без відповідних посилань.
студент Маляренко М.А._____

Київ — 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра прикладної математики

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 113 «Прикладна математика»
Освітньо-професійна програма «Наука про дані та математичне моделювання»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О. Р. Чертов
«___» _____ 2023 р.

ЗАВДАННЯ
на дипломну роботу студентові
Маляренко Микиті Артемовичу

1. Тема роботи: «Система моделювання епідемічних, ендемічних та ерадикаційних процесів з урахуванням моделі людської мобільності», керівник роботи Громова Вікторія Вікторівна, асистент, затверджені наказом по університету від «31» травня 2023 р. № 2108-С
2. Термін подання студентом роботи: «12» червня 2023 р.
3. Вихідні дані до роботи: симуляція розповсюдження хвороби з урахуванням симуляція людської мобільності.
4. Зміст роботи: виконати огляд та розробити критерії оцінки методів для симуляції розповсюдження хвороби, симуляції людської мобільності та обрати методи. Спроектувати та програмно реалізувати систему розповсюдження хвороби, провести тестування системи.
5. Перелік ілюстративного матеріалу: блок схеми розроблених алгоритмів, діаграма компонентів системи, графіки, зображення.
6. Дата видачі завдання: «06» лютого 2023 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Огляд літератури за тематикою та збір даних	12.11.2022	
2	Проведення порівняльного аналізу програмних засобів з генерації резюме	14.12.2022	
3	Проведення порівняльного аналізу математичних методів класифікації тексту	24.12.2022	
4	Підготовка матеріалів першого розділу роботи	01.02.2023	
5	Розроблення математичного забезпечення для багатозначної класифікації тексту	01.03.2023	
6	Підготовка матеріалів другого розділу роботи	15.03.2023	
7	Підготовка матеріалів третього розділу роботи	05.04.2023	
8	Розроблення програмного забезпечення для генерації резюме на основі бажаної вакансії та навиків працівника	15.04.2023	

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
9	Підготовка матеріалів четвертого розділу роботи	03.05.2023	
10	Оформлення пояснювальної записки	01.06.2023	

Студент _____ Микита МАЛЯРЕНКО

Керівник роботи _____ Вікторія ГРОМОВА

АНОТАЦІЯ

Дипломна роботу виконано на 63 аркушах, включно з 2 додатками та списком використаної літератури, що містить 12 джерел. В роботі представлено 12 ілюстрацій та 4 таблиці.

Мета цього дослідження полягає у створенні програмного рішення, яке об'єднує симуляцію розповсюдження хвороб та моделювання епідемічних, ендемічних та ерадикаційних процесів.

У роботі здійснено аналіз та порівняння моделей симуляції людської мобільності та моделей симуляції розповсюдження хвороб на основі таких критеріїв як масштабованість, гнучкість та точність. Деякі з вибраних моделей були вдосконалені або адаптовані для потреб симуляції. Було створено алгоритм симуляції, який був інтегрований у розроблене програмне забезпечення.

Ключові слова: Епідемія, ендемія, ерадикація, SEIR, Gravity model, людська мобільність, математичне моделювання.

ABSTRACT

The thesis is written on 63 pages, including 2 appendices and a list of references containing 12 sources. The work contains 12 illustrations and 4 tables.

The purpose of this study is to create a software solution that combines disease spread simulation and modeling of epidemic, endemic, and eradication processes.

The paper analyzes and compares human mobility simulation models and disease spread simulation models based on criteria such as scalability, flexibility, and accuracy. Some of the selected models were improved or adapted to the needs of the simulation. A simulation algorithm was created and integrated into the developed software.

Keywords: Epidemic, endemic, eradication, SEIR, Gravity model, human mobility, mathematical modeling.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

FRED - Framework for Reconstructing Epidemiological Dynamics

GMM - General Mobility Model

MVC - Model-View-Controller

MVVM - Model-View-ViewModel

OOD - Object-Oriented Design

UI - User Interface

ВСТУП

З приходом 21-го століття значно зросла частота і серйозність наслідків епідемій та пандемій, найсвіжішим і найвпливовішим прикладом яких є пандемія COVID-19. Здатність прогнозувати, управляти та контролювати поширення інфекційних захворювань має першорядне значення для громадського здоров'я. Ця дипломна робота присвячена розробці математичного та програмного забезпечення для моделювання процесів епідемії, ендемії та ерадикації, з особливим акцентом на моделі внутрішньоміської мобільності людей.

Ця програмна система дозволить користувачам моделювати різні сценарії поширення та контролю захворювань у міському середовищі, надаючи цінну інформацію, яка може бути використана при розробці політики та плануванні у сфері громадського здоров'я.

Кінцевою метою цієї дипломної роботи є підвищення нашої спроможності прогнозувати та управляти інфекційними захворюваннями, приділяючи особливу увагу міському середовищу. Розробляючи надійні математичні моделі та зручне програмне забезпечення, цей проект має на меті забезпечити цінний ресурс для тих, хто працює у сфері громадського здоров'я. Інструменти, створені в результаті цієї роботи, не лише сприятимуть глибшому розумінню процесів поширення та контролю захворювань, але й нададуть практичну інформацію для політиків, чиновників у сфері охорони здоров'я та дослідників.

1 ПОСТАНОВКА ЗАДАЧІ

Метою даної дипломної роботи є створення математичного та програмного забезпечення системи моделювання епідемічних, ендемічних та ерадикаційних процесів з урахуванням внутрішньоміської моделі людської мобільності.

При розробленні відповідного забезпечення потрібно розв’язати наступні завдання:

а) Математичні

1) Внутрішньоміські переміщення людей

1.1) Розробити критерії оцінки моделей

1.2) Проаналізувати та порівняти наявні моделі

1.3) Вибір та, якщо необхідно, адаптація моделі під потреби дипломної роботи.

2) Епідемічні, ендемічні та ерадикаційні процесів

2.1) Розробити критерії оцінки моделей поширення хвороби

2.2) Проаналізувати та порівняти наявні моделі

2.3) Вибір та, якщо необхідно, адаптація моделі під потреби дипломної роботи.

б) Програмні

1) Порівняти та обрати ПЗ на базі якого буде виконуватись розробка симуляції

2) Розробити алгоритму симуляції

3) Розробити архітектури

4) Реалізувати алгоритм на базі обраного програмного забезпечення з урахуванням розробленої архітектур

Імплементована система має задовольняти наступні вимоги:

1. Мати візуальну репрезентацію симуляції
2. Можливість динамічно редагувати параметри симуляції

2 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Внутрішньоміські переміщення людей

Наука про внутрішньоміську людську мобільність - це міждисциплінарна галузь, яка всебічно вивчає, як, чому і куди люди пересуваються в міському середовищі [1]. Вона передбачає збір даних за допомогою таких методів, як опитування, перепис населення, дані мобільних телефонів, GPS або дані соціальних мереж, які можуть надати вичерпну інформацію про пересування людей.

Потім ці дані ретельно вивчаються для виявлення закономірностей і тенденцій за допомогою статистичного аналізу, просторового аналізу і передових обчислювальних методів, таких як машинне навчання. Мета полягає в тому, щоб зрозуміти фактори, які мотивують рух, і розробити моделі, здатні передбачити майбутні моделі руху.

Результати цих аналізів інтерпретуються стосовно конкретної досліджуваної міської території, що дозволяє приймати рішення, пов'язані з міським плануванням, транспортом, громадським здоров'ям тощо. Наприклад, розуміння моделей пересування на роботу може допомогти у розробці маршрутів громадського транспорту, тоді як розуміння мобільності населення може допомогти у плануванні житлового будівництва та комунальних послуг. Більше того, наука про внутрішньоміську людську мобільність не зосереджена виключно на розумінні фізичних переміщень. Вона також намагається зрозуміти соціальні, економічні та екологічні фактори, які керують цими переміщеннями, включаючи такі змінні, як дохід, освіта, вік та сімейний стан.

2.1.1 Виділення критеріїв оцінки моделей

Для оцінки та порівняння методі виділимо наступні критерії:

а) Масштабованість: GMM (General Mobility Model) повинен бути здатним працювати з великою кількістю агентів. Це означає, що вона повинна бути розроблена таким чином, щоб її можна було масштабувати зі збільшенням кількості агентів.

б) Точність: GMM має бути здатною точно відображати переміщення людей у міському середовищі. Це означає, що вона повинна базуватися на реалістичних припущеннях про те, як люди пересуваються і взаємодіють у місті.

в) Часові та просторові зміни: Модель повинна враховувати часові (залежні від часу) і просторові (залежні від місцезнаходження) зміни в поведінці людей і темпах передачі захворювань.

г) Гнучкість: GMM має бути достатньо гнучким, щоб моделювати різні сценарії. Наприклад, вона повинна дозволяти змінювати такі параметри, як щільність населення, швидкість поширення хвороб або соціальну поведінку.

2.1.2 Модель випадкових блукань

У базовій моделі випадкового блукання кожен вузол переміщується зі свого поточного положення в нове, випадково обираючи напрямок і швидкість. Нова позиція визначається на основі попередньої позиції, швидкості та напрямку [2]. Швидкість і напрямок вибираються із заздалегідь визначених діапазонів. Як тільки

мобільний вузол досягає межі симуляції, він може або "відскочити" назад в область симуляції, або обернутися на іншу сторону.

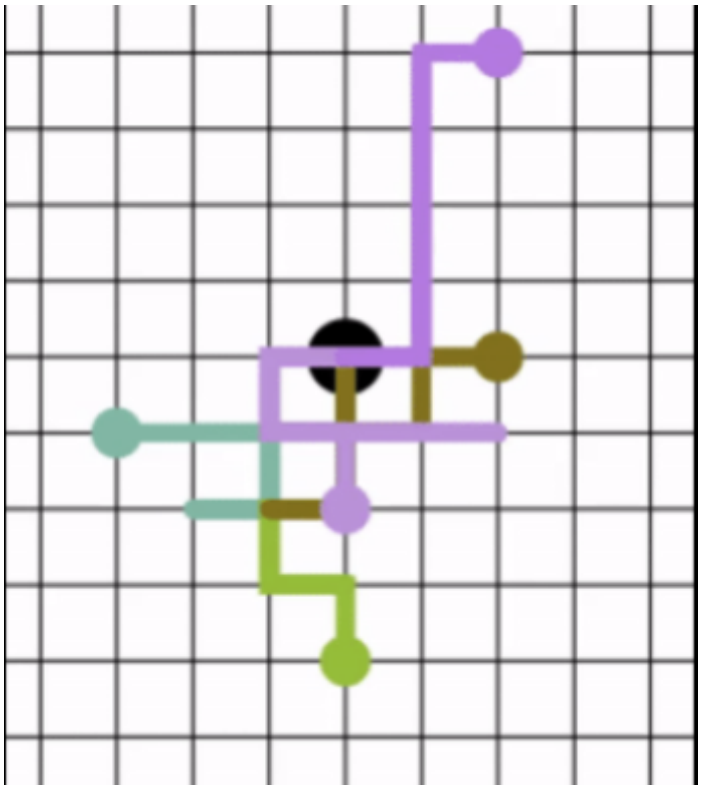

Рисунок 2.1 - П'ять випадкових прогулянок по вісім кроків від центральної точки.

Модель випадкових блукань можно записати наступним чином для часового ряду X_t:

$$X_t = X_{t-1} + e_t \qquad (2.1)$$

де X_t значення на часовому проміжку t, X_{t-1} - значення в часовому періоді плюс випадковий шок e_t (значення члена похибки на часовому проміжку) [3].

Оцінка за критеріями:

а) Масштабованість: Висока. Модель випадкових блукань добре масштабується. Оскільки кожен вузол працює незалежно від інших, додаткові вузли можуть бути

додані без збільшення складності моделі. Продуктивність моделі не буде суттєво погіршуватися з додаванням нових вузлів, що робить її придатною для сценаріїв за участю великої кількості осіб.

б) Точність: Низька. Точність моделі випадкових блукань у відображенні пересування людей у міському середовищі є більш нюансованим питанням. Хоча модель добре відображає випадковий рух, вона не враховує реалістичну поведінку людини, яка часто включає шаблони та цілі. Тому вона може неточно відображати те, як люди пересуваються містом, де рух часто є цілеспрямованим (наприклад, поїздка на роботу, похід до супермаркету) і відбувається за певними маршрутами (наприклад, дорогами, пішохідними доріжками).

в) Часові та просторові зміни: Низька. Модель випадкових блукань за своєю суттю не враховує часові (залежні від часу) або просторові (залежні від місця розташування) зміни в поведінці людей або темпи передачі захворювань. Всі переміщення в моделі є випадковими і не залежать від часу доби, місцезнаходження або будь-яких інших факторів. Хоча потенційно можна було б модифікувати модель, щоб включити ці фактори, такі модифікації збільшили б складність моделі і віддалили б її від простого випадкового блукання.

г) Гнучкість: Нормальна. Модель випадкових блукань є гнучкою в тому сенсі, що такі параметри, як напрямок і швидкість, можуть бути легко скориговані. Однак їй не вистачає можливостей для моделювання різних сценаріїв, які передбачають більш складну поведінку, наприклад, зміну щільності населення, передачу хвороб або соціальну поведінку. Модель не враховує мотивації, що стоять за переміщенням, або вплив навколишнього середовища, окрім простих граничних умов.

Хоча модель випадкових блукань має високі показники простоти та масштабованості, їй бракує точності, гнучкості та здатності враховувати часові та просторові варіації. Вона найкраще підходить для моделювання, де простота пересування є першочерговою, а реалістичність міського руху не є основною проблемою.

2.1.3 Політ Леві

Математична модель, яка використовується для опису поведінки об'єкта (або кількох об'єктів) під час його переміщення в просторі, в основному вона використовується в таких галузях, як екологія, фізика і навіть економіка. Унікальність моделі полягає в тому, що в її основі лежить ідея про те, що об'єкт має тенденцію не лише робити маленькі кроки, але й іноді робити дуже великі. Ці великі кроки або "польоти" беруться зі статистичного розподілу, часто зі степеневого розподілу.

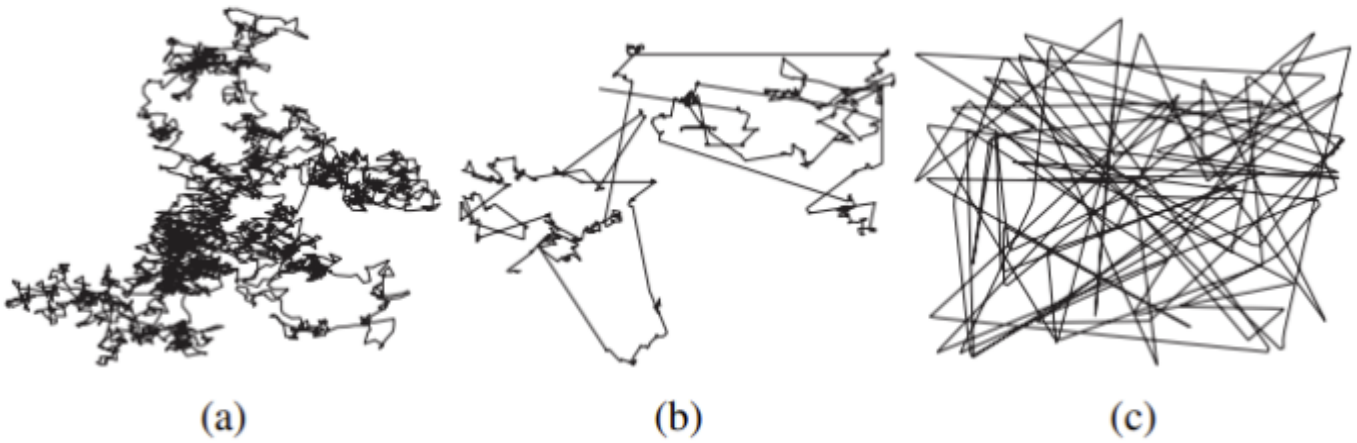

Рисунок 2.2 - Приклади траєкторій (a) броунівського руху, (b) ходьби Леві та (c)Випадкова точкового шляху [4]

Оцінка за критеріями:

а) Масштабованість: Нормальна. Модель польоту Леві може бути масштабованою. Однак це значною мірою залежатиме від того, наскільки ефективно вона буде

реалізована. Тому масштабованість є помірно хорошою, але ефективність буде залежати від специфіки реалізації.

б) Точність: Нормальна. Точність моделі польоту Леві може сильно залежати від ситуації. У деяких сценаріях, таких як рух тварин або певні види людської мобільності, вона може бути досить точною. Однак у складному міському середовищі, де на рух впливають різноманітні фактори, такі як інфраструктура, соціальні системи та інші обмеження, вона не завжди може забезпечити точне зображення.

в) Гнучкість: Нормальна. Levy Flight є досить гнучким з точки зору типів руху, які він може моделювати. Налаштовуючи такі параметри, як форма розподілу, з якого будуються польоти, можна зобразити різні типи руху. Однак, він не може так легко врахувати такі зміни, як щільність населення, передаваність хвороб або соціальну поведінку, оскільки в першу чергу має справу з просторовими переміщеннями.

г) Часова і просторова варіативність: Нормальна. Модель в першу чергу стосується просторових варіацій, оскільки він описує рух у просторі. Для часової варіації може знадобитися додаткове нашарування моделей або модифікації, що може додати складності. Таким чином, у своїй стандартній формі вона не охоплює часові варіації. Що стосується просторових варіацій, то він чудово справляється з цим завданням, оскільки в першу чергу створений саме для цього.

Загалом, модель “Політ Леві” має свої сильні сторони, особливо в простоті та врахуванні просторових варіацій. Однак, коли йдеться про гнучкість і точність, особливо в складних міських умовах, вона може потребувати коригувань або додаткових шарів, щоб ефективно представляти бажану систему.

2.1.4 Манхеттенська сіткова модель мобільності

Це модель мобільності, яка використовується для імітації руху вузлів (що представляють людей, транспортні засоби тощо) у сітчастому міському середовищі. Модель натхненна плануванням Манхеттена в Нью-Йорку, який характеризується регулярною сіткою вулиць і проспектів.

У цій моделі вузли рухаються по прямих шляхах, які відповідають "вулицям" сітки. Коли вузол досягає перехрестя (з'єднання двох вулиць), він може або продовжити рух по тому ж шляху, або вибрати новий напрямок, що відповідає повороту на 90 градусів. Вибір напрямку на перехрестях може бути випадковим, а може підпорядковуватися певним правилам або ймовірностям. Рух вузлів також може бути обмежений "вулицями з одностороннім рухом" (тобто шляхами, які можна пройти тільки в одному напрямку).

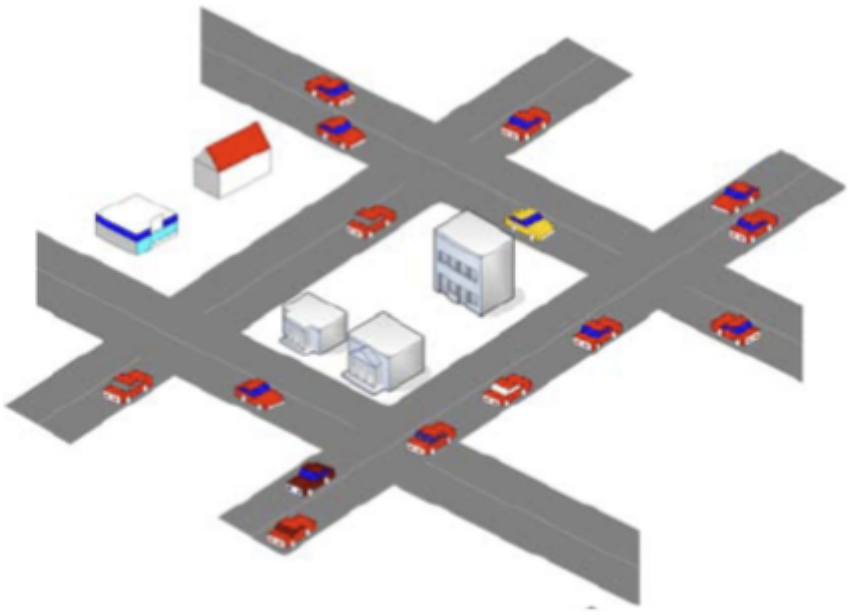

Рисунок 2.3 - Манхеттенська модель мобільності

Швидкість руху може бути постійною або змінною, залежно від специфіки моделі. Якщо швидкість змінна, то вона може змінюватися випадковим чином, або на неї можуть впливати такі фактори, як затори (наявність інших вузлів на тому ж шляху).

Оцінка за критеріями:

а) Масштабованість: Висока. Манхеттенська модель мобільності є масштабованою, оскільки кожен вузол працює незалежно від інших, рухаючись вздовж ліній сітки відповідно до визначених правил. Однак модель вимагає більше обчислювальних ресурсів порівняно з простішими моделями, такими як модель випадкових блукань, особливо коли задіяна велика кількість вузлів і складна сітка. Тим не менш, продуктивність моделі не повинна суттєво погіршуватися зі збільшенням кількості вузлів

б) Точність: Нормальна. Точність моделі Манхеттенської сітки мобільності у відображенні руху людей у міському середовищі є вищою, ніж у моделі випадкових блукань. Ця модель враховує той факт, що в місті люди, як правило, рухаються певними шляхами (наприклад, дорогами) і можуть змінювати напрямок руху лише в певних точках (наприклад, на перехрестях). Таким чином, ця модель може забезпечити більш точне відображення руху в містах з сітчастим плануванням.

в) Часові та просторові зміни: Низька. За замовчуванням модель Манхеттенської сітки мобільності не враховує часові та просторові зміни. Однак, оскільки модель працює на сітці, її можна поєднати з іншими моделями або додатковим кодом, щоб врахувати ці зміни. Наприклад, можна врахувати затори на дорогах у різний час доби або варіації у схемах руху на різних ділянках мережі. Таким чином, з додатковою складністю можна врахувати часові та просторові зміни.

г) Гнучкість: Низька. Манхеттенська модель мобільності є гнучкою в тому сенсі, що такі параметри, як швидкість і напрямок руху на перехрестях, можуть бути скориговані. Вона також дозволяє налаштовувати схему мережі відповідно до

конкретного сценарію. Однак, подібно до моделі випадкових блукань, вона не враховує більш складну поведінку, наприклад, зміни в щільності населення, передачу хвороб або соціальну поведінку.

Таким чином, Манхеттенська модель сітчастої мобільності є помірно простою і масштабованою моделлю, яка пропонує покращену точність для моделювання пересування в міському середовищі з сітчастим плануванням. Хоча вона є більш гнучкою, ніж модель випадкового блукання, вона все ще потребує додаткової складності, щоб врахувати більш реалістичну поведінку та часові або просторові варіації.

2.1.5 Gravity model

Популярна модель у моделюванні просторової дифузії та мобільності. Вона названа на честь рівняння гравітації Ньютона, яке використовується як аналогія для пояснення людської мобільності. Модель загалом стверджує, що переміщення з одного місця в інше прямо пропорційне добутку чисельності населення в цих двох місцях і обернено пропорційне відстані між ними.

Gravity model - це найпростіша гравітаційна модель з одним оцінюваним параметром, θ [5].

$$\lambda_{ij} = \theta\left(\frac{N_i N_j}{d_{ij}}\right) \qquad (2.2)$$

Параметр θ діє як константа пропорційності, яка масштабує добуток чисельності населення регіону походження та регіону призначення (N_i) та (N_j), поділений на відстань між і та j.

Оцінка за критеріями:

а) Масштабованість: Висока. Gravity model може працювати з великою кількістю осіб та локацій. Зі збільшенням кількості особин або місцезнаходжень обчислювальна складність може зростати, але не суттєво.

б) Точність: Висока. Гравітаційна модель може точно відображати рух людей у міському середовищі. Вона враховує чисельність населення в пункті відправлення та призначення, а також відстань між ними, що є важливими факторами, які впливають на пересування людей у місті.

в) Гнучкість: Висока. Гравітаційну модель можна параметризувати різними способами для моделювання різних сценаріїв. Наприклад, експонента фактора відстані може бути скоригована, щоб відобразити різні припущення про те, як відстань впливає на пересування.

г) Часова та просторова варіація: Нормальна. Гравітаційна модель враховує просторову варіацію через фактор відстані, але не враховує часову варіацію. Однак її можна адаптувати для цього, дозволивши параметрам змінюватися в часі.

Підсумовуючи, гравітаційна модель забезпечує хороший баланс простоти, масштабованості, точності та гнучкості, а також добре враховує просторову варіацію. Її найбільшим обмеженням може бути врахування часових змін

2.1.6 Scaled Power Law Gravity Model

Scaled Power Law Gravity Model є більш нюансованим варіантом Gravity model. Ця модель включає масштабований степеневий закон у традиційну гравітаційну модель і вимагає оцінки трьох параметрів ω, ρ та α [5].

$$\lambda_{ij} = N_j^{\omega}(1 + \frac{d_{ij}}{\rho})^{-\alpha} \quad (2.3)$$

Де, ω коригує силу притягання для кожного j населених пунктів призначення, ρ масштабує відстань від і до j та α визначає ядро степеневого закону відстані.

Оцінка за критеріями:

а) Масштабованість: Висока. Подібно до гравітаційної моделі, масштабований степеневий закон - гравітаційна модель може працювати з великою кількістю осіб і місць. Обчислювальна складність може дещо збільшитися з включенням більшої кількості параметрів, але не суттєво.

б) Точність: Дуже висока. Ця модель є вдосконаленою гравітаційною моделлю і розроблена для усунення деяких обмежень останньої. Завдяки включенню масштабованого степеневого закону, вона може більш точно врахувати нюанси людської мобільності та просторових взаємодій, які можуть сильно відрізнятися залежно від різних факторів.

в) Гнучкість: Висока. Масштабований степеневий закон - гравітаційна модель, як і базова гравітаційна модель, може бути параметризована по-різному для різних сценаріїв. Додаткові параметри ω, ρ і α забезпечують більшу гнучкість у моделюванні складних моделей людської мобільності та просторових взаємодій.

г) Часова та просторова варіація: Висока. Масштабований степеневий закон - гравітаційна модель зберігає здатність гравітаційної моделі враховувати просторову варіацію. Крім того, додаткові параметри дозволяють цій моделі враховувати більш різноманітні форми часової варіації, що робить її потенційно більш надійною у моделюванні змін з часом.

Загалом Масштабований степеневий закон, є вдосконаленим варіантом традиційної гравітаційної моделі в моделюванні просторової дифузії та мобільності. Хоча вона дещо складніша через включення трьох параметрів (ω, ρ і α), вона пропонує вищий рівень точності та гнучкості. Масштабованість моделі залишається

високою, вона здатна обробляти велику кількість осіб і місць. Зокрема, вона включає масштабований степеневий закон, що дає змогу врахувати нюанси людської мобільності та просторових взаємодій. Ця модель є потенційно більш надійною у врахуванні різноманітних часових варіацій, що розширює її можливості щодо просторових варіацій.

2.1.7 Висновки до розділу

У даному розділі відбулася детальна оцінка декількох загальноприйнятих моделей мобільності, з оцінюванням за чотирма критичними параметрами: Масштабованість, точність, чутливість до часових і просторових змін та гнучкість. Після ретельного аналізу виявилось, що гравітаційна модель із маштабованим законом розподілу потужності виявляється найбільш придатною для використання.

Таблиця 2.1 - Порівняльна таблиця моделей симуляції людської мобільності

Модель	Масштабованість	Точність	Часові та просторові зміни	Гнучкість
Модель випадкових блукань	Висока	Низька	Низька	Нормальна
Політ Леві	Нормальна	Нормальна	Нормальна	Нормальна
Манхеттенська сіткова модель мобільності	Висока	Нормальна	Низька	Низька
Gravity model	Висока	Висока	Нормальна	Висока
Scaled Power Law Gravity Model	Висока	Дуже висока	Висока	Висока

Зазначена модель використовує масштабований степеневий закон і потребує оцінювання трьох параметрів для ефективної реалізації. Хоча задовольняє більшість встановлених критеріїв, виявлено обмеження - нездатність враховувати зміни в часі. Тому, потребує внесення змін у зазначену модель для виправлення цього недоліку.

Додамо до моделі параметр t, який є часом, та приймає значення $t \in [0; 24]$

$$\lambda_{ij}(t) = N_j^{\omega}(t)\,(1 + \frac{d_{ij}}{\rho(t)})^{-\alpha(t)} \qquad (2.4)$$

Населення регіону розраховується як добуток щільності населення на площу регіону

$$N(t) = \rho_d * S_r \qquad (2.5)$$

2.2 Епідемічні, ендемічні та ерадикаційні процеси

Динаміка інфекційних захворювань є давньою і багатою галуззю математичної біології, яка бере свій початок з піонерських робіт Хеймера в 1906 році і Росса в 1911 році, що призвело до більш сучасних досягнень таких вчених, як Андерсон, Мей, Дітц, Хеткот, Кастілло-Чавес та інших. З роками ця галузь зазнала значного зростання як у масштабах, так і в значущості. Швидке розширення, однак, призвело до розмитості предмету, що свідчить про необхідність консолідації різноманітних математичних підходів в узгодженій структурі [6].

Епідемія характеризується раптовим збільшенням числа випадків конкретного захворювання, що перевищує звичайні очікування на визначеній території або серед

окремої групи населення. Така вибухова динаміка захворюваності часто викликає місцеві або міжнародні заходи реагування, з метою контролю поширення та зменшення впливу. Спалах тяжкого гострого респіраторного синдрому у 2002 році, обмежений певними географічними рамками, виступає класичним прикладом.

Ендемічна хвороба ж зберігає постійну присутність на певній території або серед окремої групи населення. Це захворювання, яке очікується в конкретному місці або серед певної групи людей. Малярія, що є ендемічною для окремих регіонів Африки, служить відповідним прикладом. У цих місцях малярія виявляється регулярним явищем і стабільною проблемою охорони здоров'я.

Нульове появлення нових випадків захворювання завдяки цілеспрямованим діям на глобальному рівні характеризує стан викорінення. Досягнення цього стану вимагає великих зусиль і вдалося лише у небагатьох випадках, наприклад, з віспою. Ліквідація віспи стала історичним досягненням, яке відображає силу громадського здоров'я на глобальному рівні.

Таким чином, будь-яка хвороба в кожний момент може бути в одному із зазначених станів.

2.2.1 Виділення критеріїв оцінки моделей

Для оцінки та порівняння методі виділимо наступні критерії:

а) Характеристики хвороби: Різні хвороби мають різні характеристики, які повинні бути представлені в моделі. Наприклад, деякі хвороби мають значний інкубаційний період. Інші хвороби можуть поширюватися безсимптомними носіями, що також потребує моделі, яка може це відобразити.

б) Масштабованість: Модель повинна бути здатна працювати з великою кількістю осіб. Це означає, що вона повинна бути розроблена таким чином, щоб її можна було

масштабувати зі збільшенням кількості особин. Продуктивність моделі не повинна суттєво погіршуватися зі збільшенням кількості особин.

в) Інтерпретованість: Якщо модель занадто складна, вона може перетворитися на "чорну скриньку", куди надходять вхідні дані, а виходять вихідні, але процес між ними недостатньо зрозумілий. Брак прозорості може ускладнити довіру інших до результатів, особливо якщо вони приймають важливі рішення на основі цих результатів. На противагу цьому, модель, яку можна інтерпретувати, чітко показує, як вхідні дані перетворюються на вихідні, що може зміцнити довіру до прогнозів моделі.

2.2.2 SIR

Модель SIR є однією з найпростіших компартментних моделей в епідеміології. Вона поділяє населення на три групи [6]:

а) Сприйнятливий (Susceptible): особи, які можуть заразитися хворобою

б) Інфікований (Infectious): особи, які заразилися і можуть поширювати хворобу

в) Одужав (Recovered): особи, які одужали від хвороби і тепер мають імунітет.

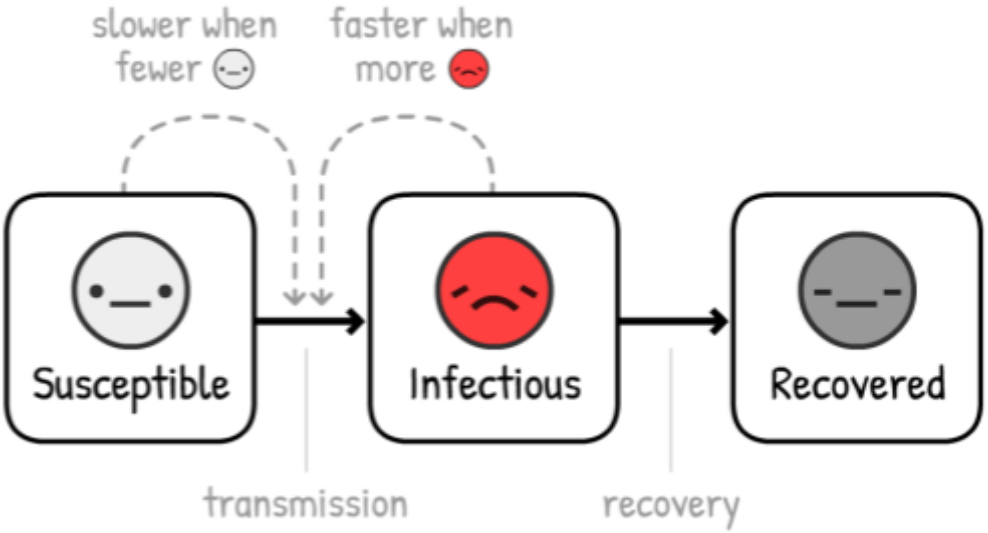

Рисунок 2.4 - Візуалізація станів SIR [7]

Оригінальна модель SIR, в якій Мальтузіанська модель зростання була адаптована Кермаком і МакКендріком, є добре відомою моделлю для моделювання епідемічного зростання з використанням звичайних диференціальних рівнянь, де S, I і R - це сприйнятлива, інфікована і евакуйована популяції, а параметри β і γ - це темпи інфікування і темпи евакуації (темпи одужання). Для кожного моменту часу t рівняння описуються наступним чином [8]:

$$\frac{dS(t)}{dt} = -\ \beta S(t)I(t) \tag{2.6}$$

$$\frac{dI(t)}{dt} = \beta S(t)I(t)\ -\ \gamma I(t) \tag{2.7}$$

$$\frac{dR(t)}{dt} = \gamma I(t) \tag{2.8}$$

Оцінка за критеріями:

а) Характеристики хвороби: Дуже низька. Модель SIR є досить простою і не враховує багатьох характеристик хвороби. Наприклад, вона не включає інкубаційний період, тому не є ідеальною для хвороб зі значним інкубаційним періодом. Вона також не враховує безсимптомних носіїв, оскільки припускає, що всі інфіковані особи однаково заразні.

б) Масштабованість: Дуже висока. SIR-модель добре масштабується залежно від кількості осіб. Оскільки вона використовує диференціальні рівняння для опису швидкості змін між відділеннями, не має значення, чи популяція становить 100 осіб або 1 мільйон осіб. Рівняння все одно будуть точно описувати темпи змін.

в) Інтерпретованість: Дуже висока. Модель SIR є відносно простою і добре інтерпретується. Диференціальні рівняння, що використовуються в моделі, чітко показують, як кількість сприйнятливих, інфікованих та одужалих осіб змінюється з часом.

Таким чином, модель SIR є простою, масштабованою та інтерпретованою моделлю, яка може бути корисною для розуміння базової динаміки поширення хвороб. Однак її простота означає, що вона може неточно відображати хвороби зі складнішими характеристиками, наприклад, зі значним інкубаційним періодом або безсимптомним перебігом.

2.2.3 Модель FRED (Framework for Reconstructing Epidemiological Dynamics)

FRED (Framework for Reconstructing Epidemiological Dynamics), є продуктом Лабораторії динаміки громадського здоров'я Пітта (Pitt Public Health Dynamics Laboratory). Ця система функціонує на основі агентного моделювання [9].

FRED унікально представляє кожну людину в межах певної географічної області як індивідуальну сутність, з характерними особливостями здоров'я,

поведінки, демографічними, соціальними та сімейними рисами. Взаємодія між цими особами реалістично зображена в рамках соціальних мереж, таких як робочі місця, домогосподарства та школи.

Спочатку FRED був розроблений для імітації спалахів інфекційних захворювань, але з часом його можливості були розширені. Тепер користувачі можуть використовувати FRED для моделювання широкого спектру станів здоров'я і досліджувати, як частота цих станів змінюється з часом у конкретному регіоні.

Оцінка за критеріями:

а) Характеристики хвороб: Дуже висока. Модель FRED є дуже гнучкою і може бути адаптована для відображення широкого спектру характеристик захворювань. Вона може враховувати хвороби з інкубаційним періодом, безсимптомним носійством і різним ступенем інфекційності та сприйнятливості серед людей. Це пов'язано з тим, що кожна людина в моделі представлена як агент з власним набором атрибутів, які можуть бути скориговані відповідно до характеристик захворювання, що моделюється.

б) Масштабованість: Дуже низька. Агентні моделі, такі як FRED, можуть стикатися зі значними проблемами масштабування. Хоча теоретично вони можуть працювати з великими популяціями, на практиці необхідність обчислювати поведінку кожного окремого агента може призвести до значних обчислювальних витрат. Зі збільшенням кількості агентів значно зростають і необхідні обчислювальні ресурси (пам'ять та обчислювальна потужність). Це означає, що продуктивність моделі може погіршуватися зі збільшенням кількості індивідів, що потенційно обмежує її застосовність для дуже великих популяцій або складних сценаріїв.

в) Інтерпретованість і складність: Нормальна. Агентні моделі за своєю суттю є більш складними, ніж прості математичні моделі, такі як SIR або SEIR. Це пов'язано з тим, що вони включають більш детальне представлення індивідуальної поведінки та взаємодій. Однак ця складність також робить їх більш реалістичними і потенційно більш точними. З точки зору інтерпретованості, поведінка моделі в цілому випливає

із взаємодії окремих агентів, що може зробити її більш складною для розуміння, ніж моделі, засновані на агрегованих величинах. Однак правила, що регулюють поведінку та взаємодію агентів, як правило, прості та зрозумілі. Тому, хоча загальна поведінка моделі може бути складною, механізми, що лежать в її основі, є прозорими і зрозумілими.

Таким чином, хоча модель FRED є гнучким інструментом для моделювання поширення хвороб і може представляти широкий спектр характеристик захворювань, її масштабованість є суттєвим обмеженням. Необхідність обчислювати поведінку кожного окремого агента може призвести до значних обчислювальних витрат, що обмежує її застосовність для дуже великих популяцій або складних сценаріїв.

2.2.4 Модель SEIR

Модель SEIR є розширенням моделі SIR, що включає додатковий стан:

а) Сприйнятливі (Susceptible): особи, які можуть заразитися хворобою

б) Вразливі (Exposed): особи, які були інфіковані, але ще не є заразними

в) Інфіковані (Infected): особи, які заразилися і можуть поширювати хворобу

г) Одужавший (Recovered): особи, які одужали від хвороби і тепер мають імунітет.

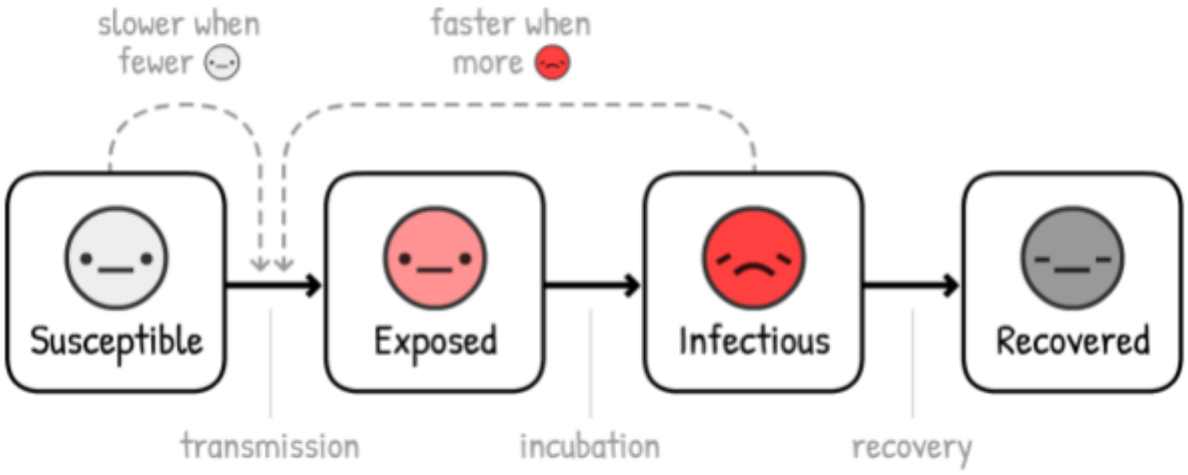

Рисунок 2.4 - Візуалізація станів SEIR [7]

Модель SEIR є розширенням моделі SIR, що включає додатковий стан:

а) Сприйнятливі (Susceptible): особи, які можуть заразитися хворобою

б) Вразливі (Exposed): особи, які були інфіковані, але ще не є заразними

в) Інфіковані (Infected): особи, які заразилися і можуть поширювати хворобу

г) Одужавший (Recovered): особи, які одужали від хвороби і тепер мають імунітет.

Необхідно мати на увазі, що змінні представляють частку осіб - іншими словами, нормалізовані відповідним чином [10]:

$$S + E + I + R = 1 \qquad (2.9)$$

Крім того, припустимо наступне:

а) Коефіцієнти народжуваності та смертності однакові і представлені через μ,

б) Середній латентний період хвороби позначено через $\frac{1}{\alpha}$,

в) Середній інфекційний період позначено через $\frac{1}{\gamma}$,

г) Ті, хто одужав, набувають постійного імунітету,

д) Частота контактів β може змінюватися з часом.

Таким чином можемо записати модель:

$$\frac{dS(t)}{dt} = \mu - \beta(t)SI - \mu S \quad (2.10)$$

$$\frac{dE(t)}{dt} = \beta(t)SI - (\mu + \alpha)E \quad (2.11)$$

$$\frac{dI(t)}{dt} = \alpha E - (\mu + \gamma)I \quad (2.12)$$

Оцінка за критеріями:

а) Характеристики хвороби: Нормальна. Модель SEIR є більш гнучкою, ніж модель SIR, з точки зору характеристик захворювання. Вона включає інкубаційний період. Це робить її придатною для захворювань зі значним інкубаційним періодом. Однак, як і модель SIR, модель SEIR не враховує в явному вигляді безсимптомних носіїв.

б) Масштабованість: Дуже висока. Як і модель SIR, модель SEIR добре масштабується залежно від кількості осіб. Вона використовує диференціальні рівняння для опису швидкості змін між відділеннями, тому не має значення, чи населення становить 100 осіб або 1 мільйон осіб. Рівняння все одно будуть точно описувати темпи змін. Однак вони також припускають добре перемішану популяцію, що може бути неточним для великих популяцій з окремими підгрупами.

в) Інтерпретованість: Висока. Модель SEIR є дещо складнішою, ніж модель SIR, через додавання секції "Exposed". Однак, вона все ще є відносно простою і добре інтерпретується. Диференціальні рівняння, що використовуються в моделі, чітко показують, як кількість сприйнятливих, експонованих, інфікованих та одужалих осіб змінюється з часом.

Таким чином, модель SEIR є дещо складнішою, але все ще добре інтерпретованою моделлю, яка може представляти хвороби з інкубаційним періодом. Вона добре масштабується залежно від кількості осіб, але, як і модель SIR, може

неточно відображати хвороби з більш складними характеристиками, такими як значний інкубаційний період або безсимптомне поширення.

2.2.6 Висновки до розділу

У даному розділі проведено поглиблене дослідження математичних моделей, використовуваних для симуляції поширення хвороб. Основна увага була приділена трьом моделям: SIR, FRED та SEIR, оскільки вони широко застосовуються і актуальні у галузі епідеміології.

Оцінка цих моделей здійснювалась на основі трьох критичних критеріїв. Першим критерієм була здатність моделі точно відтворювати характеристики захворювання. Різні захворювання мають унікальні особливості, такі як інкубаційний період та можливість безсимптомного перенесення, які мають бути вірно відображені в моделі. Цей критерій має вирішальне значення для ефективності моделі в прогнозуванні поширення та впливу хвороби.

Другим критерієм була масштабованість. Ефективна модель повинна бути здатна працювати з великою кількістю людей без значного погіршення продуктивності. Це особливо важливо в контексті масштабних епідемій або пандемій, коли модель повинна враховувати взаємодію мільйонів або навіть мільярдів людей.

Третім критерієм була інтерпретованість. Занадто складна модель може стати "чорною скринькою", у результаті чого іншим стає важко зрозуміти її результати та довіряти їм. З іншого боку, модель, яку можна інтерпретувати, чітко показує, як вхідні дані перетворюються на вихідні, що підвищує довіру до прогнозів моделі та полегшує її використання у процесах прийняття рішень.

Таблиця 2.2 - Порівняльна таблиця моделей симуляції епідемічних, ендемічних та ерадикаційних процесів.

Модель	Характеристики хвороби	Масштабованість	Інтерпретованість
SIR	Дуже низька	Дуже висока	Дуже висока
FRED	Дуже висока	Дуже низька	Нормальна
SEIR	Нормальна	Дуже висока	Висока

Оцінивши моделі, виявлено, що модель SIR, хоча і отримала дуже високі бали за масштабованість та інтерпретованість, не дуже добре відображає характеристики захворювання. Модель FRED, навпаки, чудово відображала характеристики захворювання, але мала обмежену масштабованість, а її інтерпретаційна здатність була визнана середньою. Модель SEIR, натомість, забезпечила збалансовану ефективність за всіма критеріями. Вона була оцінена як нормальна в представленні характеристик захворювання, дуже висока в масштабуванні та висока в інтерпретації.

Після ретельного аналізу та порівняння сильних і слабких сторін кожної моделі дійшов висновку, що модель SEIR є найбільш підходящим вибором для проекту. Збалансовані показники за всіма критеріями оцінки роблять її найбільш універсальною та надійною моделлю для моделювання поширення хвороби.

2.3 Висновки до розділу

У цьому розділі дипломної роботи були встановлені критерії вибору відповідних математичних моделей для моделювання мобільності людей та

поширення хвороб, як невід'ємні компоненти системи моделювання процесів епідемії, ендемії та ерадикації.

Процес відбору моделей розпочався з ретельної оцінки численних математичних моделей з урахуванням таких факторів, як простота, застосовність, емпірична перевірка та обчислювальна ефективність. Були вибрані дві конкретні моделі: Гравітаційна модель зі степеневим законом для моделювання мобільності людей та модель SEIR для моделювання поширення хвороб.

Scaled Power Law Gravity Model була обрана через свою здатність ефективно відображати переміщення людей між різними місцями. Ця модель враховує чисельність населення та відстань між місцями, і вона широко використовується та підтверджена в різних дослідженнях, що підтверджує її придатність для моделювання мобільності людей в контексті дослідження.

З іншого боку, модель SEIR була обрана для моделювання поширення хвороб через її комплексну структуру, яка враховує сприйнятливих, експонованих, інфікованих та одужалих осіб. Здатність моделі відтворювати інкубаційний та інфекційний періоди добре узгоджується з потребою у створенні точних та детальних симуляцій поширення хвороб.

Використовуючи ці дві математичні моделі разом, створюється надійна, точна та ефективна система для моделювання різних процесів захворювання, з урахуванням критичної ролі мобільності людей.

3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Огляд існуючих рішень для розробки симуляції

3.1.1 NetLogo

NetLogo - це динамічне, багатоагентне програмоване середовище моделювання, призначене для моделювання природних і суспільних явищ. Спочатку його створив Урі Віленський у 1999 році, і з тих пір воно постійно вдосконалюється [11].

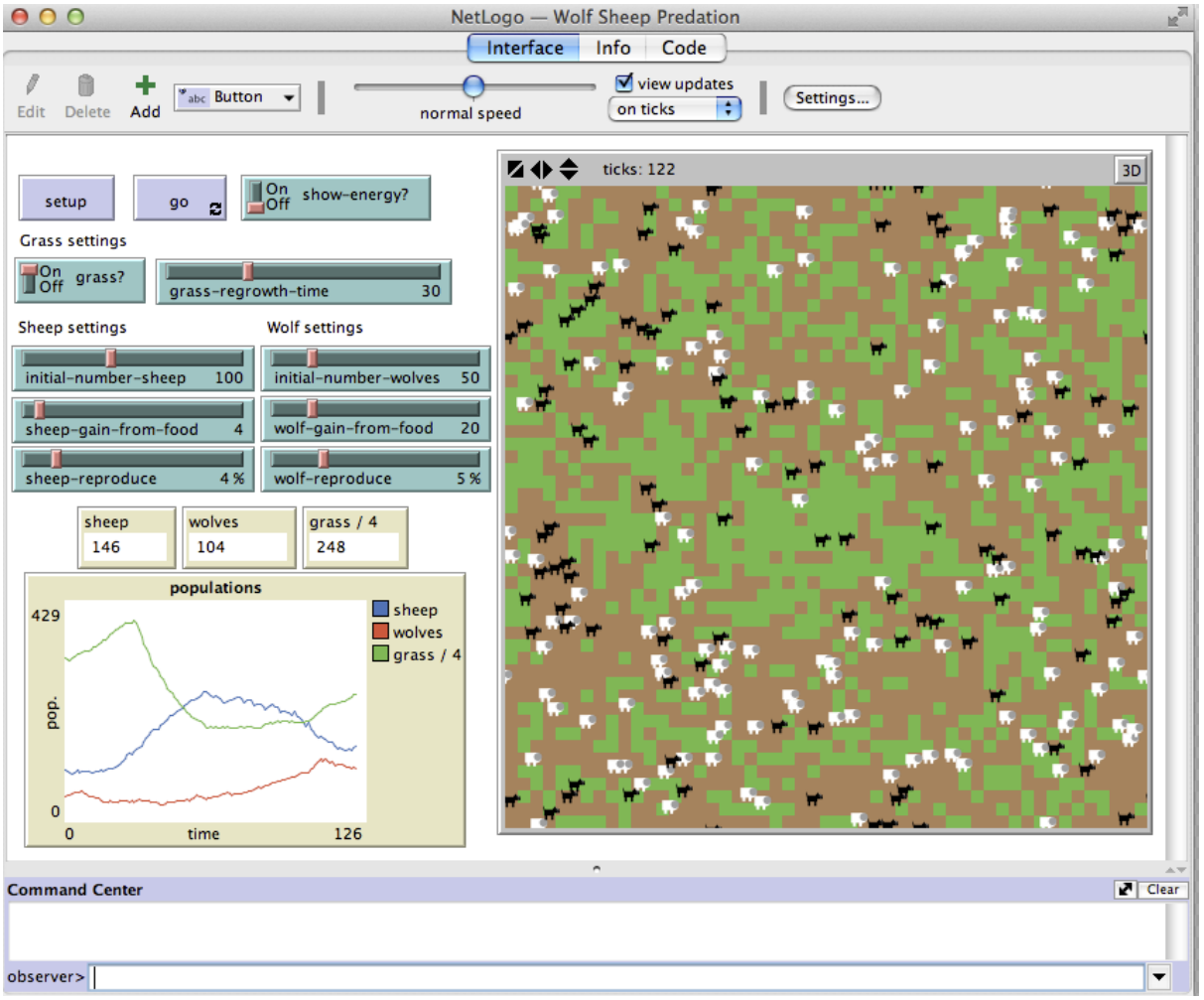

Рисунок 3.1 – Інтерфейс NetLogo [11]

NetLogo особливо ефективний для моделювання складних систем, які еволюціонують у часі. Він дозволяє призначати інструкції численним "агентам", які функціонують автономно. Ця можливість полегшує вивчення взаємозв'язку між індивідуальною поведінкою на мікрорівні та новими моделями макрорівня, що виникають в результаті їхньої взаємодії.

NetLogo надає вичерпну документацію та навчальні ресурси. Він також включає бібліотеку вже існуючих моделей, величезну колекцію попередньо закодованих симуляцій, які користувачі можуть використовувати і модифікувати. Ці симуляції охоплюють широкий спектр дисциплін, від природничих до соціальних наук.

Станом на останнє оновлення у вересні 2021 року, NetLogo підтримує моделювання у форматі 2D та 3D. Він включає власну мову програмування для визначення поведінки агентів та загального середовища. NetLogo є популярним інструментом у дослідницькій, освітній та навчальній сферах.

Переваги:

а) Простота використання: NetLogo розроблений таким чином, щоб бути зручним і доступним навіть для тих, хто не має великого досвіду програмування. Вбудована мова проста і розроблена спеціально для агентного моделювання.

б) Створено для симуляцій: NetLogo спеціально розроблений для багатоагентних симуляцій, що робить його природним інструментом для моделювання складних систем, таких як епідемії.

в) Візуалізація: NetLogo надає надійні інструменти візуалізації, які можуть допомогти в розумінні та представленні динаміки поширення хвороби.

г) Попередні моделі: Бібліотека вже існуючих моделей NetLogo може слугувати корисною відправною точкою, потенційно заощаджуючи час та зусилля.

Недоліки:

а) Масштабованість: NetLogo може мати проблеми з дуже великими симуляціями. Якщо модель включає мільйони агентів або складне просторове середовище, ви можете зіткнутися з проблемами продуктивності.

б) Гнучкість та контроль: Простота NetLogo є сильною стороною, але йому може не вистачати розширених можливостей більш складних інструментів моделювання або мов програмування загального призначення.

в) Мова програмування: NetLogo використовує пропрієтарну мова програмування, що може стати проблемою для інтеграції з іншими системами, а також спільнота яка працює з NetLogo набагато менше ніж з іншими мовами програмування.

Отже, хоча NetLogo пропонує зручне середовище і добре підходить для моделювання складних систем, він не може бути оптимальним вибором для симуляції.

По-перше, NetLogo має погану масштабованість, що може стати проблемою, так як наша симуляція включає велику кількість агентів та складне середовище.

По-друге, NetLogo пропонує обмежену гнучкість і контроль порівняно з іншими платформами, що може обмежити нашу здатність налаштовувати модель під наші конкретні потреби.

Нарешті, власна мова програмування NetLogo, хоча і розроблена таким чином, щоб бути доступною, не є широко використовуваною за межами середовища NetLogo. Це може обмежити здатність інтегрувати нашу модель з іншими системами, використовувати бібліотеки або інструменти, доступні для інших мов, а також обмежить майбутнє розширення системи іншими розробниками або дослідниками.

3.1.2 EpiModel

EpiModel - це програмний пакет, розроблений для використання з мовою програмування R. Його основна функція - полегшити математичне моделювання інфекційних захворювань у структурованих популяціях. Він може враховувати різноманітні епідеміологічні ситуації завдяки використанню детермінованих компартментних моделей, стохастичних моделей індивідуальних контактів і стохастичних мережевих моделей [12].

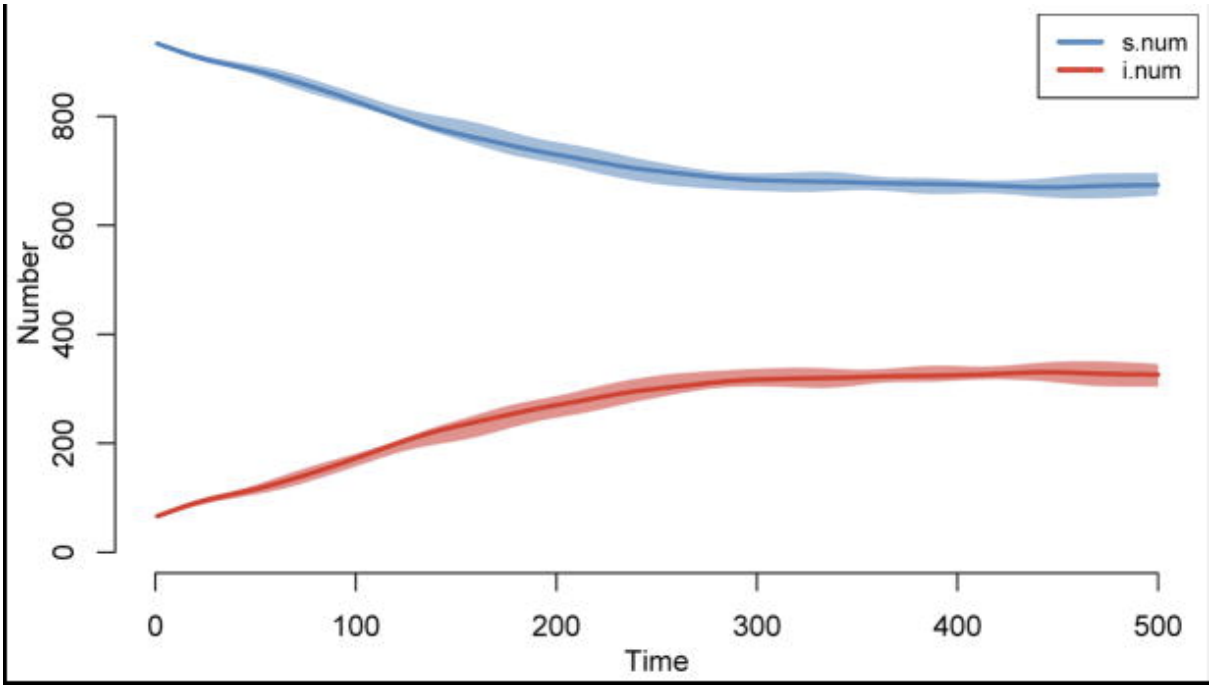

Рисунок 3.2 – Інтерфейс EpiModel [12]

EpiModel використовує можливості пакетів "network" і "tergm" для моделювання та аналізу динамічних мереж в контексті передачі захворювань. Крім того, він використовує модульну структуру для визначення, розв'язання та аналізу епідемічних моделей. Ця модульна структура може бути розширена користувачами для створення власних модулів для нових епідеміологічних процесів. Сфера

застосування програми широка, включаючи можливість моделювання складних соціальних процесів і структур, які мають значний вплив на передачу захворювань, таких як мережі сексуальних контактів або вживання наркотиків.

Переваги:

а) Надійний та перевірений: EpiModel широко використовується і перевірений епідеміологами, що може підвищити довіру до результатів вашого моделювання.

б) Простота у використанні: надає простий у використанні інтерфейс для визначення моделей захворювань, що може бути особливо корисним, якщо ви не є фахівцем у галузі інфекційних захворювань.

в) Розширений аналіз: Він оснащений багатьма функціями для аналізу та візуалізації епідемічних процесів.

Недоліки:

а) Обмежена інтерактивність: Хоча EpiModel чудово підходить для статистичного моделювання, він не пристосований для створення інтерактивних, захоплюючих симуляцій.

б) Обмеженість у користувацьких візуалізаціях: Візуалізація за допомогою R є потужною, але може не забезпечити необхідний рівень гнучкості та інтерактивності.

в) Обмежене моделювання мобільності: Хоча EpiModel може моделювати мобільність до певної міри, він може бути менш придатним для детального моделювання внутрішньоміських моделей людської мобільності порівняно зі спеціально розробленим симулятором.

Незважаючи на те, що EpiModel є надійним і широко визнаним інструментом для епідеміологічного моделювання, він може бути непридатним для певних застосувань, особливо тих, що вимагають детального моделювання мобільності, розширеної інтерактивності та розширеної користувацької візуалізації.

По-перше, EpiModel має обмеження щодо інтерактивних функцій, які особливо актуальні, коли для моделювання необхідні зміни або динамічне коригування параметрів моделі в реальному часі, керовані користувачем. Це

обмежує його використання в сценаріях, де бажаним є інтерактивне моделювання з ефектом занурення.

По-друге, хоча EpiModel є потужним інструментом для аналізу і має базові можливості візуалізації, він не підходить для створення індивідуальних, комплексних візуальних представлень. Якщо ваш проект вимагає детальних або унікальних елементів візуалізації, цей інструмент може не виправдати очікувань.

Нарешті, EpiModel може бути не найкращим вибором для детального моделювання внутрішньоміських моделей людської мобільності. Він може впоратися з певним рівнем моделювання мобільності, але для складних моделей міської мобільності його можливості можуть виявитися недостатніми.

3.1.3 Unity

Unity - це кросплатформний ігровий рушій, розроблений компанією Unity Technologies. Unity є однією з найпопулярніших платформ як для незалежних (інді) розробників ігор, так і для великих студій.

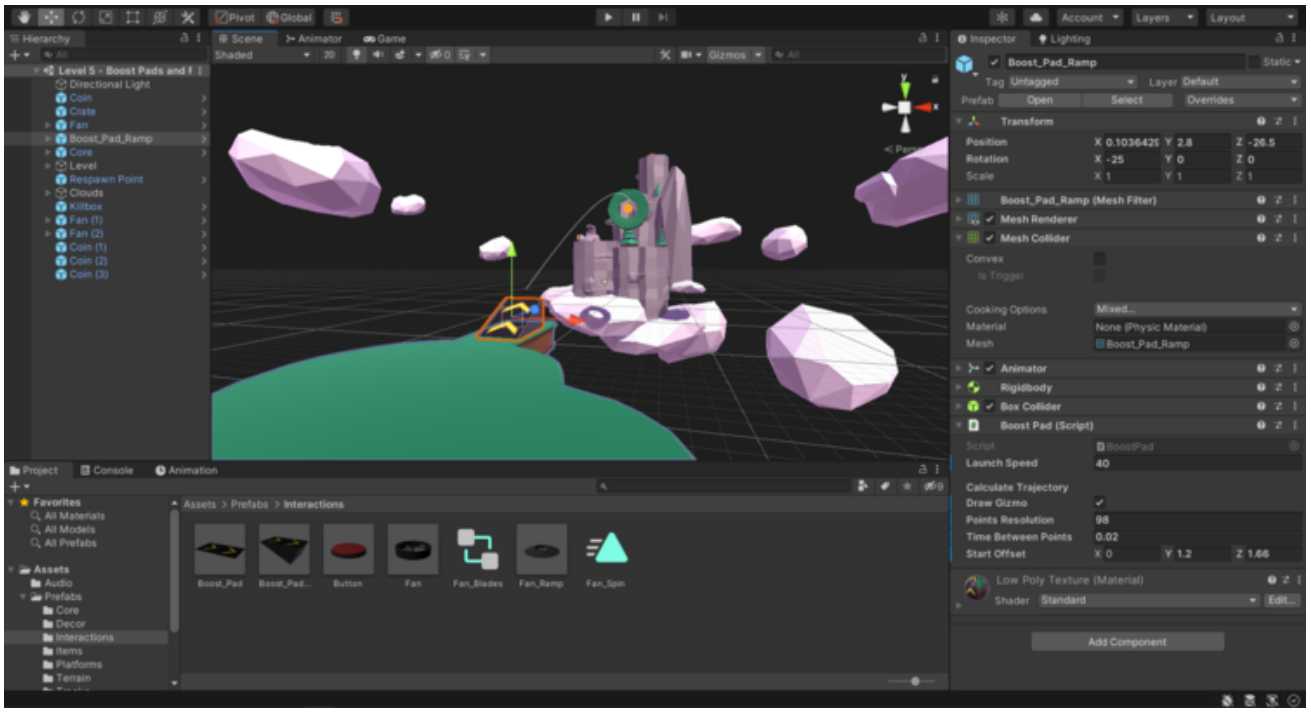

Рисунок 3.3 – Інтерфейс Unity

Переваги:

а) Захоплюючі візуалізації: Основною перевагою Unity є його здатність створювати візуально привабливі, інтерактивні 3D-середовища. Це може бути особливо корисно для ілюстрації складних явищ, таких як поширення хвороб у міських умовах.

б) Кастомізація та гнучкість: Unity пропонує широку гнучкість і може бути дуже добре налаштований. Це може бути важливим при моделюванні складних систем, таких як внутрішньоміська мобільність людей.

в) Інтерактивність: Ігрова платформа Unity добре підходить для розробки симуляцій, які дозволяють користувачам взаємодіяти з моделлю в режимі реального часу, що може допомогти в розумінні складних епідеміологічних процесів.

г) Широкий вибір платформ: Unity може створювати додатки для багатьох платформ (Windows, MacOS, Linux, iOS, Android, Web, VR/AR), розширюючи сферу застосування вашої моделі.

Недоліки:

а) Час розробки: Моделювання складних систем в Unity може зайняти багато часу, особливо якщо розробники ще не знайомі з інструментом.

б) Відсутність вбудованих епідеміологічних інструментів: На відміну від EpiModel та NetLogo, Unity не містить інструментів епідеміологічного або агентного моделювання, а це означає, що вам доведеться реалізовувати ці функції самостійно або інтегрувати з іншим інструментом.

3.1.4 Висновки до розділу

У даному розділі проведено поглиблений порівняльний аналіз трьох відомих програмних рішень: Unity, NetLogo та EpiModel. Вибір платформи базувався на вимогах проекту та очікуваних результатів.

Unity виявилася найбільш підходящою платформою, перевершуючи NetLogo та EpiModel завдяки своїм розширеним можливостям та перевагам. Інтегровані імерсивні візуалізації забезпечують високий рівень деталізації та покращують користувацький досвід. Кастомізація Unity дозволяє адаптувати симуляції відповідно до унікальних вимог проекту. Гнучкість системи дозволяє враховувати різноманітні сценарії та внести необхідні модифікації.

3.2 Розробка алгоритму симуляції

Розробимо алгоритм який поєднає принципи моделі SEIR та Scaled Power Law Gravity Model.

Перший крок алгоритму передбачає поділ області моделювання на окремі регіони, що гарантує точне відображення унікальної сприйнятливості та рівнів експозиції, притаманних кожному регіону. Такий деталізований підхід дозволяє всебічно відстежувати розвиток хвороби, розуміючи, що мікросвіт кожного окремого регіону робить свій внесок у загальну картину епідемії.

Після цього розраховуються специфічні для регіону характеристики на основі часових змінних в рамках симуляції. На цьому етапі враховуються постійно мінливі фактори, такі як коливання щільності населення, коливання рівня імунітету і зміни в регіональних зв'язках, що дозволяє більш динамічно і детально моделювати поширення хвороби.

На наступному етапі для розрахунку гравітаційної взаємодії між кожним регіоном застосовується степеневий закон гравітації, який враховує специфічні для кожного регіону характеристики. Це призводить до формування матриці "Перехід-Призначення", яка діє як кількісний показник потенційного переміщення людей між регіонами, імітуючи реальний сценарій переміщення людей між різними регіонами.

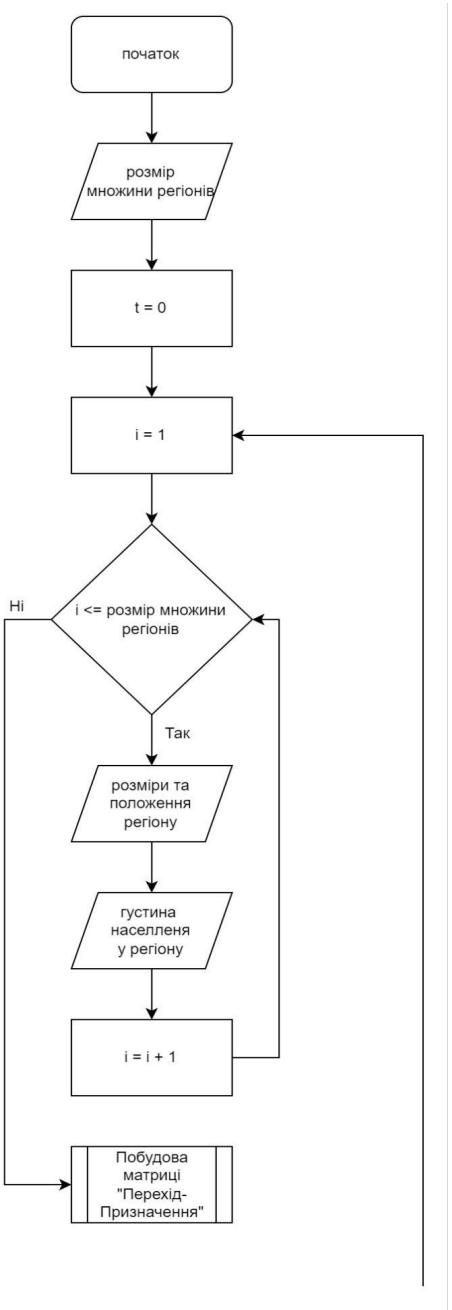

Рисунок 3.4 – Схема алгоритму формування матриці “Перехід-Призначення”

Згодом, на основі матриці переходів і пунктів призначення моделюються переміщення агентів (які представляють людей). Однак складність і масштаб симуляції виключають можливість запуску симуляції окремо для кожного агента через обмеження обчислювальної продуктивності, які можуть виникнути в результаті. Щоб обійти цю проблему, алгоритм був розроблений таким чином, щоб

запускати симуляції для всіх агентів одночасно, оптимізуючи обчислювальну ефективність без шкоди для точності та достовірності моделі.

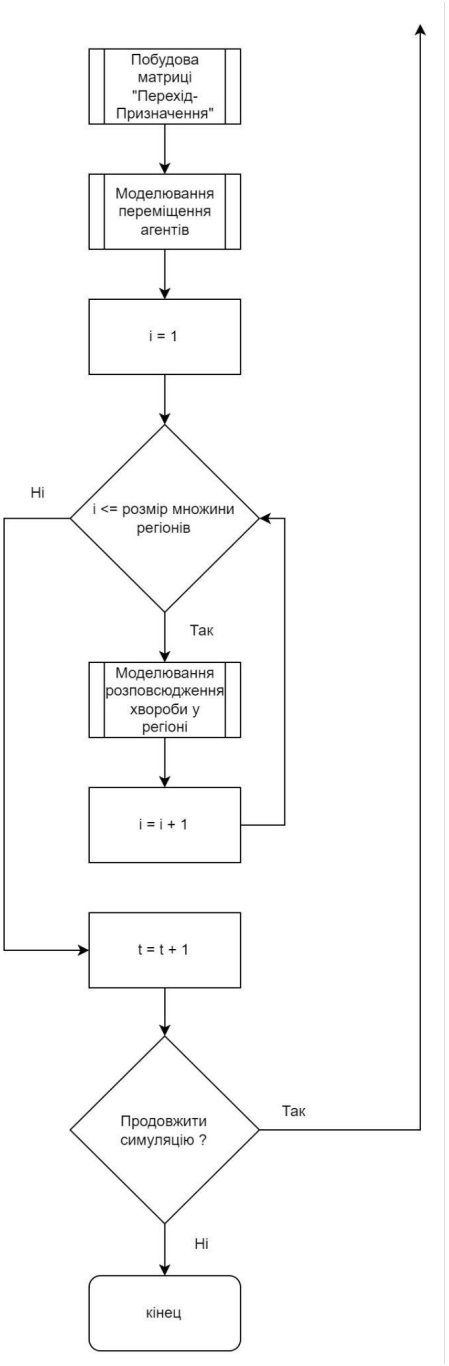

Рисунок 3.5 – Схема алгоритму симуляції

Нарешті, модель SEIR використовується в алгоритмі для розрахунку поширення хвороби для кожного окремого регіону, враховуючи специфічні для нього характеристики. Це дозволяє безперервно відстежувати розвиток епідемії в кожному регіоні в різні моменти часу, охоплюючи всі стадії хвороби - сприйнятливість, контакт, інфікування та одужання.

3.3 Розробка архітектури

Вибір найбільш підходящої програмної архітектури для системи є критично важливим завданням, що впливає на ефективність, розширюваність і ремонтопридатність системи в цілому. Є три потенційні архітектури: MVC (Model-View-Controller), MVVM (Model-View-ViewModel) та OOD (Object-Oriented Design), кожна з яких має свої унікальні переваги та можливості застосування.

Патерн MVC широко використовується в програмній інженерії, переважно в додатках з користувацьким інтерфейсом. Його структура поділяє додаток на три взаємопов'язані компоненти: модель, яка втілює основні дані та пов'язану з ними бізнес-логіку, представлення, яке відповідає за відображення даних і будь-якого результату, з яким взаємодіє користувач, і контролер, який керує зв'язком між моделлю і представленням. Такий чіткий розподіл завдань є корисним для розробки додатків, де один аспект може змінюватися незалежно від інших, пропонуючи хороший ступінь гнучкості.

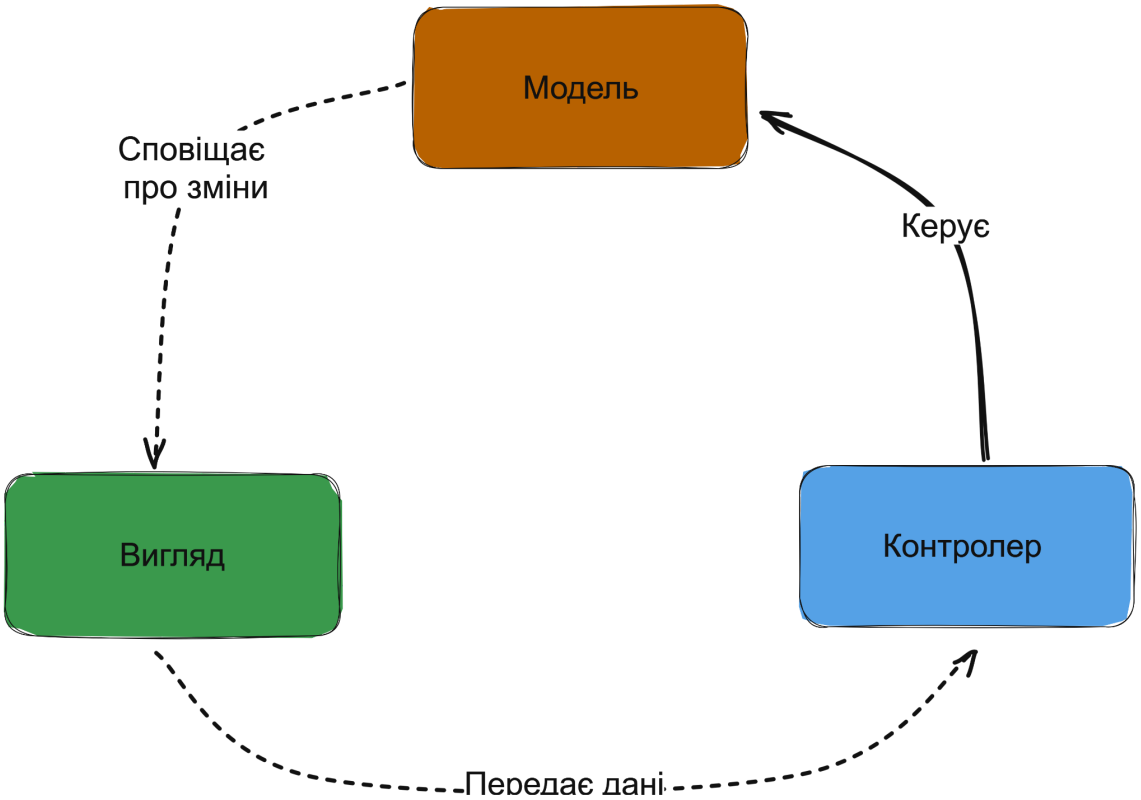

Рисунок 3.6 – MVC архітектура

Тим часом, патерн MVVM є розширенням дизайну MVC і забезпечує додатковий рівень абстракції через ViewModel. ViewModel виступає посередником між моделлю та представленням, допомагаючи розділити логіку представлення та користувацький інтерфейс. Це може виявитися надзвичайно корисним у більш складних додатках, особливо там, де користувацький інтерфейс повинен представляти велику кількість даних або де взаємодія з користувачем є складною. MVVM підтримує двостороннє зв'язування даних між View і ViewModel, що зменшує кількість шаблонного коду і ще більше полегшує тестування і супровід коду.

Рисунок 3.7 – MVVM архітектура

У контексті системи, призначеної для моделювання складних реальних явищ, таких як епідемії, ендемічні стани та процеси ліквідації, архітектура OOD є найбільш підходящою. OOD надає основу для представлення цих багатогранних компонентів у спосіб, що відображає реальні об'єкти та їхню взаємодію. Дані та поведінка визначаються в термінах об'єктів, що дозволяє інкапсулювати відповідні деталі (дані) та дії над цими даними (методи) в межах об'єктів. Наприклад, об'єкт "людина" може мати атрибути, такі як вік, стан здоров'я, місцезнаходження тощо, і поведінку, таку як пересування та зміна інфекційного статусу.

Додатково, OOD дозволяє успадковування, коли нові об'єкти можуть бути створені на основі існуючих, успадковуючи їхні атрибути та поведінку. Це дозволяє створювати ієрархічні класифікації об'єктів, що сприяє повторному використанню коду і робить систему більш керованою. Поліморфізм, ще один принцип OOD, дозволяє обробляти об'єкти по-різному залежно від їхнього типу або класу даних, що надає єдиний інтерфейс для управління різними типами об'єктів, що може бути безцінним при моделюванні різноманітних сценаріїв епідемій. Принцип абстракції в OOD дозволяє приховати складні деталі реалізації, забезпечуючи при цьому прості інтерфейси для взаємодії, що сприяє управлінню складністю системи.

Враховуючи ці фактори, можна зробити висновок, що OOD забезпечує найкращу архітектурну основу для нашої системи, яка ефективно підтримуватиме моделювання процесів епідемії, ендемії та ередикації. Це підвищить інтуїтивність системи, полегшить управління складними взаємодіями та забезпечить кращу ремонтопридатність і розширюваність у довгостроковій перспективі.

3.4 Діаграма компонентів

Система, призначена для моделювання процесів епідемії, ендемії та ерадикації, складається з п'яти основних компонентів: Моделювання мобільності, Моделювання поширення хвороб, Моделювання агентів, Регіональний компонент та Компонент інтерфейсу користувача.

Компонент "Моделювання мобільності" відіграє важливу роль у створенні матриці "Перехід - Пункт призначення" між різними географічними регіонами. Ця матриця є основою для моделювання переміщення агентів (осіб) між цими регіонами.

Компонент "Моделювання розповсюдження хвороб" реалізує відому епідеміологічну модель SEIR для кожного регіону. Цей компонент враховує такі фактори, як поточний рівень інфікування, сприйнятливість і рівень одужання, що полегшує розрахунок і прогнозування поширення хвороб у кожному регіоні з часом.

Результати компонентів "Моделювання мобільності" та "Моделювання поширення хвороб" об'єднані в Регіональному компоненті. Як центральний вузол, Регіональний компонент інтегрує дані про мобільність і поширення хвороб, щоб забезпечити комплексне уявлення про ситуацію в кожному регіоні.

Компонент "Моделювання агентів" взаємодіє з компонентами "Моделювання мобільності" і "Моделювання поширення хвороб". Він використовує матрицю "Перехід-Призначення", надану компонентом "Моделювання мобільності", для визначення переміщення кожного агента в просторі моделювання. Водночас, він використовує дані про поширення хвороби з компонента "Моделювання мобільності" для визначення інфекційного статусу кожного агента.

Нарешті, компонент інтерфейсу користувача безпосередньо взаємодіє з компонентом регіону. Він використовує інтегровані дані з Регіонального компонента

для візуалізації поширення хвороби і мобільності людей в межах кожного регіону. Компонент інтерфейсу користувача може пропонувати графічні зображення, таблиці даних, оновлення в режимі реального часу та інші корисні функції для забезпечення інтуїтивно зрозумілого та інформативного користувацького досвіду.

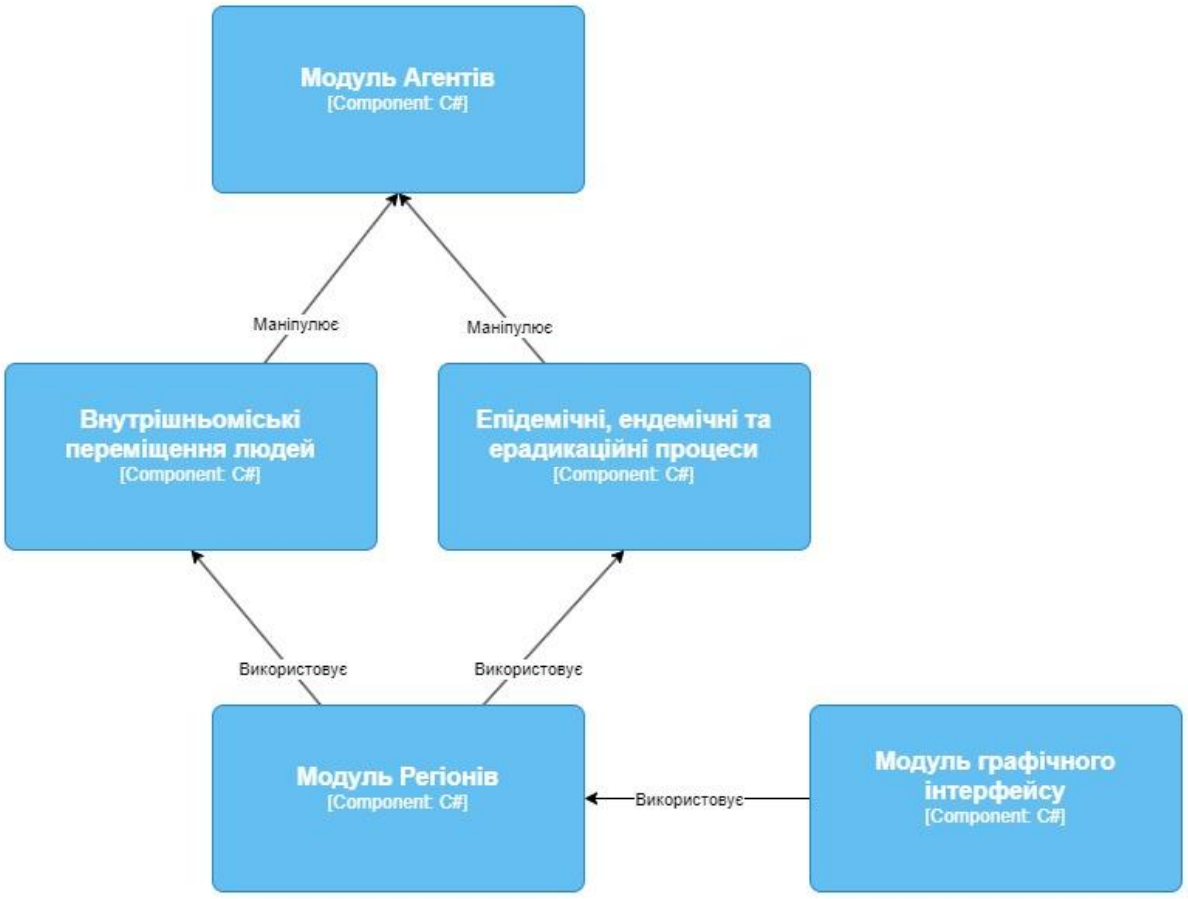

Рисунок 3.8 – Модульна діаграма системи

Ця діаграма компонентів ефективно інкапсулює структуру системи, деталізуючи компоненти, їхні ролі та взаємозв'язки. Така архітектура дозволяє моделювати складні епідеміологічні процеси з урахуванням факторів людської мобільності та представляти результати в доступній і зручній для користувача формі.

3.5 Розробка програмного забезпечення

3.5.1 Модуль “Внутрішньоміські переміщення”.

Модуль моделювання людської мобільності, один з ключових компонентів програмного забезпечення, використовує Scaled Power Law Gravity Model для відтворення та прогнозування моделей людського руху. Модель грунтується на принципі, що рух людини одночасно притягується силою тяжіння регіону і відштовхується від відстані до нього.

Користувач може розділити простір симуляції на різні регіони, кожен з яких має параметри, такі як щільність населення та матриця пріоритетності, залежну від часу доби. Ці елементи використовуються для розрахунку гравітації регіону, що відображає його привабливість для переселенців. Гравітація регіону залежить від його розміру, пріоритетності та щільності населення.

Після розрахунку гравітації для кожного регіону створюється матриця "Перехід-Призначення". Ця матриця відображає ймовірності переміщення між регіонами. Кожний рядок і стовпець матриці представляють регіон, а точка перетину рядка і стовпця відображає ймовірність переходу агента з одного регіону в інший. Ця ймовірність визначається гравітацією відповідних регіонів.

Матриця "Перехід-Призначення" дозволяє симуляції прогнозувати моделі переміщення між регіонами, що є важливим для розуміння поширення хвороб в конкретному середовищі. Ця методологія, заснована на моделі гравітації масштабованого степеневого закону, надає реалістичне зображення людської

мобільності і створює надійну основу для наступних модулів: Моделювання агентів, Моделювання регіонів та Інтерфейс користувача.

3.5.2 Модуль “Епідемічні, ендемічні та ерадикаційних процесів”.

Під час симуляції розповсюдження хвороби необхідно визначити змінні, які специфічні для кожного регіону. Кожен регіон має свої унікальні характеристики, такі як щільність населення та час доби, які впливають на частоту контактів та рівень народжуваності та смертності. У густонаселених регіонах або в години пік очікується вищий рівень контактів, що сприяє поширенню хвороби.

Середній інфекційний період і латентний період вважаються константами для всіх регіонів. Інфекційний період визначає тривалість, протягом якої інфікована особа може передавати хворобу, а латентний період вказує на час від моменту інфікування до моменту, коли особа стає заразною.

Після розрахунку цих змінних для кожного регіону, розраховується кількість агентів у кожній категорії SEIR. Ці числа вказують на поточний стан поширення хвороби в регіоні.

За допомогою зібраних даних - специфічних для регіону змінних, констант та поточного стану поширення хвороби - розв'язуються диференціальні рівняння SEIR. Це дозволяє визначити нову кількість сприйнятливих, експонованих, інфікованих та одужалих агентів у регіоні. Це оновлення враховує всі змінні та константи і є основою для наступної ітерації симуляції.

Такий підхід з використанням моделі SEIR та специфічних для регіону змінних дозволяє точно відтворити динаміку поширення хвороби. Він забезпечує узгодженість прогнозів моделі з характеристиками населення та поведінкою людей, що підвищує надійність симуляції хвороби. Він гарантує, що прогнози моделі

залишаються узгодженими з нюансами розподілу населення та поведінки людей, тим самим сприяючи її надійності.

3.5.3 Модуль “Регіонів”.

Модуль "Регіонів" виконує важливу роль у нашій симуляції, являючись основним компонентом, що містить критичну інформацію для моделювання поширення хвороб і людської мобільності. Цей модуль дозволяє розділити середовище моделювання на окремі регіони та налаштувати локальні умови, що необхідні для точного відтворення реальних сценаріїв.

Перед запуском симуляції, користувач самостійно створює, розташовує та налаштовує кожен регіон. Це дає можливість користувачу налаштувати симуляцію відповідно до конкретних умов або сценаріїв. Такий підхід забезпечує гнучкість та адаптивність, оскільки різні сценарії можуть вимагати різних регіональних налаштувань.

У компоненті "Регіон" зберігаються різні дані, включаючи параметри, такі як щільність населення та матриця пріоритетності регіонів, яка змінюється залежно від часу доби. Ці параметри впливають на моделювання поширення хвороби та людської мобільності, визначаючи частоту контактів та ймовірність переміщення.

Компонент "Регіон" є основою системи, забезпечуючи важливе зв'язку між моделюванням людської мобільності та поширенням хвороб. Його можливість визначати параметри, специфічні для кожного регіону, додає реалістичності та актуальності симуляції, роблячи її придатною для різноманітних сценаріїв.

3.5.3 Модуль “Агентів”.

Модуль "Агенти" виконує значну функцію в нашій системі, діючи як медіатор для передачі даних між регіонами згідно з моделями, що передбачені для симуляції людської мобільності. Цей модуль представляє окремі об'єкти в середовищі моделювання, які піддаються впливу хвороби і відповідають за її поширення.

В контексті симуляції поширення хвороб, агенти несуть дані, що відображають їхній стан, який може бути сприйнятливим, експонованим, інфікованим або одужавшим (SEIR) - ці категорії широко використовуються в епідеміології моделі SEIR. Ці дані мають вирішальне значення, оскільки дозволяють системі відстежувати поширення хвороби в регіоні та між регіонами.

Але агенти не функціонують автономно в цьому процесі. Їх переміщення між регіонами визначається модулем моделювання людської мобільності, який розраховує ймовірності переходів між регіонами. Це включає динаміку людської мобільності в модель поширення хвороб.

Після призначення агенту нового регіону згідно з матрицею "Перехід-Призначення", він переходить в новий регіон разом зі своїми даними про стан хвороби, і це ефективно імітує процес переміщення людини з хворобою з одного місця в інше. Цей обмін даними через переміщення агентів є ключовим механізмом поширення хвороби в нашому симуляційному середовищі.

Отже, компонент агентів забезпечує важливий зв'язок між симуляцією людської мобільності та симуляцією розповсюдження хвороби, успішно усуваючи розрив між цими двома ключовими аспектами системи. Інтегруючи схеми переміщення людей і статусу захворювання з моделлю SEIR, він підвищує реалістичність і точність системи при моделюванні.

3.6 Висновки до розділу

В даному розділі презентується розробка програмної системи для моделювання процесів епідемій, ендемій та ліквідації, заснована на моделі мобільності людей. Для реалізації системи було обрано платформу Unity, яка надає розширені можливості візуалізації, кастомізації та гнучкості.

Основою системи є алгоритм, який поєднує принципи моделі SEIR і Scaled Power Law Gravity Model. Симуляція розпочинається з поділу області моделювання на регіони з їх унікальними характеристиками сприйнятливості та рівнем впливу. Під час моделювання обчислюються змінні, що залежать від часу, специфічні для кожного регіону, і застосовується степеневий закон гравітації. Це дозволяє сформувати матрицю "Перехід-Призначення", яка відображає потенційне переміщення людей між регіонами. Потім агенти переміщуються одночасно для забезпечення ефективності обчислень, а модель SEIR відстежує поширення хвороби в межах кожного регіону.

Для архітектури програмного забезпечення було використано об'єктно-орієнтоване проектування (OOD), що дозволяє ефективно інкапсулювати складні явища реального світу. OOD дозволяє представляти дані та поведінку у вигляді об'єктів (наприклад, людей з властивостями, такими як вік та стан здоров'я) і використовувати успадкування, поліморфізм та абстракцію для ефективного кодування та управління складністю системи.

Остаточно, програма розділена на кілька модулів: Моделювання поширення хвороб, Моделювання мобільності людей, Інтерфейс користувача, Агенти та Регіони. Ці модулі співпрацюють, щоб забезпечити комплексне, точне і динамічне моделювання процесів епідемій, ендемій та ліквідації в людській популяції.

4 ВЕРИФІКАЦІЯ ТА ВАЛІДАЦІЯ

4.1 Верифікація та валідація симуляції людської мобільності

4.1.1 Попередні налаштування тестування

У межах процесу валідації, план тестування включає ручне тестування трьох областей, які будуть ідентифіковані як точки A, B, C і D. Ці точки розташовані на осі ZX з такими координатами: A(-15;0), B(15; 0), C(15;30) і D(-15;30). Для розрахунку гравітаційного притягання між цими точками буде використовуватися модель масштабованого степеневого закону.

4.1.2 Розрахунок данних за моделлю масштабованого степеневого закону

За допомогою моделі обчислюється сила тяжіння між кожною парою регіонів, і ці результати можна побачити в Таблиці 4.1. В цій таблиці різні регіони представлені в рядках і стовпцях, а перетини вказують на обчислену силу тяжіння між відповідними регіонами.

Під час розрахунків використовуємо наступні значення констант:

- $\rho(t) = 1$
- $\alpha(t) = 1$
- $\omega(t) = 1$

Також припустимо, що в кожному регіоні присутня густина населення 0.25 людей на 1 м2, і площа регіону становить 100 м2. З цього можна зробити висновок, що розмір популяції в регіоні становить 25 осіб.

Таблиця 4.1 - Гравітаційна взаємодія на основі моделі масштабованого степеневого закону

	A	B	C	D
A	25	0,81	0,58	0,81
B	0,81	25	0,81	0,58
C	0,58	0,81	25	0,81
D	0,81	0,58	0,81	25

Після виконання розрахунків, використовується інструмент моделювання для відтворення аналогічної гравітаційної динаміки. Вводячи ті самі параметри, прагне до порівняння результатів із початковими розрахунками. Результати моделювання, отримані в таблиці 4.2.

Таблиця 4.2 - Симульована гравітаційна взаємодія на основі моделі масштабованого степеневого закону

	A	B	C	D
A	25	0,81	0,58	0,81
B	0,81	25	0,81	0,58
C	0,58	0,81	25	0,81
D	0,81	0,58	0,81	25

Після здійснення порівняльного аналізу між даними, виведеними за моделлю масштабованого степеневого закону, та результатами моделювання, висновок свідчить про ефективність та надійність симуляції, яка працює відповідно до очікувань.

4.2 Висновки до розділу

У даному розділі розглядались контрольні приклади, на основі яких перевірялась робота симуляції. Описаний процес забезпечив детальніше оцінювання функціональності та точності моделювання. Систематичне проведення симуляції з використанням різних контрольних прикладів підтвердило належну роботу моделі.

За результатами спостерігалось відповідність дії симуляції заданим контрольним прикладам. Така спостереженість підтвердила правильність роботи застосованих алгоритмів і методів в процесі моделювання та відтворення очікуваних результатів.

ВИСНОВКИ

а) Для оцінки моделей людської мобільності були розроблені критерії, які включають простоту, застосовність, емпіричну перевірку та обчислювальну ефективність. Проведено детальний аналіз та порівняння численних математичних моделей, що відображають переміщення людей між різними локаціями. В результаті вибрано Гравітаційну модель зі степеневим законом, яка найкраще відповідала критеріям та потребам дипломної роботи. Адаптовано цю модель для контексту, враховуючи чисельність населення та відстань між населеними пунктами. Для оцінки моделей поширення хвороби розроблені критерії, які включають комплексність структури, здатність моделювати інкубаційний та інфекційний періоди, а також точність і деталізацію симуляцій. Проведено аналіз та порівняння різних моделей поширення хвороб. В результаті вибрано модель SEIR, яка найкраще відповідала критеріям.

б) Після ретельного аналізу різних платформ, для розробки програмної системи моделювання процесів епідемії, ендемії та ерадикації, вирішено використовувати Unity через його розширені можливості візуалізації, кастомізації та гнучкості. Наступним кроком було розроблення алгоритму, який інтегрує принципи моделі SEIR та Scaled Power Law. Цей алгоритм розроблено з урахуванням унікальних характеристик кожного регіону, а також з урахуванням потенційного міжрегіонального переміщення людей. Для реалізації цього алгоритму використано об'єктно-орієнтоване проектування, яке дозволяє ефективно інкапсулювати складні явища реального світу. Завдяки ООД, можна представляти дані та поведінку в термінах об'єктів, що спрощує управління складністю системи. Нарешті, програму розділено на кілька модулів: Моделювання поширення хвороб, Моделювання мобільності людини, Інтерфейс користувача, Агенти та Регіони. Ці модулі працюють

разом, щоб забезпечити комплексне, точне і динамічне моделювання процесів епідемії, ендемії та ерадикації в людській популяції.

ПЕРЕЛІК ПОСИЛАНЬ

1. Human mobility: models and applications / Hugo Barbosa [та ін.] // Physics reports. – 2018. – Т. 734. – С. 1–74.
2. Pe. Aspects and applications of the random walk. / Pe, G. H. Weiss // Journal of the american statistical association. – 1995. – Т. 90, № 429. – С. 398.
3. Imdad Ullah M. Random walk model | time series analysis | basic statistics [Електронний ресурс] / Muhammad Imdad Ullah // Basic Statistics and Data Analysis - Statistics Lecture Notes, Online MCQs. – Режим доступу: https://itfeature.com/time-series-analysis-and-forecasting/random-walk-model (дата звернення: 15.06.2023). – Назва з екрана.
4. On the Levy-Walk Nature of Human Mobility / I. Rhee [та ін.] // 27th IEEE International Conference on Computer Communications (INFOCOM 2008), Phoenix, AZ, 13–18 квіт. 2008 р. – [Б. м.], 2008.
5. Giles J. R. A curated list of mobility models [Електронний ресурс] / John R. Giles, Amy Wesolowski. – Режим доступу: https://covid-19-mobility-data-network.github.io/mobility/articles/V5_list_models.html (дата звернення: 15.06.2023). – Назва з екрана.
6. Capasso V. Mathematical structures of epidemic systems / Vincenzo Capasso. – [Б. м.] : Springer London, Limited, 2008. – 283 с.
7. Salathé M. What Happens Next? COVID-19 Futures, Explained With Playable Simulations [Електронний ресурс] / Marcel Salathé, Nicky Case // What Happens Next? COVID-19 Futures, Explained With Playable Simulations. – Режим доступу: https://ncase.me/covid-19/ (дата звернення: 15.06.2023). – Назва з екрана.

8. Maki Y. Infectious disease spread analysis using stochastic differential equations for SIR model / Yoshihiro Maki, Hideo Hirose // International conference on intelligent systems, modelling and simulation. – 2013. – Т. 4. – e6498254.
9. FRED (A Framework for Reconstructing Epidemic Dynamics): an open-source software system for modeling infectious diseases and control strategies using census-based populations / John J. Grefenstette [та ін.] // BMC public health. – 2013. – Т. 13, № 1.
10. Cai M. Fractional SEIR model and data-driven predictions of COVID-19 dynamics of Omicron variant / Min Cai, George Em Karniadakis, Changpin Li // Chaos: an interdisciplinary journal of nonlinear science. – 2022. – Т. 32, № 7. – С. 071101.
11. NetLogo home page [Електронний ресурс] // The CCL. – Режим доступу: https://ccl.northwestern.edu/netlogo/ (дата звернення: 14.06.2023). – Назва з екрана.
12. EpiModel [Електронний ресурс] // EpiModel. – Режим доступу: https://www.epimodel.org/ (дата звернення: 14.06.2023). – Назва з екрана.

Додаток А

Лістинги програми

AgentSpawner.cs

```
using UnityEngine;
using UnityEngine.AI;
using Random = UnityEngine.Random;

namespace Agent
{
   public class AgentSpawner : MonoBehaviour
   {
      [Range(1, 500)]
      public int actorsCount;
      public GameObject agentPrefab;

      public Transform topLeftCornerBoundary;
      public Transform bottomRightCornerBoundary;

      private void Start()
      {
         for (var i = 0; i < actorsCount; i++)
         {
            Instantiate(agentPrefab, GetRandomPosition(), Quaternion.identity);
         }
      }

      private Vector3 GetRandomPosition()
      {
         NavMeshHit hit = default;
         var isFound = false;
         var iterationCounter = 0;

         while (!isFound && iterationCounter < 250)
         {
            var randomPosition = new Vector3(
               Random.Range(bottomRightCornerBoundary.position.x, topLeftCornerBoundary.position.x),
               0,
               Random.Range(bottomRightCornerBoundary.position.z, topLeftCornerBoundary.position.z)
            );

            isFound = NavMesh.SamplePosition(randomPosition, out hit, 10f, 1);
            iterationCounter++;
         }

         return hit.position;
      }

      private void OnDrawGizmosSelected()
      {
         if (topLeftCornerBoundary == null | bottomRightCornerBoundary == null)
            return;

         Gizmos.color = Color.green;
         Gizmos.DrawWireCube(topLeftCornerBoundary.position, new Vector3(1, 100, 1));
         Gizmos.DrawWireCube(bottomRightCornerBoundary.position, new Vector3(1, 100, 1));
      }
   }
}
```

MathConstants.cs

```
using System.Collections.Generic;
using SimulationTime;
using UnityEngine;

namespace Data
```

```
{
    public struct ContactRateEntry {
        public TimeOfDay timeOfDay;
        public float rate;
    }

    public class MathConstants
    {
        [Header("Diseases Spread")]
        public float bornAdnDeathRate;
        public float latentPeriod;
        public float meanInfectiousPeriod;
        public List<ContactRateEntry> contactRates;
    }
}
```

Agent.cs

```
using System;
using UnityEngine;
using UnityEngine.AI;

namespace MobilitySimulation
{
    public interface IAgentState
    {
        public bool IsWaitingForMovementTarget();
    }

    public class Moving : IAgentState
    {
        public bool IsWaitingForMovementTarget() => false;
    }

    public class Waiting : IAgentState
    {
        public bool IsWaitingForMovementTarget() => true;
    }

    [RequireComponent(typeof(NavMeshAgent))]
    public class Agent : MonoBehaviour
    {
        public IAgentState currentState = new Waiting();

        [SerializeField]
        public Region currentRegion;

        private NavMeshAgent _navMeshAgent;

        public void Awake()
        {
            _navMeshAgent = GetComponent<NavMeshAgent>();
        }

        public void Update()
        {
            if (_navMeshAgent.pathPending)
                return;

            if (_navMeshAgent.remainingDistance > _navMeshAgent.stoppingDistance)
                return;

            if (_navMeshAgent.hasPath || _navMeshAgent.velocity.sqrMagnitude != 0f)
                return;

            currentState = new Waiting();
        }

        public void MoveTo(Vector3 position)
        {
            _navMeshAgent.destination = position;
            currentState = new Moving();
        }
```

```
        void OnTriggerEnter(Collider other)
        {
            var region = other.GetComponent<Region>();
            if (region is null)
                return;

            currentRegion = region;
        }
    }
}
```

MathConstants.cs

```
using UnityEngine;

namespace MobilitySimulation
{
    [CreateAssetMenu(menuName="ScriptableObjects/MobilitySimulation/MathConstants")]
    public class MathConstants : ScriptableObject
    {
        public int rho;
        public int alpha;
    }
}
```

PopulationDensity.cs

```
using UnityEngine;

namespace MobilitySimulation
{
    [CreateAssetMenu(menuName="ScriptableObjects/MobilitySimulation/PopulationDensity")]
    public class PopulationDensity : ScriptableObject
    {
        public double valuePer100SquareMeters;
    }
}
```

Region.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using SimulationTime;
using UnityEditor;
using UnityEngine;
using UnityEngine.AI;
using Random = UnityEngine.Random;

namespace MobilitySimulation
{
    [RequireComponent(typeof(BoxCollider))]
    [ExecuteInEditMode]
    public class Region : MonoBehaviour
    {
        [Header("Config")]
        public PopulationDensity populationDensity;
        public RegionPriorityTable regionPriorityTable;
        public Color debugColor;

        [Header("Debug Settings")]
        public double gravity;
        public double populationSize;
        public RegionPriority currentRegionPriority;

        [SerializeField]
        public List<Route<Region>> routes;

        private Transform _transform;
        private Bounds _colliderBounds;

        public void Awake()
        {
            populationSize = CalculatePopulationSize();
            _transform = GetComponent<Transform>();
            _colliderBounds = GetComponent<BoxCollider>().bounds;
```

```
}

public void Update()
{
    if (EditorApplication.isPlaying)
        return;

    Awake();
}

public Vector3 RandomPoint()
{
    var randomPoint = new Vector3(
        Random.Range(_colliderBounds.min.x, _colliderBounds.max.x),
        Random.Range(_colliderBounds.min.y, _colliderBounds.max.y),
        Random.Range(_colliderBounds.min.z, _colliderBounds.max.z)
    );

    NavMesh.SamplePosition(randomPoint, out var hit, 10f, 1);
    return hit.position;
}

public double CalculateGravityTo(Region otherRegion, MathConstants constants, TimeOfDay timeOfDay)
{
    var distance = CalculateDistanceTo(otherRegion);
    var otherRegionGravity = otherRegion.CalculateGravity(timeOfDay);

    return otherRegionGravity * Math.Pow(1 + distance / constants.rho, -constants.alpha);
}

private double CalculateGravity(TimeOfDay timeOfDay)
{
    gravity = CalculatePopulationSize() * CalculateRegionPriority(timeOfDay);
    return gravity;
}

public double CalculateDistanceTo(Region otherRegion)
{
    var path = new NavMeshPath();
    NavMesh.CalculatePath(
        new Vector3(_transform.position.x, 0, _transform.position.z),
        new Vector3(otherRegion._transform.position.x, 0, otherRegion._transform.position.z),
        1,
        path
    );

    if (path.corners.Length < 2)
        return 0;

    var lengthSoFar = 0.0F;
    for (var i = 1; i < path.corners.Length; i++) {
        lengthSoFar += Vector3.Distance(path.corners[i - 1], path.corners[i]);
    }

    return Math.Round(lengthSoFar, 2);
}

private double CalculatePopulationSize()
{
    var area = (_colliderBounds.max.x - _colliderBounds.min.x) *
            (_colliderBounds.max.z - _colliderBounds.min.z);

    populationSize = populationDensity.valuePer100SquareMeters * area;
    return populationSize;
}

private double CalculateRegionPriority(TimeOfDay timeOfDay)
{
    currentRegionPriority = regionPriorityTable.GetRegionPriority(timeOfDay);
    return currentRegionPriority switch
    {
        RegionPriority.Low => 1,
        RegionPriority.Medium => 2,
```

```
            RegionPriority.High => 6,
            _ => throw new ArgumentOutOfRangeException()
        };
    }
  }
}
```

RegionPriorityTable.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using UnityEngine;
using SimulationTime;

namespace MobilitySimulation
{
  public enum RegionPriority
  {
    Low,
    Medium,
    High
  }

  [Serializable]
  public struct RegionPriorityEntry {
    public TimeOfDay timeOfDay;
    public RegionPriority priority;
  }


  [CreateAssetMenu(menuName="ScriptableObjects/MobilitySimulation/RegionPriorityTable")]
  public class RegionPriorityTable : ScriptableObject
  {
    public List<RegionPriorityEntry> priorityTable;


    public RegionPriority GetRegionPriority(TimeOfDay timeOfDay)
    {
      return (
        from priorityRow in priorityTable
        where priorityRow.timeOfDay == timeOfDay
        select priorityRow.priority
      ).FirstOrDefault();
    }
  }
}
```

Route.cs

```
using System;
using UnityEngine;

namespace MobilitySimulation
{
  [Serializable]
  public class Route<T> where T: MonoBehaviour
  {
    public T origin;
    public T destination;

    public double gravity;
    public double probabilityOfUsage;

    public Route(T origin, T destination, double gravity)
    {
      this.origin = origin;
      this.destination = destination;
      this.gravity = gravity;
      probabilityOfUsage = -1;
    }
  }
}
```

RouteBuilder.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using SimulationTime;
using UnityEditor;
using UnityEngine;

namespace MobilitySimulation
{

   [ExecuteInEditMode]
   public class RouteBuilder : MonoBehaviour
   {
      [SerializeField]
      private MathConstants mathConstants;

      [SerializeField]
      private List<Region> regions;

      private TimeOfDay _currentTimeOfDay;

      public void Start()
      {
         _currentTimeOfDay = TimeOfDay.Morning;
         BuildRoutes();
      }

      public void Update()
      {
         if (EditorApplication.isPlaying)
            return;

         Start();
      }

      [ContextMenu("Build Routes")]
      public void BuildRoutes()
      {
         regions = FindObjectsOfType<Region>().ToList();
         if (regions == null)
            return;

         foreach (var regionA in regions)
         {
            var newRegionARoutes = new List<Route<Region>>();
            foreach (var regionB in regions) {
               var route = new Route<Region>(
                  regionA,
                  regionB,
                  regionA.CalculateGravityTo(regionB, mathConstants, _currentTimeOfDay)
               );

               newRegionARoutes.Add(route);
            }

            var totalGravitySum = newRegionARoutes
               .ConvertAll(region => region.gravity)
               .Sum();

            foreach (var route in newRegionARoutes)
               route.probabilityOfUsage = Math.Round(route.gravity / totalGravitySum, 2);

            regionA.routes = newRegionARoutes;
         }
      }

      [ContextMenu("Clean All Routes")]
      public void CleanAllRoutes()
      {
         if (regions == null)
```

```
            return;

        foreach (var region in regions)
        {
            region.routes = new List<Route<Region>>();
        }
    }
  }
}
```

Simulation.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using UnityEditor;
using UnityEngine;
using Random = System.Random;
using UnityRandom = UnityEngine.Random;


namespace MobilitySimulation
{
  [ExecuteInEditMode]
  public class Simulation : MonoBehaviour
  {
    [SerializeField]
    private List<Agent> agents;

    [SerializeField]
    private List<Region> regions;

    private Dictionary<Region, int> _agentsPerRegion = new();

    private void Start()
    {
      agents = FindObjectsOfType<Agent>().ToList();
      regions = FindObjectsOfType<Region>().ToList();

      foreach (var region in regions)
      {
        _agentsPerRegion[region] = 0;
      }
    }

    public void FixedUpdate()
    {
      foreach (var agent in agents)
      {
        if (!agent.currentState.IsWaitingForMovementTarget())
          continue;

        var currentRegion =
          agent.currentRegion ?
            agent.currentRegion :
            regions[new Random().Next(regions.Count)];

        if (agent.currentRegion)
          _agentsPerRegion[agent.currentRegion] -= 1;

        var destinationRegion = PickRandomDestination(currentRegion);
        _agentsPerRegion[destinationRegion] += 1;

        agent.MoveTo(destinationRegion.RandomPoint());
      }
    }

    private static Region PickRandomDestination(Region region)
    {
      var random = (float) UnityRandom.Range(0, 100) / 100;
      var accumulatedProbability = 0.0;

      foreach (var route in region.routes)
      {
```

```
            if (accumulatedProbability + route.probabilityOfUsage >= random)
                return route.destination;

            accumulatedProbability += route.probabilityOfUsage;
        }

        return region.routes.Last().destination;
    }

    public void OnDrawGizmos()
    {
        if (!EditorApplication.isPlaying)
            return;

        var totalAgentCount = agents.Count;
        foreach (var (region, agentsCount) in _agentsPerRegion)
        {
            var regionPosition = region.gameObject.GetComponent<Transform>().position;

            Gizmos.color = Color.red;
            Gizmos.DrawCube(
                new Vector3(regionPosition.x, 50, regionPosition.z),
                new Vector3(10, ((float) agentsCount / totalAgentCount) * 100, 10)
            );
        }

    }
  }
}
```

Time.cs

```
using System;

namespace SimulationTime
{
  [Serializable]
  public struct SimulationTime
  {
    private const int TotalSecondsInDay =  60 * 60 * 24;

    public double totalSeconds;
    public int Hours => (int) totalSeconds / 3600 % 3600;
    public int Minutes => (int)totalSeconds / 60 % 60;
    public float PercentageOfDayPassed => (float) totalSeconds / TotalSecondsInDay;

    public TimeOfDay TimeOfDay
    {
      get
      {
        return Hours switch
        {
          > 18 => TimeOfDay.Night,
          > 12 => TimeOfDay.Noon,
          _ => TimeOfDay.Morning
        };
      }
    }

    public void AddSeconds(double seconds)
    {
      totalSeconds += seconds;
      if (totalSeconds >= TotalSecondsInDay)
        totalSeconds = 0;
    }
  }
}
```

TimeController.cs

```
using UnityEngine;
using UnityEngine.Events;

namespace SimulationTime
{
```

```
    public class TimeController : MonoBehaviour
    {
        [Header("Configuration")]
        public float inGameTimeMultiplier = 30;
        public float simulationSpeed = 4;

        [Header("Debug")]
        [SerializeField]
        private SimulationTime inGameTime;

        public UnityEvent<SimulationTime> onTimeChange;

        private void FixedUpdate()
        {
            inGameTime.AddSeconds(Time.deltaTime * inGameTimeMultiplier * simulationSpeed);
            onTimeChange.Invoke(inGameTime);

            Time.timeScale = simulationSpeed;
        }
    }
}


TimeOfDay.cs
namespace SimulationTime
{
    public enum TimeOfDay
    {
        Morning,
        Noon,
        Night
    }
}


CameraController.cs
using System;
using System.Collections.Generic;
using UnityEngine;

[RequireComponent(typeof(Camera))]
public class CameraController : MonoBehaviour
{
    [Serializable]
    public struct PositionMemento
    {
        public Vector3 position;
        public Quaternion rotation;
    }

    [SerializeField]
    private float speed = 5F;

    [SerializeField]
    private List<PositionMemento> mementos = new ();

    private int _currentMementoIndex;


    private Transform _transform;

    private void Start()
    {
        _transform = GetComponent<Transform>();

        RestoreSnapshot(mementos[0]);
        _currentMementoIndex = 0;
    }

    void Update()
    {
        var currentMemento = mementos[_currentMementoIndex];
```

```
        var isPositionEqualToMementoPosition = (_transform.position - currentMemento.position).magnitude < 1;
        var isRotationEqualsToMementoRotation =
            (_transform.rotation.eulerAngles - currentMemento.rotation.eulerAngles).magnitude < 1;

        var isStillRestoringMemento = !isPositionEqualToMementoPosition || !isRotationEqualsToMementoRotation;
        if (isStillRestoringMemento)
        {
            transform.position = Vector3.Lerp(
                transform.position,
                currentMemento.position,
                Time.unscaledDeltaTime * speed
            );
            transform.rotation = Quaternion.Lerp(
                transform.rotation,
                currentMemento.rotation,
                Time.unscaledDeltaTime * speed
            );
        }

        if (Input.GetKeyDown(KeyCode.D))
            RestoreNextSnapshot();
        else if (Input.GetKeyDown(KeyCode.A))
            RestorePrevSnapshot();
    }

    [ContextMenu("Make Snapshot")]
    public void MakeSnapshot()
    {
        mementos.Add(new PositionMemento
        {
            position = transform.position,
            rotation = transform.rotation
        });
    }

    private void RestoreNextSnapshot()
    {
        if (_currentMementoIndex != (mementos.Count - 1))
            _currentMementoIndex += 1;
        else
            _currentMementoIndex = 0;
    }

    private void RestorePrevSnapshot()
    {
        if (_currentMementoIndex != 0)
            _currentMementoIndex -= 1;
        else
            _currentMementoIndex = mementos.Count - 1;
    }

    private void RestoreSnapshot(PositionMemento memento)
    {
        _transform.position = memento.position;
        _transform.rotation = memento.rotation;
    }

}
```

SunController.cs

```
using UnityEngine;

[RequireComponent(typeof(Light))]
public class SunController : MonoBehaviour
{
    private Transform _transform;

    private void Start()
    {
        _transform = GetComponent<Transform>();
    }

    public void OnTimeUpdate(SimulationTime.SimulationTime time)
```

```
    {
        _transform.rotation = Quaternion.Euler(180 * time.PercentageOfDayPassed, 0, 0);
    }

}
```

Додаток Б

Ілюстративний матеріал

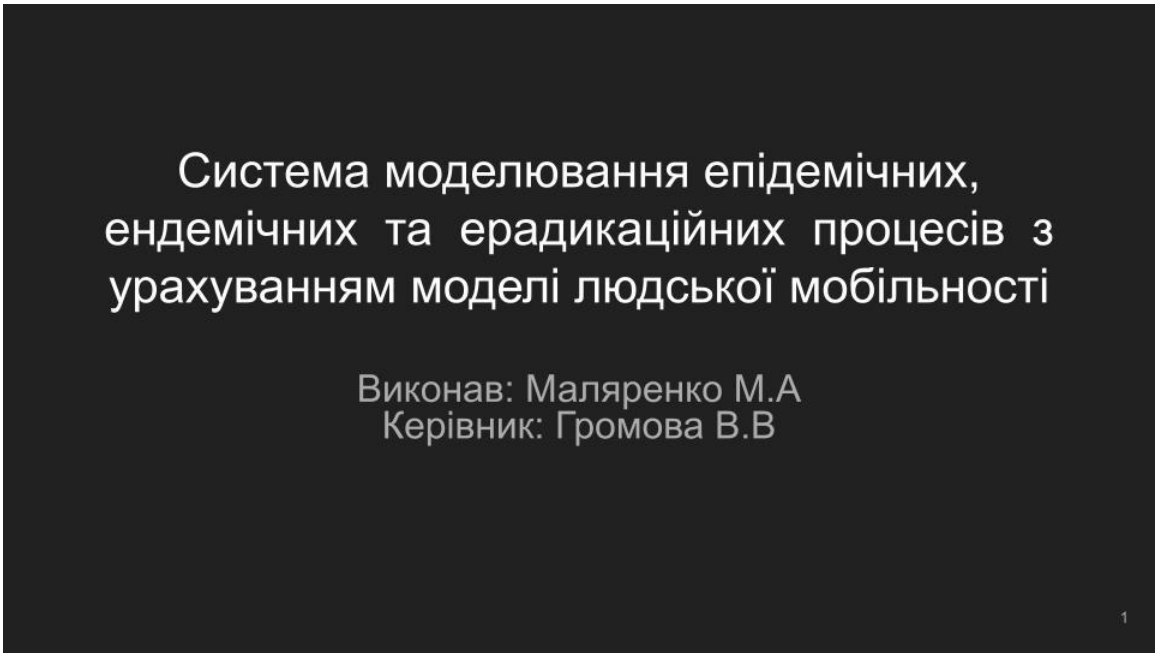

Рисунок Б.1 - Слайд 1

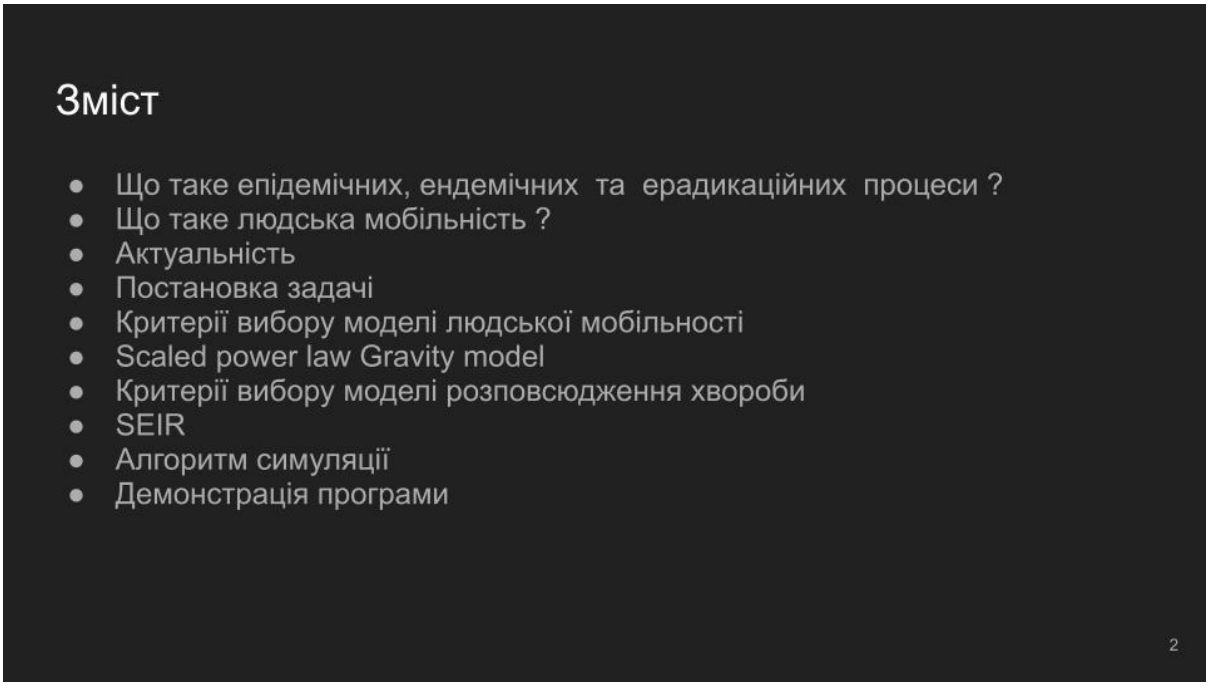

Рисунок Б.2 - Слайд 2

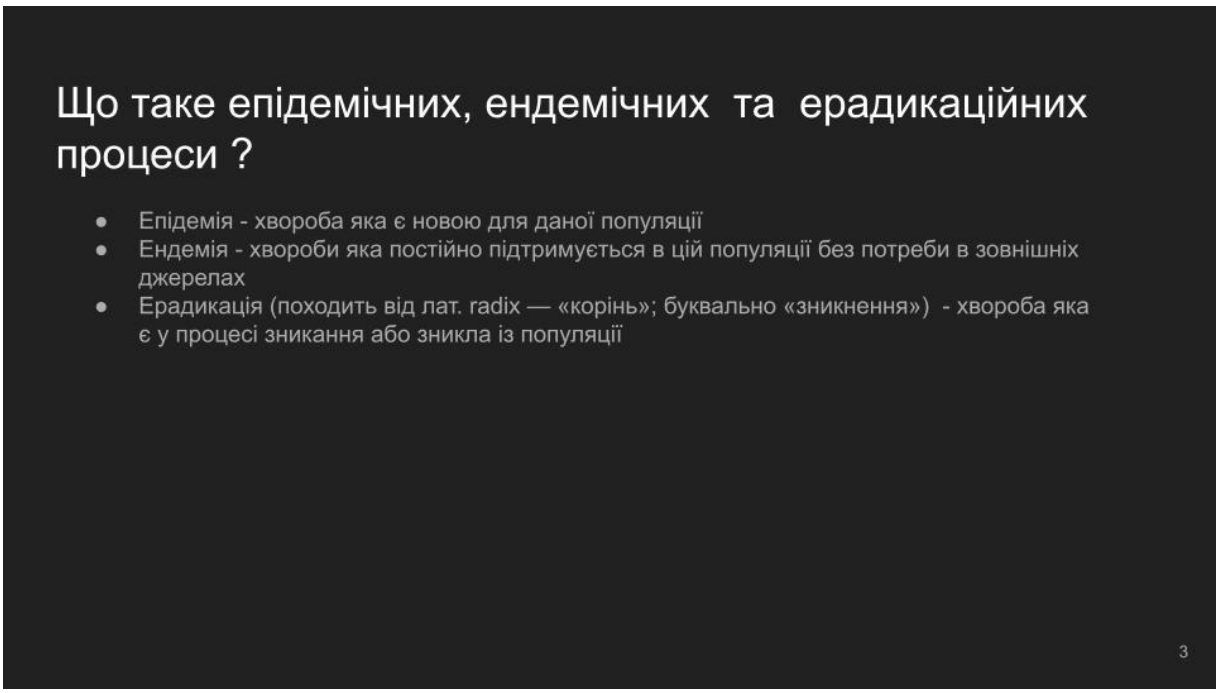

Рисунок Б.3 - Слайд 3

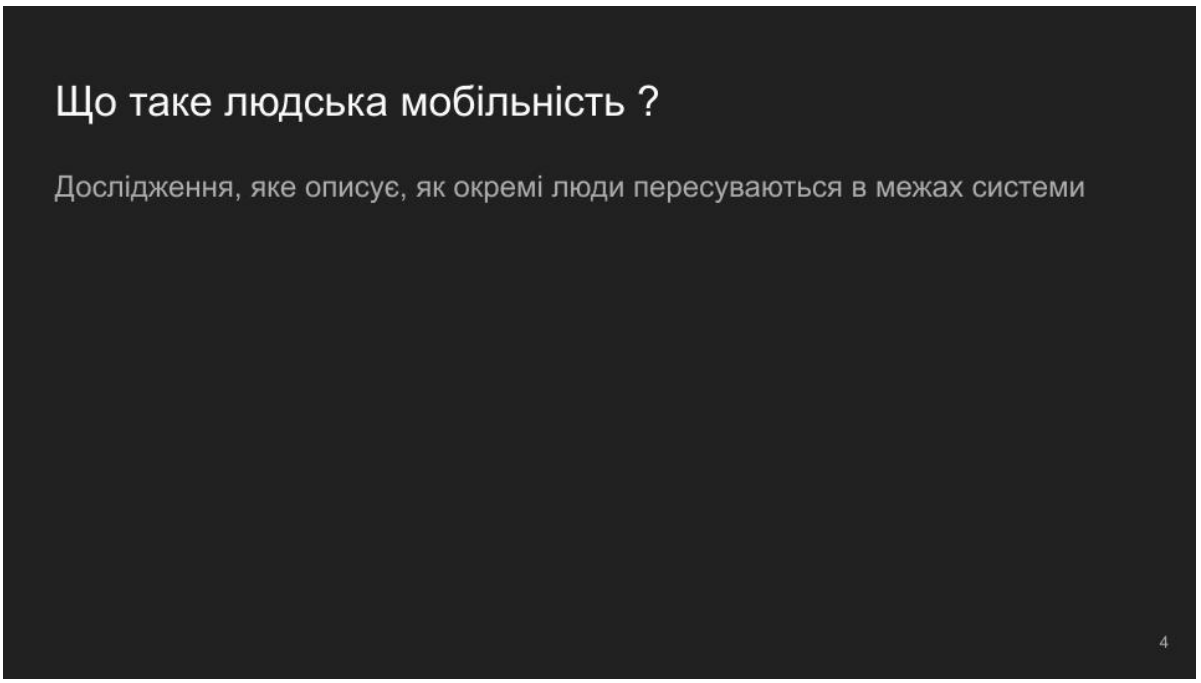

Рисунок Б.4 - Слайд 4

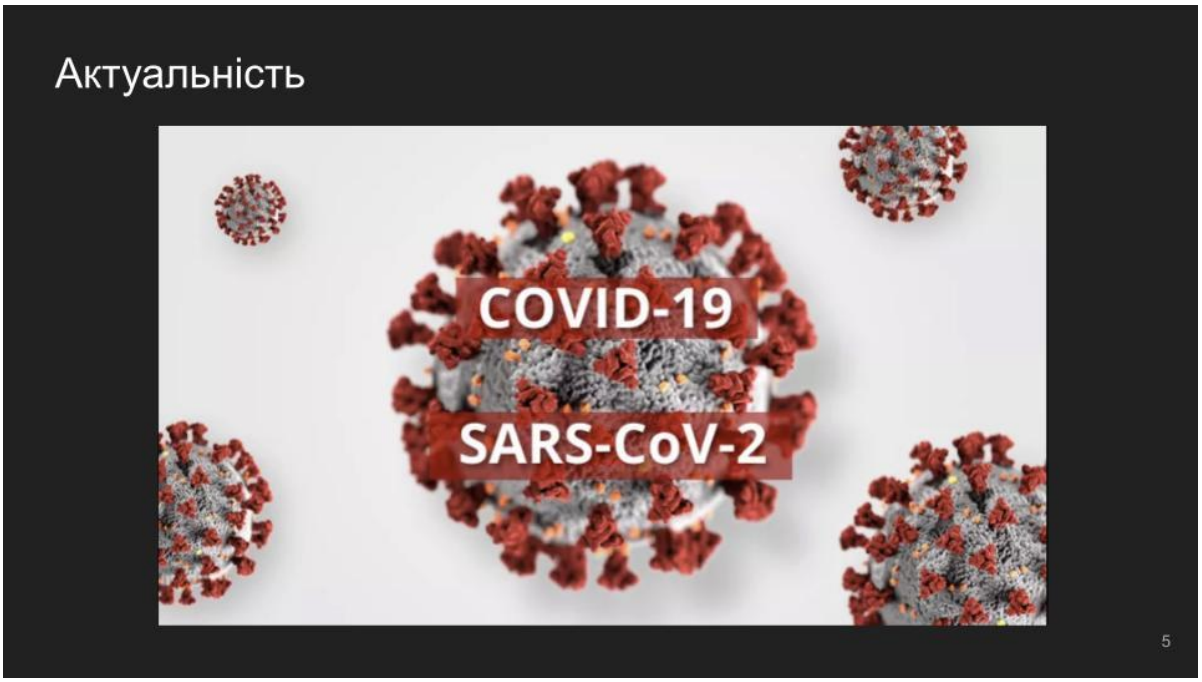

Рисунок Б.5 - Слайд 5

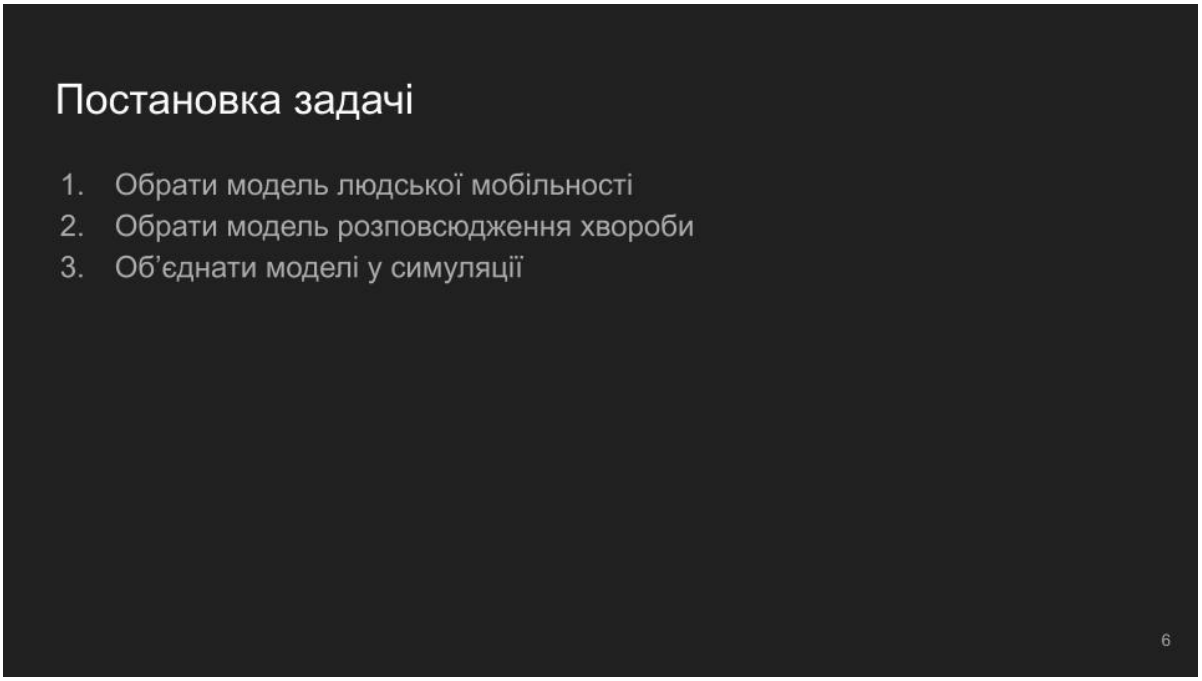

Рисунок Б.6 - Слайд 6

Критерії вибору моделі людської мобільності

- Масштабованість
- Точність
- Часові та просторові зміни
- Гнучкість

7

Рисунок Б.7 - Слайд 7

Scaled power law Gravity model

$$\lambda_{ij} = N_j^{\omega} \left(1 + \frac{d_{ij}}{\rho}\right)^{-\alpha}$$

8

Рисунок Б.8 - Слайд 8

Модифікований Scaled power law Gravity model

$$\lambda_{ij}(t) = N_j^{\omega}(t)\left(1 + \frac{d_{ij}}{\rho(t)}\right)^{-\alpha(t)}$$

$$N(t) = \rho_d * S_r$$

9

Рисунок Б.9 - Слайд 9

Критерії вибору моделі розповсюдження хвороби

- Характеристики хвороби
- Масштабованість
- Інтерпретованість

10

Рисунок Б.10 - Слайд 10

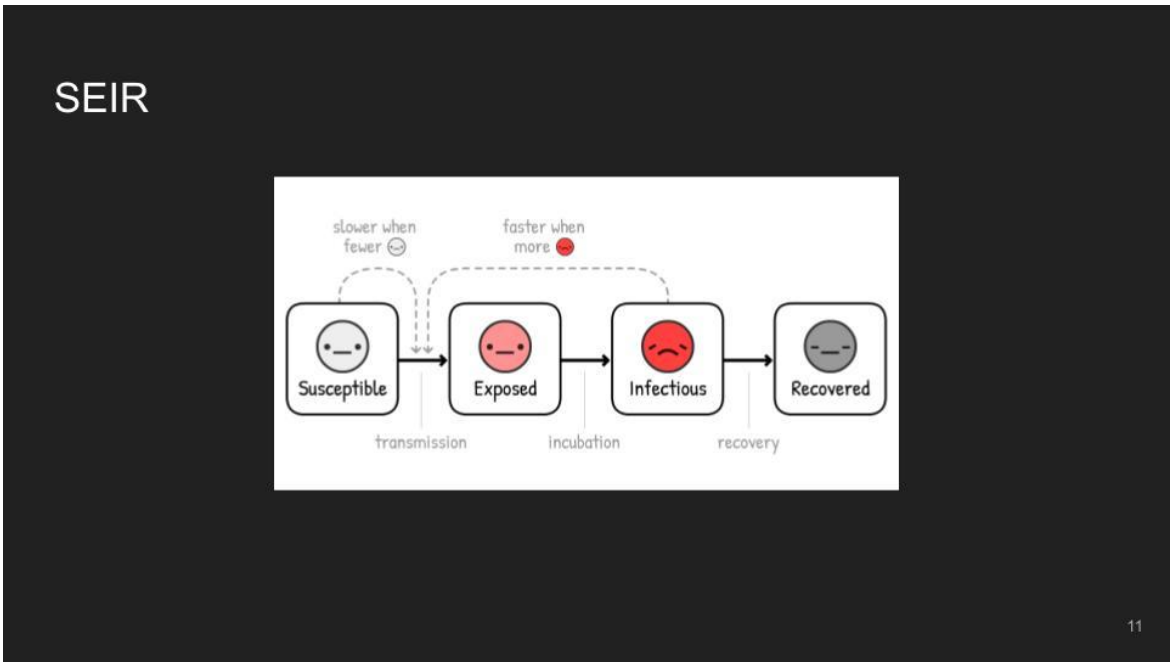

Рисунок Б.11 - Слайд 11

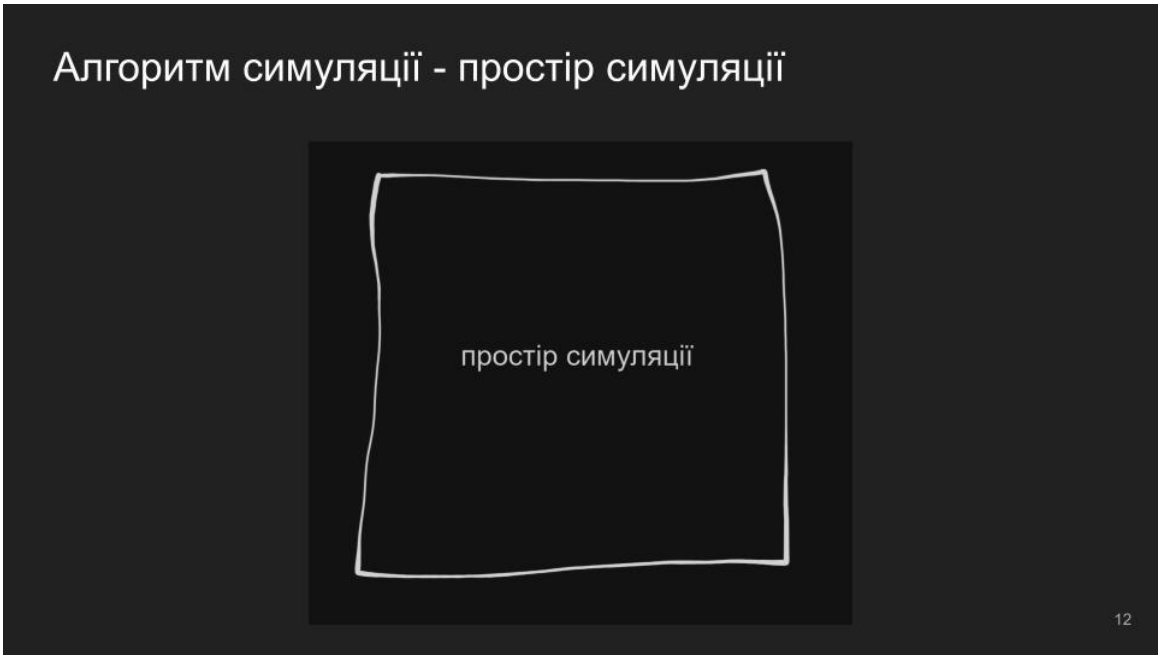

Рисунок Б.12 - Слайд 12

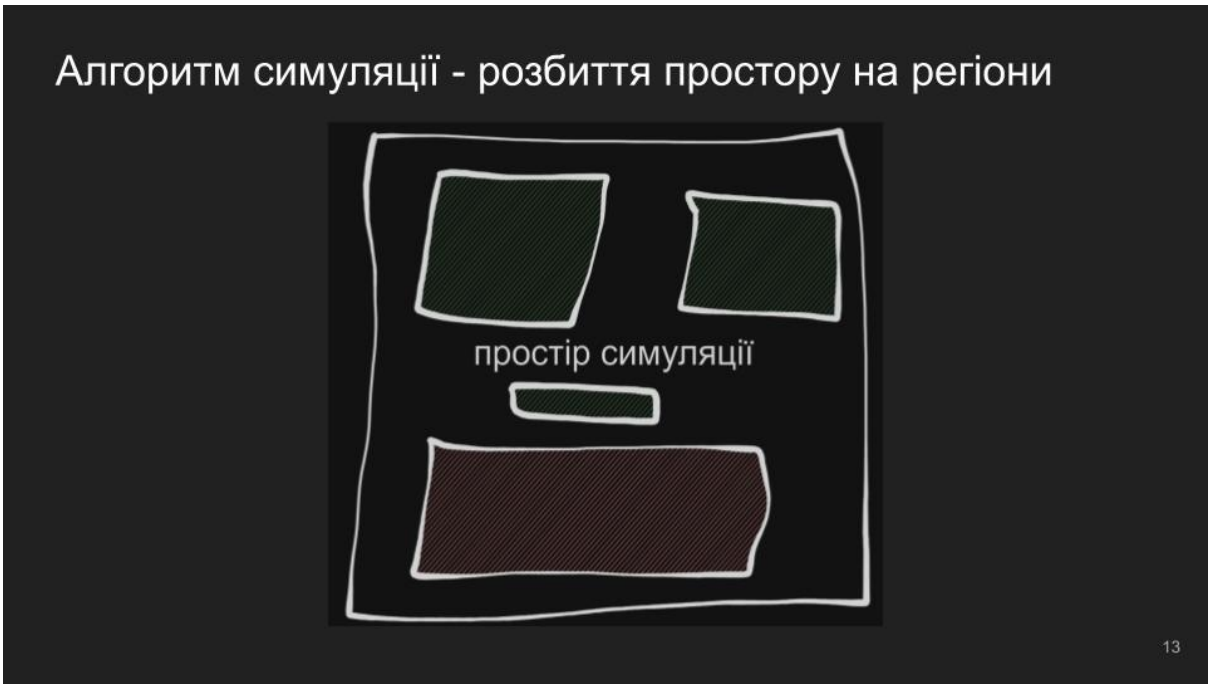

Рисунок Б.13 - Слайд 13

Рисунок Б.14 - Слайд 14

Рисунок Б.15 - Слайд 15

Рисунок Б.16 - Слайд 16