НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

УДК 004.8:004.93

«ПОГОДЖЕНО»	**«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»**
Декан факультету інформаційних технологій	Завідувач кафедри комп'ютерних наук
Болбот І.М., д.п.н., професор	Голуб Б.Л., к.т.н., доцент
_____	_____
_____ _____ 2024 р.	**_____ _____ 2024 р.**

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему _Інтелектуальна система симуляції мікроскопічних істот_____

Спеціальність __121 "Інженерія програмного забезпечення_____

(код і назва)

Освітня програма __Програмне забезпечення інформаційних систем_____

(назва)

Орієнтація освітньої програми ____освітньо-професійна_____

(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми

____доцент, к.е.н____ _____ ___Семко В.В.___

(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник магістерської кваліфікаційної роботи

_____ст викладач_____ _____ _____Міловідов Ю.О._____

(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ _____Замниус А.О._____

(підпис) (ПІБ студента)

КИЇВ-2024

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

доцент к.т.н. Голуб Б. Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"1" листопада 2023 року

З А В Д А Н Н Я
ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

Замниус А.О.
(прізвище, ім'я, по батькові)

Спеціальність 121 «інженерія програмного забезпечення»
(код і назва)

Освітня програма Програмне забезпечення інформаційних систем
(назва)

Орієнтація освітньої програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Тема магістерської кваліфікаційної роботи Інтелектуальна система симуляції мікроскопічних істот

затверджена наказом ректора НУБіП України від "1" листопада 2023р. № 2002 С
Термін подання завершеної роботи на кафедру 2024.11.29
(рік, місяць, число)

Вхідні дані до магістерської кваліфікаційної роботи: Наукові публікації, що висвітлюють застосування генетичних алгоритмів та методів моделювання популяцій мікроорганізмів, а також технічна документація платформи Unity і мови C#.

Перелік питань, що підлягають дослідженню:

Провести аналіз існуючих рішенньі та описати їх плючи та мінуси.

Розробити концепцію та архітектуру додатка для комп'ютерної симуляції

Провести тестування готової системи та проаналізувати результати роботи.

Проаналізувати можливості подальшого розвитку системи та дослідженнь в цій галузі.

Перелік графічного матеріалу (за потреби)

Дата видачі завдання "1" листопада 2023 р.

Керівник магістерської кваліфікаційної роботи Міловідов Ю.О.
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання Замниус А.О. .
(підпис) (прізвище та ініціали студента)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕННЬ

- GA – Genetic Algorithm (генетичний алгоритм);
- UI – User Interface (користувацький інтерфейс);
- API – Application Programming Interface (інтерфейс програмування застосунків);
- RAM – Random Access Memory (оперативна пам'ять);
- GPU – Graphics Processing Unit (графічний процесор);
- CPU – Central Processing Unit (центральний процесор);
- DNA – Deoxyribonucleic Acid (дезоксирибонуклеїнова кислота, умовний код еволюції організму);
- 2D – Two-Dimensional (двовимірний простір);
- 3D – Three-Dimensional (тривимірний простір).

ВСТУП

Сучасний розвиток інформаційних технологій стимулює створення інтелектуальних систем, здатних моделювати й аналізувати складні процеси. Одним із таких напрямів є симуляція мікроскопічних істот – організмів, які існують на мікроскопічному рівні, таких як бактерії, віруси, амеби, тощо. Вивчення їхньої поведінки має важливе значення для біології, медицини, екології та біоінженерії. Завдяки застосуванню сучасних комп'ютерних технологій можливо створювати системи, які дозволяють симулювати життя цих організмів, прогнозувати їх взаємодію з оточенням і проводити експерименти віртуально, мінімізуючи витрати на реальні дослідження.

Актуальність теми обумовлена потребою в розробці ефективних інструментів для моделювання складних біологічних систем. Класичні підходи, засновані на математичному описі й фізичних експериментах, часто є трудомісткими, дорогими та обмеженими в масштабах. Інтелектуальні системи симуляції здатні вирішувати ці проблеми, забезпечуючи можливість вивчення динамічних процесів, моделювання різних умов існування та оцінки впливу зовнішніх факторів на поведінку мікроскопічних організмів.

Мета роботи є розробка інтелектуальної системи для симуляції мікроскопічних істот, яка враховує їхні біологічні характеристики, поведінкові моделі й середовище існування. Основна увага приділяється створенню програмної платформи, яка дозволяє користувачам легко проводити моделювання та отримувати релевантні наукові результати.

Наукова новизна полягає у використанні сучасних методів штучного інтелекту, зокрема машинного навчання та агентних підходів, для симуляції взаємодії мікроскопічних істот у складних середовищах. Це дає змогу моделювати емерджентні властивості, тобто явища, що виникають унаслідок колективної поведінки організмів, і які важко передбачити аналітичними методами.

Практична значущість полягає в можливості застосування розробленої системи для вирішення прикладних задач:

- прогнозування розвитку захворювань, спричинених мікроорганізмами;
- оптимізації антибактеріальних стратегій;
- моделювання впливу екологічних змін на мікроскопічні екосистеми;
- створення освітніх симуляцій для популяризації наукових знань.

Апробація. За результатами роботи були опубліковані тези під назвами:

1. «Інтелектуальна система симуляції мікроскопічних істот» в збірнику наукових праць за матеріалами 6-ї всеукраїнської науково-практичної конференції студентів і аспірантів “теоретичні та прикладні аспекти розробки комп’ютерних систем” за 25 квітня 2024 р.

2. «Інтелектуальна система симуляції мікроскопічних істот» в збірнику наукових праць за матеріалами 15-ї Міжнародної науково-практичної конференції молодих вчених «Інформаційні технології: економіка, техніка, освіта» за 8 листопада 2024 р.

Зміст публікацій представлено в Додатку А.

У вступі окреслюється структура роботи:

1. У першому розділі представлено огляд існуючих підходів до симуляції мікроскопічних істот.

2. Другий розділ присвячено розробці концепції й архітектури інтелектуальної системи.

3. У третьому розділі описуються результати тестування та практичного застосування системи.

4. Заключний розділ містить висновки та перспективи подальших досліджень.

Таким чином, магістерська робота спрямована на внесок у розвиток сучасних технологій для дослідження мікросвіту, що може мати як наукову, так і прикладну цінність.

ОГЛЯД ІСНУЮЧИХ ПІДХОДІВ ДО СИМУЛЯЦІЇ МІКРОСКОПІЧНИХ ІСТОТ

1.1 Математичне моделювання

Математичне моделювання є класичним підходом, що використовується для опису популяційних динамік і взаємодій між мікроорганізмами. Воно базується на системах диференціальних рівнянь (ODE) або часткових диференціальних рівнянь (PDE), які описують такі явища, як ріст популяції, поширення хвороб чи метаболічні процеси.

Наприклад, модель Лотки-Вольтерри, яка описує взаємодію хижаків і жертв, може бути адаптована до симуляції бактерій та вірусів. Такі моделі дозволяють аналізувати стійкість екосистем, прогнозувати розвиток епідемій або вплив змін у середовищі на популяцію мікроорганізмів.

Переваги математичного моделювання:

- висока швидкість обчислень;
- відносна простота реалізації для базових сценаріїв.

Недоліки математичного моделювання:

- недостатня гнучкісь прорахунків для складних систем.

1.2 Агентно-орієнтовані системи

Агентно-орієнтовані підходи базуються на моделюванні мікроорганізмів як автономних агентів із визначеними правилами поведінки. Кожен агент взаємодіє з іншими агентами та середовищем, формуючи складну поведінкову динаміку на макрорівні.

Прикладом є системи моделювання росту біоплівок, де бактерії функціонують як агенти, здатні прикріплюватися до поверхні, розмножуватися, виділяти хімічні речовини та взаємодіяти з оточенням.

Переваги агентно-орієнтованих систем:

- гнучкість у налаштуванні поведінки кожного агента;
- можливість моделювання емерджентних явищ.

Недоліки агентно-орієнтованих систем:

- значні обчислювальні ресурси через що дуже складно модулювати велику кількість агентів.

1.3 Застосування методів машинного навчання

Машинне навчання відкриває нові можливості в симуляції мікроскопічних організмів. За допомогою глибоких нейронних мереж можливо моделювати складні процеси, що відбуваються в мікроскопічному середовищі, наприклад:

- прогнозування росту колоній бактерій на основі зовнішніх умов;
- визначення взаємодій між вірусами та клітинами-хазяями;
- виявлення нових закономірностей у поведінці мікроорганізмів.

Останнім часом популярності набули генеративно-змагальні мережі (GANs) для створення віртуальних середовищ, які відтворюють природні мікроскопічні процеси.

Переваги машинного навчання:

- автоматизація складних процесів;
- швидка адаптація до нових даних.

Недоліки машинного навчання:

- метод вимагає великих обсягів даних для навчання, що не завжди доступно.

1.4 Біоінформатичне моделювання

Біоінформатика пропонує підходи, що дозволяють моделювати поведінку мікроскопічних істот на молекулярному рівні. Одним із таких прикладів є симуляція геномів або протеїнів для розуміння метаболічних процесів.

Існують інструменти, які дозволяють відтворювати повний цикл життя мікроорганізмів із урахуванням їх генетичних характеристик. Програми, як-от COPASI або CellDesigner, дозволяють моделювати клітинні мережі, виявляючи ключові вузли для терапевтичного втручання.

Цей підхід має значення для медицини, оскільки дозволяє прогнозувати реакцію мікроорганізмів на ліки або мутації.

1.5 Фізико-хімічні симуляції

Фізико-хімічне моделювання базується на використанні рівнянь фізики та хімії для симуляції міжмолекулярних взаємодій. Це дозволяє відтворювати поведінку організмів у контексті їхнього середовища, включаючи:

- поширення метаболітів;
- транспортування речовин через мембрани;
- хімічні реакції між компонентами середовища.

Сучасні програми, такі як LAMMPS або COMSOL Multiphysics, дозволяють створювати детальні моделі, які включають термодинамічні та гідродинамічні властивості середовища.

Переваги:

- висока точність моделювання фізико-хімічних процесів;
- широке застосування у фундаментальних дослідженнях.

Недоліки:

- високі обчислювальні витрати;
- складність інтеграції з біологічними моделями.

РОЗРОБКА КОНЦЕПЦІЇ Й АРХІТЕКТУРИ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ

2.1 Концепція системи

Мета розробки: основна мета створення інтелектуальної системи симуляції мікроорганізмів — надати користувачам доступний інструмент для моделювання поведінки та розвитку мікроорганізмів без значних витрат на апаратне забезпечення чи спеціалізовані програмні рішення. Система має бути універсальною та орієнтованою на широке коло користувачів: науковців, студентів, лікарів і навіть розробників.

Функціональність: система дозволяє користувачам створювати стартові параметри мікроорганізмів, моделювати їхнє життя, взаємодію з середовищем і один з одним. Особливістю є простота використання та можливість вивчення еволюційних змін у популяціях.

Типи організмів і рівень деталізації: система зосереджується на моделюванні мікроорганізмів, таких як бактерії та віруси. Генетична та фізіологічна різноманітність не враховується на молекулярному рівні; натомість організми мають початкові характеристики, які змінюються в процесі симуляції.

Моделювання охоплює популяційні динаміки, а також еволюційні зміни в організмах. Внутрішньоклітинні процеси чи молекулярні моделі не інтегруються, що забезпечує швидкість і простоту симуляції.

2.2 Архітектура системи

Система складається з кількох ключових модулів:

Модуль користувацького інтерфейсу (UI): має надавати користувачу можливість налаштувати стартові параметри мікроорганізмів (форма, розмір, швидкість розмноження тощо). Інструменти для розташування організмів на полі

симуляції. Візуалізувати динаміки популяції через графіки, діаграми та інфографіку.

Модуль симуляції: Має реалізувати генетичний алгоритм для моделювання еволюційних змін. Обробляти динамічні зміни середовища, які задаються користувачем (температура, доступ до поживних речовин, тощо). Відповідати за моделювання взаємодії між організмами (конкуренція, співіснування, взаємодія з ресурсами).

Модуль обробки даних: Збирає дані симуляції для подальшого аналізу. Генерує звіти у вигляді графіків або моделей.

Вибір технологій

Вибрана Платформа Unity, що дозволяє створювати інтерактивні графічні середовища. Мовою програмування було вибрано C#, що забезпечує гнучкість у розробці алгоритмів і взаємодії між компонентами.

Код програми представлений в Додатку Б.

2.3 Реалізація моделі симуляції

Генетичний алгоритм (ГА) є основним механізмом, що забезпечує еволюцію мікроорганізмів у системі. Це метод оптимізації, натхненний біологічними процесами природного відбору та спадковості. У контексті симуляції мікроорганізмів ГА дозволяє відтворити процеси адаптації популяцій до змін середовища й розвиток нових характеристик організмів.

Етапи роботи генетичного алгоритму:

1. Ініціалізація популяції: початкова популяція організмів створюється з випадковими характеристиками або на основі параметрів, заданих користувачем. Кожен організм має набір атрибутів (наприклад, швидкість розмноження, стійкість до змін температури, здатність до конкуренції за ресурси), які визначають його "генотип".

2. Оцінка пристосованості (Fitness Evaluation): для кожного організму розраховується рівень пристосованості до середовища. Функція пристосованості (Fitness Function) враховує зовнішні умови (температуру, кількість ресурсів тощо) і успішність організму у виживанні та розмноженні.

Наприклад:

$$F=w1\cdot S+w2\cdot R-w3\cdot E$$

- S — стійкість до умов середовища;
- R — здатність знаходити й споживати ресурси;
- E — енергетичні витрати організму;
- w1,w2,w3 — вагові коефіцієнти для врахування важливості кожного параметра.

3. Селекція (Selection): організми з високим рівнем пристосованості мають більше шансів передати свої "гени" наступному поколінню. Використовуються методи рулетка (Roulette Wheel Selection) - організми обираються пропорційно до їхньої пристосованості та турнірна селекція - кілька організмів випадково обираються, і найкращий серед них переходить у наступне покоління.

4. Кросинговер (Crossover) генетична інформація (атрибути організмів) комбінується між двома "батьківськими" організмами, утворюючи нових "нащадків".

Наприклад:

Якщо два організми мають гени [A,B,C][A, B, C][A,B,C] та [X,Y,Z][X, Y, Z][X,Y,Z], то результат кросинговеру може бути [A,Y,Z][A, Y, Z][A,Y,Z];

Основні методи кросинговеру це: одноточковий - гени розбиваються в одній точці, після якої відбувається обмін. Багатоточковий - розбиття в кількох точках для більшої варіативності.

5. Мутація (Mutation) випадкові зміни в генотипі організмів додають різноманітності та дозволяють уникнути локальних максимумів у пристосованості.

Приклад: якщо атрибут "швидкість розмноження" має значення 5, мутація може змінити його на 6. Ймовірність мутації зазвичай низька (наприклад, 1–5%).

6. Формування нового покоління виходить шляхом поєднання найкращих організмів (елітний підхід) і "нащадків", отриманих після кросинговеру та мутації. Кількість організмів у популяції залишається сталою для стабільності симуляції.

7. Процес повторюється, поки не досягнуто заданої кількості поколінь або поки популяція не адаптується до середовища (наприклад, досягнення заданої пристосованості).

Генотип представляє набір характеристик, які зберігаються у вигляді числових значень або параметрів, таких як швидкість розмноження, розмір тощо. Фенотип є проявом цих характеристик у середовищі, що безпосередньо впливає на виживання. Наприклад, велика швидкість розмноження може підвищувати шанс виживання.

Організми мають адаптуватися до змін у середовищі, таких як зниження температури. Генетичний алгоритм сприяє появі нових характеристик, які покращують шанси на виживання організмів.

Щоб уникнути надмірного використання ресурсів, кількість характеристик організмів і складність функцій обмежуються.

Користувач може змінювати параметри генетичного алгоритму, зокрема швидкість мутацій, метод селекції, кількість поколінь тощо.

Приклад сценарію роботи генетичного алгоритму

Створюється початкова популяція з 50 організмів, кожен із трьома характеристиками: швидкістю розмноження (у діапазоні від 1 до 10), стійкістю до температури (від -10 °C до +10 °C) та здатністю до конкуренції за ресурси (від 1 до 10). Користувач задає температуру середовища +5 °C і встановлює обмежену кількість поживних речовин.

Організми з низькою стійкістю до температури або недостатньою здатністю до конкуренції за ресурси не виживають. Генетичний алгоритм формує нове

покоління, зберігаючи найкращі характеристики організмів. З кожним циклом симуляції популяція стає краще адаптованою до умов середовища.

Після 20 поколінь популяція здебільшого складається з організмів, здатних виживати при +5 °C та ефективно конкурувати за ресурси.

Користувач задає початкові параметри середовища (температура, рівень поживних речовин), які впливають на виживання та розмноження організмів. Ці параметри можуть змінюватися з часом, створюючи додаткові виклики для організмів.

2.4 Особливості користувацького інтерфейсу

Графічний інтерфейс повинен бути інтуїтивно зрозумілим і включати такі ключові функції:

Налаштування стартових умов. Користувач може вручну вводити параметри організмів і середовища або використовувати шаблони з бібліотеки.

Поле симуляції. Інтерфейс має візуалізувати процес взаємодії організмів і змін у середовищі. Важливі події, такі як утворення нових видів або зникнення популяцій, повинні підсвічуватися.

Аналітика. Передбачено відображення графіків росту популяцій, звітів про еволюційні зміни та візуалізацію впливу зовнішніх умов на популяцію.

2.5 Потенційні обмеження системи

Відсутність молекулярного рівня моделювання є важливим аспектом, який може значно знизити точність симуляцій для наукових досліджень, де внутрішньоклітинні процеси відіграють ключову роль. Наприклад, вивчення генетичних мутацій, метаболічних шляхів або молекулярної взаємодії між мікроорганізмами залишається поза межами можливостей поточної версії системи. Це може обмежувати її використання в галузях, де необхідна деталізація на рівні біохімічних процесів, таких як біотехнологія або молекулярна біологія.

Вимоги до апаратних ресурсів також створюють певні бар’єри для користувачів. Система орієнтована на роботу на локальних комп'ютерах, однак виконання складних симуляцій із великою кількістю організмів або змінними параметрами середовища може перевантажувати апаратне забезпечення. Наприклад, моделювання динаміки декількох популяцій з великою кількістю факторів може значно знизити продуктивність навіть на потужних ПК. Це може викликати потребу у використанні оптимізаційного коду або розгляді можливостей масштабування через хмарні обчислення.

Верифікація вхідних даних є ще одним важливим обмеженням. У системі не передбачено автоматичної перевірки коректності введених даних користувачем. Це може стати причиною некоректних результатів, якщо введені дані не відповідають реальним умовам. Наприклад, введення нереалістичних параметрів для організмів або середовища (наприклад, значень, які фізично неможливі) може призводити до непередбачуваних або недостовірних результатів. Для зменшення цього ризику можна було б додати модуль базової перевірки даних, який би аналізував допустимість значень перед запуском симуляції.

РЕЗУЛЬТАТИ ТЕСТУВАННЯ ТА ПРАКТИЧНЕ ЗАСТОСУВАННЯ СИСТЕМИ

3.1 Цілі тестування

Тестування системи мало на меті перевірку наступних аспектів:

1. Функціональність — оцінка коректної роботи всіх модулів системи, включно з налаштуванням параметрів організмів, симуляцією еволюції та відображенням результатів.
2. Продуктивність — вимірювання швидкості обчислень і симуляцій у різних умовах.
3. Стабільність — перевірка роботи системи під час тривалих симуляцій і в умовах навантаження (великої кількості організмів або різких змін параметрів).
4. Користувацький інтерфейс — тестування зручності та інтуїтивності взаємодії з графічним інтерфейсом.

3.2 Тестове середовище

Основна конфігурація для тестування:

- процесор: 12-ядерний Intel Core i5-13500H (2.6 – 4.7 ГГц);
- оперативна пам’ять: 32 ГБ DDR5-5200 МГц;
- дискретна відеокарта: NVIDIA GeForce RTX 4060, 8 ГБ відеопам’яті.

Додаткові конфігурації:

1. ПК середнього класу: процесор Intel Core i5-10400F, оперативна пам’ять 16 ГБ DDR4-3200 МГц, відеокарта NVIDIA GTX 1650 (4 ГБ).
2. Ноутбук базового рівня: процесор AMD Ryzen 5 5500U, оперативна пам’ять 8 ГБ DDR4-2666 МГц, відеокарта інтегрована AMD Radeon Vega.
3. Високопродуктивний ПК: процесор AMD Ryzen 9 7950X, оперативна пам’ять 64 ГБ DDR5-6000 МГц, відеокарта NVIDIA RTX 4090 (24 ГБ).

3.3 Методи тестування

Сценарії тестування:

1. Симуляція еволюції 100 організмів у стабільному середовищі.
2. Створення популяцій із різними початковими параметрами (швидкість розмноження, стійкість до температури).
3. Зміна умов середовища кожні 10 хвилин (зміна температури, кількості ресурсів).
4. Зіткнення двох популяцій з різними характеристиками та аналіз їхньої взаємодії.
5. Максимальне навантаження: симуляція 10 000 організмів одночасно.
6. Вивчення адаптації організмів до нових умов середовища (наприклад, раптового падіння температури).
7. Симуляція безперервної еволюції протягом 48 годин.

Стрес-тести:

1. Перевірка роботи системи з максимальною кількістю організмів (10 000).
2. Різкі зміни середовища, наприклад, зниження температури на 30 °C протягом кількох секунд.

3.4 Результати тестування

Ключові показники:

1. Середній час симуляції: для 100 організмів — 1 секунда на крок еволюції. Для 10 000 організмів — 12 секунд на крок.
2. Стабільність: система витримала безперервну роботу протягом 48 годин.
3. Точність моделювання: адаптація організмів до змін середовища відповідала очікуваним еволюційним сценаріям.

Виявлені недоліки:

1. При великій кількості організмів (понад 8 000) інтерфейс працює з помітними затримками.
2. Недостатня деталізація візуалізації для складних сценаріїв.
3. Відсутність збереження прогресу симуляції в середині процесу.

Зміни після тестування:

1. Оптимізовано алгоритми обробки даних для великих популяцій.
2. Додано функцію автоматичного збереження стану симуляції.
3. Поліпшено графічну візуалізацію результатів.

3.5 Практичне застосування

Сценарії використання:

1. Вивчення адаптації мікроорганізмів до різних температур і ресурсів.
2. Моделювання конкуренції між двома популяціями за обмежені ресурси.
3. Аналіз впливу стійкості організмів до умов середовища на швидкість їхньої еволюції.

Спеціальні сценарії:

1. Симуляція колонізації нової екосистеми організмами з випадковими параметрами.
2. Моделювання еволюції популяції під дією негативного фактора (наприклад, токсичної речовини).
3. Дослідження ефективності "генетичних мутацій" у виживанні організмів.

Реальні біологічні процеси, які вдалося відтворити:

1. Адаптація бактерій до антибіотиків: за допомогою симуляції вдалося показати, як мутації підвищують стійкість популяції.

2. Ефект міжвидової конкуренції: більш пристосовані організми витісняли менш ефективних конкурентів.

3. Еволюція в умовах змінного середовища: організми з високою пластичністю характеристик мали перевагу.

3.6 Порівняння з аналогами

Ефективність:

1. Швидкість обробки даних для великих популяцій (у середньому на 20% швидше) ніж в аналогів.

2. Зручність користувацького інтерфейсу краща ніж в аналогів.

3. Простота налаштування початкових умов симуляції легша ніж в аналогів.

Унікальні функції:

1. Динамічне змінення параметрів середовища в реальному часі.

2. Генерація візуальних графіків і статистики під час симуляції.

3. Підтримка "збереження" прогресу для тривалих симуляцій.

3.7 Можливості вдосконалення

Проблеми:

1. Високі вимоги до ресурсів ПК для симуляцій із понад 10 000 організмів.

2. Відсутність можливості автоматичної інтеграції з зовнішніми даними.

3. Обмеженість налаштувань для створення складних екосистем.

Покращення:

1. Оптимізація використання ресурсів для роботи на слабших ПК.

2. Додавання функції аналізу отриманих даних безпосередньо в системі.

3. Розширення можливостей налаштування параметрів середовища.

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

4.1 Висновки про систему

Розробка інтелектуальної системи симуляції мікроскопічних істот досягла всіх поставлених цілей. Основні результати роботи:

1. Система успішно реалізує можливість моделювання поведінки мікроорганізмів на персональному комп'ютері без значних витрат ресурсів.
2. Вдалося створити простий і зрозумілий користувацький інтерфейс, який забезпечує налаштування параметрів організмів та середовища, а також візуалізацію результатів.
3. Генетичний алгоритм, що лежить в основі симуляції, продемонстрував ефективність у моделюванні еволюційних процесів та адаптації організмів до змінних умов.

Аспекти, які перевершили очікування:

1. Швидкість обчислень: система ефективно працює навіть з великими популяціями, що дозволяє виконувати складні симуляції на стандартних ПК.
2. Стабільність роботи: тривалі симуляції (до 48 годин) проходять без збоїв або помітного падіння продуктивності.
3. Інтуїтивність використання: позитивні відгуки тестувальників підтверджують, що система легко освоюється навіть без попереднього досвіду роботи зі схожими програмами.

4.2 Аналіз обмежень

Незважаючи на успішну реалізацію основного функціоналу, система має кілька обмежень:

1. Високі вимоги до ресурсів при максимальному навантаженні: симуляції з популяціями понад 10 000 організмів потребують значної обчислювальної потужності.

2. Обмежена деталізація організмів: система не враховує внутрішньоклітинні процеси та генетичну різноманітність, що обмежує точність відтворення реальних біологічних систем.

3. Фіксовані умови моделювання: зовнішні параметри задаються вручну користувачем і не можуть автоматично змінюватися за заданими сценаріями.

4.3 Практична цінність

Система має широкий спектр можливостей для застосування у науковій, освітній та практичній діяльності. У наукових дослідженнях вона може використовуватися для моделювання адаптації мікроорганізмів до нових умов середовища, вивчення конкуренції між різними видами мікроорганізмів, а також аналізу впливу мутацій на виживаність та еволюцію популяцій.

В освітньому процесі система стане корисною для ілюстрації біологічних процесів, таких як еволюція та адаптація, під час лекцій і практичних занять. Вона може демонструвати вплив середовищних факторів на популяції мікроорганізмів, а також давати змогу студентам проводити віртуальні експерименти без потреби у лабораторному обладнанні.

У практичній діяльності система дозволяє моделювати ефективність нових антимікробних засобів у віртуальних умовах і прогнозувати еволюційну стійкість мікроорганізмів до хімічних впливів.

Крім того, система має комерційний потенціал і може стати основою для створення навчальних програм для шкіл та університетів або наукових симуляторів для дослідницьких лабораторій.

4.4 Перспективи розвитку

Можливості розширення функціоналу системи:

1. Оптимізація алгоритмів: зменшення навантаження на обчислювальні ресурси для роботи з великими популяціями.

2. Додаткові сценарії: автоматизація зміни середовищних параметрів відповідно до заданих сценаріїв (наприклад, циклічні коливання температури або рівня ресурсів).

3. Покращення візуалізації: створення більш детальної графіки для відображення поведінки організмів і результатів симуляцій.

Обмеження, які потребують вивчення:

1. Дослідження можливостей інтеграції молекулярного моделювання для точнішого відтворення внутрішніх процесів у мікроорганізмах.

2. Розширення адаптивних характеристик системи для моделювання більш складних екосистем.

ВИСНОВОК

Розроблена інтелектуальна система симуляції мікроскопічних істот є інструментом для моделювання біологічних процесів, який об'єднує сучасні алгоритми, простоту використання та ефективність роботи на стандартних персональних комп’ютерах. Система дозволяє моделювати поведінку мікроорганізмів у різноманітних умовах, досліджувати еволюційні процеси, взаємодію популяцій, а також їх адаптацію до змін середовища. Завдяки використанню генетичного алгоритму симуляція досягає високого рівня реалістичності, дозволяючи вивчати динаміку змін характеристик організмів та популяцій без необхідності складного лабораторного обладнання.

Ключовою перевагою системи є її універсальність: вона може бути корисною як у наукових дослідженнях, так і в освітніх та практичних завданнях. У науці її застосування охоплює аналіз конкуренції між мікроорганізмами, моделювання впливу мутацій, дослідження умов, що сприяють виживанню або знищенню популяцій. В освітній сфері система надає студентам і викладачам інструмент для демонстрації біологічних явищ, проведення віртуальних експериментів і практичних завдань. Практичне застосування може включати тестування антимікробних засобів у змодельованих умовах та прогнозування стійкості мікроорганізмів до зовнішніх впливів.

Серед недоліків системи можна виділити обмеження у рівні деталізації процесів, а саме: відсутність моделювання внутрішньоклітинних змін і складної генетичної структури організмів. Крім того, використання ресурсоємних симуляцій може потребувати оптимізації для роботи з великими популяціями. Ці аспекти визначають перспективи для подальших досліджень і вдосконалення, таких як інтеграція додаткових алгоритмів або створення більш складних сценаріїв моделювання.

Загалом, система успішно вирішує поставлені задачі, демонструючи високу ефективність і стабільність. Її розробка закладає основу для подальшого розвитку

в напрямку покращення функціоналу, збільшення масштабності симуляцій та інтеграції інноваційних технологій. Таким чином, інтелектуальна система симуляції мікроскопічних істот має значний потенціал для впровадження у наукову, освітню та практичну діяльність, сприяючи розширенню можливостей вивчення складних біологічних процесів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. "Unity – Game Development Platform" [Електронний ресурс] - Режим доступу до ресурсу: https://unity.com/

2. “Генетичні алгоритми та їх затосування” [Електронний ресурс] - Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Генетичний_алгоритм

3. "Genetic Algorithms in Search, Optimization, and Machine Learning" — David E. Goldberg [Електронний ресурс] - Режим доступу до ресурсу: https://www.amazon.com/Genetic-Algorithms-Optimization-Machine-Learning/dp/0201157675

4. "Artificial Life: A Report from the Frontier Where Computers Meet Biology" — Steven Levy [Електронний ресурс] - Режим доступу до ресурсу: https://www.amazon.com/Artificial-Life-Report-Frontier-Computers/dp/0679751423

5. "Introduction to the Theory of Evolutionary Algorithms" — Thomas Back [Електронний ресурс] - Режим доступу до ресурсу: https://www.springer.com/gp/book/9783642101454

6. "The Role of Artificial Intelligence in Biological Simulations" — Nature Reviews Drug Discovery [Електронний ресурс] - Режим доступу до ресурсу: https://www.nature.com/articles/s41573-019-0006-4

7. "A Survey of Simulation Tools for Microbial Population Dynamics" — Journal of Microbiological Methods [Електронний ресурс] - Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/pii/S016770121630048X

8. "Simulating Evolution in Artificial Life" — IEEE Xplore [Електронний ресурс] - Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/897543

9. "Artificial Life and Evolutionary Computation" — SpringerLink [Електронний ресурс] - Режим доступу до ресурсу: https://link.springer.com/chapter/10.1007/978-3-642-34888-6_11

10. "Modeling Microbial Evolution" — ScienceDirect [Електронний ресурс] - Режим доступу до ресурсу: https://www.sciencedirect.com/topics/biochemistry-genetics-and-molecular-biology/microbial-evolution

11. "Agent-Based Modeling and Simulation of Microorganisms in Ecosystems" — MDPI [Електронний ресурс] - Режим доступу до ресурсу: https://www.mdpi.com/2076-3417/10/11/3815

12. "The Evolution of Computational Biology and Modeling: A Review of Tools and Techniques" — Journal of Computational Biology [Електронний ресурс] - Режим доступу до ресурсу: https://www.liebertpub.com/doi/full/10.1089/cmb.2022.0098

13. "Genetic Algorithm for Optimization in Simulation Modeling" — IEEE Xplore [Електронний ресурс] - Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/9192829

14. "Modeling and Simulation in the Microbial World: An Overview" — SpringerLink [Електронний ресурс] - Режим доступу до ресурсу: https://link.springer.com/article/10.1007/s11356-017-0089-3

15. "Introduction to Genetic Algorithms" — Rajeev M. R [Електронний ресурс] - Режим доступу до ресурсу: https://www.amazon.com/Introduction-Genetic-Algorithms-Computational-Intelligence/dp/0387799675

16. "Evolutionary Algorithms: A Review and Applications" — ScienceDirect [Електронний ресурс] - Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/pii/S2405452616300531

17. "Simulating Biological Evolution Using Agent-Based Models" — Elsevier [Електронний ресурс] - Режим доступу до ресурсу: https://www.elsevier.com/books/simulating-biological-evolution/biological-modeling/978-0-12-814712-6

18. "Artificial Life and Evolutionary Algorithms in Microbial Modeling" — Nature Communications [Електронний ресурс] - Режим доступу до ресурсу: https://www.nature.com/articles/s41598-019-45133-0

19. "Computational Tools for Microbial Population Dynamics Modeling" — MDPI [Електронний ресурс] - Режим доступу до ресурсу: https://www.mdpi.com/2076-3417/9/3/202

20. "Unity for Game Development and Simulation Modeling" — Unity Technologies [Електронний ресурс] - Режим доступу до ресурсу: https://learn.unity.com/

21. "Biological Simulation and Virtual Populations: A Guide to Modern Methods" — Wiley Online Library [Електронний ресурс] - Режим доступу до ресурсу: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118265495

ДОДАТОК А

ЗМІСТ ПУБЛІКАЦІЙ

Сторінок 6

Зміст публікацій з VI всеукраїнської науково-практичної інтернет-конференції студентів і аспірантів «Теоретичні та прикладні аспекти розробки комп’ютерних систем 2024»

УДК 004.42

ІНТЕЛЕКТУАЛЬНА СИСТЕМА СИМУЛЯЦІЇ МІКРОСКОПІЧНИХ ІСТОТ

Замниус А.О., науковий керівник Міловідов Ю.О.

Актуальність теми. Сучасний розвиток технологій у сфері біології та інформаційних технологій відкриває нові можливості для вивчення мікроскопічних організмів. Інтелектуальні системи симуляції можуть стати потужним інструментом для аналізу та дослідження їхньої поведінки та взаємодії з оточуючим середовищем. Зокрема, така система може бути використана для:

- Медичних досліджень: Вивчення впливу мікроскопічних істот на здоров'я людини та розробка нових методів боротьби з патогенними мікроорганізмами.
- Екологічних досліджень: Аналіз взаємодії мікроскопічних істот з оточуючим середовищем для прогнозування змін в екосистемах.
- Наукових досліджень: Дослідження біологічних процесів на найдрібніших рівнях для розуміння принципів життя.
- Освітніх цілей: Використання в навчальних цілях для демонстрації складних біологічних процесів.

Розробка інтелектуальних систем симуляції мікроскопічних істот може стати першим кроком у напрямку створення нових методів дослідження та аналізу біологічних об'єктів. Вона дозволить поєднати знання з біології,

інформаційних технологій та математики для вирішення складних проблем у цих галузях.

Результати досліджень, проведених за допомогою інтелектуальних систем симуляції, можуть мати практичне застосування в медицині, екології, науці та

освіті. Вони можуть допомогти в розробці нових лікарських препаратів, методів боротьби зі збудниками захворювань, а також в управлінні екосистемами.

Отже, тема "Інтелектуальна система симуляції мікроскопічних істот" має великий потенціал для подальших

досліджень та може призвести до важливих відкриттів у сфері біології та інформаційних технологій.

Метою є створення інтелектуальної системи, яка дозволяє симулювати еволюцію мікроскопічних істот за допомогою генетичних алгоритмів. Основна мета - вивчення процесів взаємодії та еволюції в мікросвіті та аналіз адаптації істот до змін у середовищі.

Ключові функції системи:

- Генерація істот: Створення випадкових мікроскопічних істот з властивостями, що можуть піддаватися еволюції.
- Моделювання середовища: Розробка віртуального середовища, де істоти взаємодіють і змагаються за ресурси.
- Генетичний алгоритм: Реалізація механізму генетичного відбору та мутацій для покращення властивостей істот.
- Візуалізація еволюційних процесів: Відображення графічної інформації про еволюцію істот та їх взаємодію.

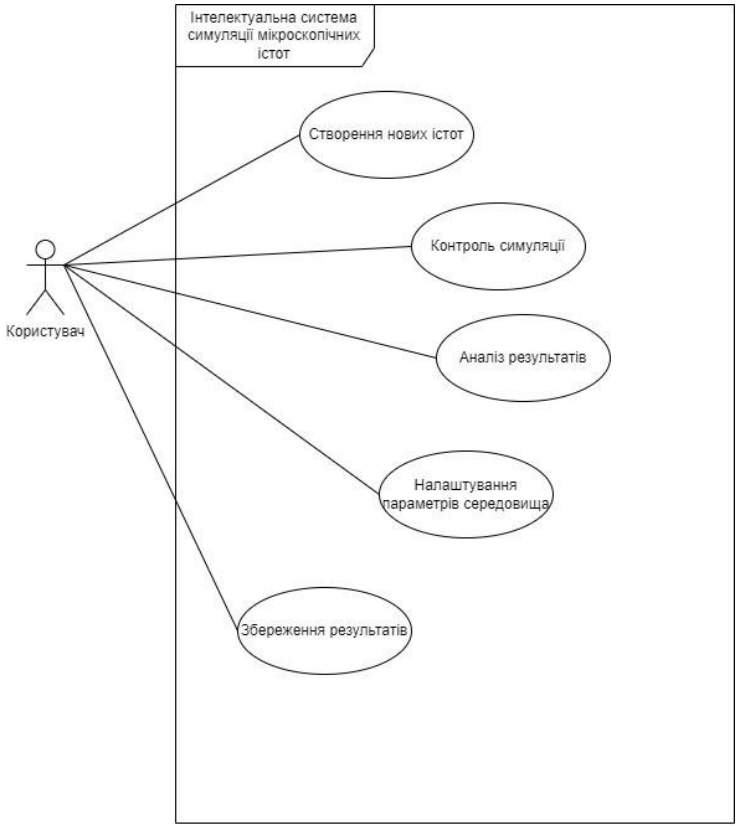

Рисунок 1 Діаграма прецедентів

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Міністерство охорони здоров'я України. (2023). "Сучасні тенденції та перспективи розвитку симуляції мікроскопічних організмів." Публікація МОЗ України No. 23-4567.
2. Інтелектуальна інформаційна система (посилання)
3. Український екологічний фонд. (2022). "Симуляція мікроскопічних організмів для управління довкіллям." Звіт Українського екологічного фонду No. 2022/789.
4. Українська медична академія. (2021). "Симуляція мікроскопічних організмів та її вплив на громадське здоров'я." Технічний звіт УМА, 2021(789).

Національна академія наук України. (2020). "Виклики та можливості симуляції мікроскопічних організмів." EUR 2020/4567

Зміст публікацій з 15-ї Міжнародної науково-практична конференція молодих вчених «Інформаційні технології: економіка, техніка, освіта» за 2024 р.

УДК 004.42

ІНТЕЛЕКТУАЛЬНА СИСТЕМА СИМУЛЯЦІЇ МІКРОСКОПІЧНИХ ІСТОТ

Замниус А. О., старший викладач Міловідов Ю. О.

Актуальність. Дослідження інтелектуальних систем для симуляції мікроскопічних істот ґрунтується на широких можливостях застосування таких технологій у науці, медицині, екології та інших сферах. Створення точних симуляцій дозволяє вивчати поведінку та властивості мікроскопічних організмів, таких як бактерії, віруси та клітини, без потреби в дорогих лабораторних експериментах. Це є особливо важливим у випадках, коли пряме дослідження цих організмів обмежене через складність або небезпеку.

Інтелектуальна система для симуляції мікроскопічних істот дозволить виконувати низку завдань, які значно розширюють можливості сучасних досліджень і технологій, зокрема:

1. **Дослідження поведінки мікроорганізмів** — моделювання руху, взаємодії та поведінкових патернів мікроорганізмів у різних середовищах дозволяє науковцям краще зрозуміти, як вони функціонують, виживають і адаптуються.
2. **Дослідження розповсюдження хвороб** — моделюючи поширення інфекцій на рівні мікроорганізмів, система може допомогти зрозуміти шляхи передачі хвороб та можливі методи їх зупинки, що має значення для епідеміології та боротьби з інфекціями.
3. **Вивчення екосистем** — у контексті навколишнього середовища така система може симулювати взаємодії мікроорганізмів у ґрунті, воді та

інш их екосистемах, що допоможе зрозуміти їх роль у процесах, таких як розкладання органічних речовин, очищення води, утворення ґрунтів.

4. **Навчання та візуалізація для освітніх цілей** — симуляції можуть стати навчальним інструментом для студентів та дослідників, візуалізуючи процеси на мікроскопічному рівні.

Розроблюване програмне забезпечення для інтелектуальної системи симуляції мікроскопічних істот складається з двох частин: клієнтської та серверної.

Дані про поведінку істот та їхню взаємодію з навколишнім середовищем передаються на сервер бази даних для зберігання. Далі система проводить комплексний аналіз цих даних, порівнюючи параметри поведінки мікроорганізмів.

На стороні клієнта користувач може переглянути раніше симульовані мікроорганізми, налаштувати та запустити нову симуляцію.

Структуру програми та взаємозв’язки між її пакетами зображено на діаграмі пакетів (рис 1.):

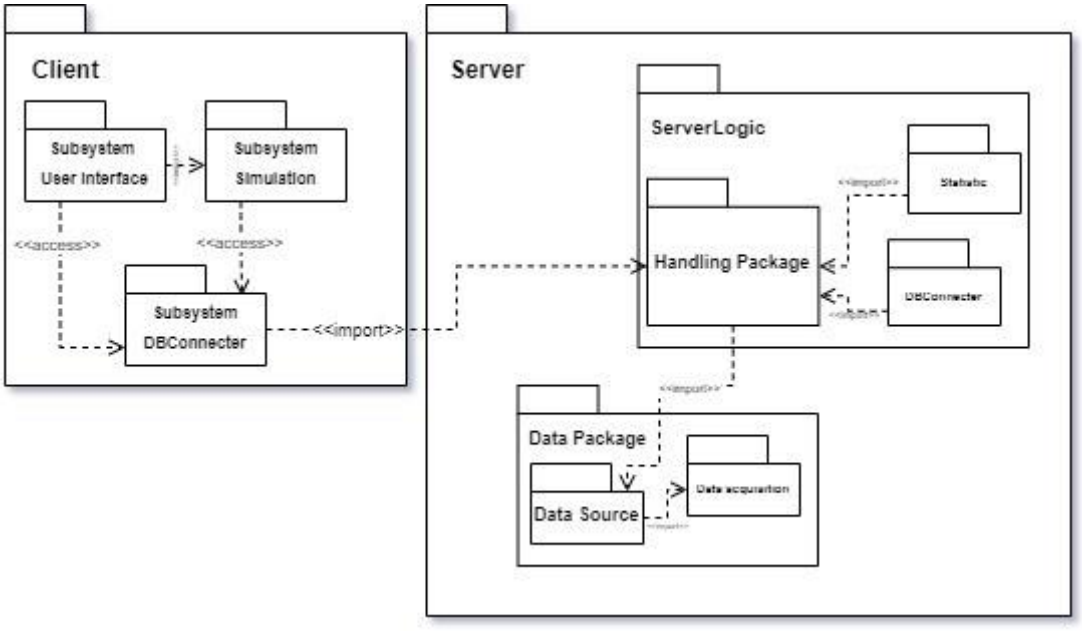

Рис 1. Діаграма пакетів

Сторінка 6

В ході розробки системи використовуються технології та мови програмування, такі як: C#, Unity – для розробки клієнської частини, JavaScript, Playfab – для розробки серверної частини-частини.

Інтерфейс симуляції в програмі (**Рис 2**):

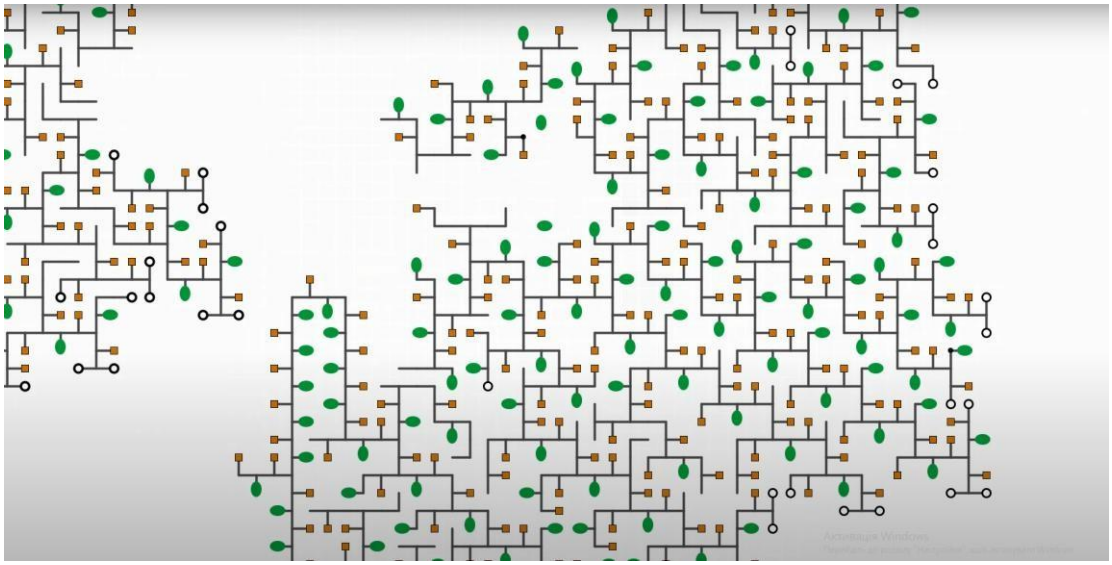

Рис 2. Інтерфейс симуляції в програмі

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. "Генетичні алгоритми та їх застосування"
2. "Genetic Algorithms in Search, Optimization, and Machine Learning" - David E. Goldberg
3. "Introduction to Evolutionary Computing" - Agoston E. Eiben, J.E. Smith
4. Еволюційне моделювання

ДОДАТОК Б

КОД ПРОГРАМИ

Скрипт який моделює базову поведінку організму

```
using UnityEngine;

public class Organism : MonoBehaviour
{
    public float speed = 2.0f;
    public float health = 100.0f;
    public float energy = 50.0f;
    public Vector3 direction;

    void Start()
    {
        RandomizeDirection();
    }

    void Update()
    {
        Move();
        ConsumeEnergy();
        CheckHealth();
    }

    void RandomizeDirection()
    {
        direction = new Vector3(Random.Range(-1f, 1f), 0, Random.Range(-1f, 1f)).normalized;
    }

    void Move()
    {
        transform.Translate(direction * speed * Time.deltaTime);
        if (Random.Range(0, 100) < 5) RandomizeDirection();
    }

    void ConsumeEnergy()
    {
        energy -= Time.deltaTime * 1.0f;
        if (energy <= 0)
        {
            health -= Time.deltaTime * 5.0f;
            energy = 0;
        }
    }

    void CheckHealth()
    {
        if (health <= 0)
        {
            Die();
        }
    }

    void Die()
    {
```

```
Destroy(gameObject);
    }

    void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.CompareTag("Food"))
        {
            ConsumeFood(collision.gameObject);
        }
    }

    void ConsumeFood(GameObject food)
    {
        energy += 20.0f;
        if (energy > 100) energy = 100;
        Destroy(food);
    }
}
```

Скрипт який відповідає за генетичний код організмів

```
using UnityEngine;

public class Genetics : MonoBehaviour
{
    public string dna; // Генетичний код
    public float mutationRate = 0.05f;

    void Start()
    {
        GenerateDNA();
    }

    void GenerateDNA()
    {
        dna = "";
        for (int i = 0; i < 10; i++)
        {
            dna += Random.Range(0, 2) == 0 ? "A" : "T";
        }
    }

    public string MutateDNA()
    {
        char[] newDNA = dna.ToCharArray();
        for (int i = 0; i < dna.Length; i++)
        {
            if (Random.value < mutationRate)
            {
                newDNA[i] = newDNA[i] == 'A' ? 'T' : 'A';
            }
        }
        return new string(newDNA);
    }

    public void CombineDNA(string parent1, string parent2)
    {
        dna = "";
        for (int i = 0; i < parent1.Length; i++)
        {
            dna += Random.value > 0.5f ? parent1[i] : parent2[i];
        }
        dna = MutateDNA();
    }

    void PrintDNA()
    {
        Debug.Log("DNA: " + dna);
    }
}
```

Скрипт який відповідає за створення та взаємодію з їжею

```
using UnityEngine;

public class Food : MonoBehaviour
{
    public float nutritionValue = 20.0f;

    void Start()
    {
        RandomizePosition();
    }

    void RandomizePosition()
    {
        transform.position = new Vector3(Random.Range(-10f, 10f), 0, Random.Range(-10f, 10f));
    }

    void Update()
    {
        // Опціональна механіка: зникає через певний час
        if (Time.timeSinceLevelLoad > 60)
        {
            Destroy(gameObject);
        }
    }

    void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.CompareTag("Organism"))
        {
            collision.gameObject.GetComponent<Organism>().energy += nutritionValue;
            Destroy(gameObject);
        }
    }
}
```

Скрипт який відповідає за контроль параметрів середовища

```
using UnityEngine;

public class Environment : MonoBehaviour
{
    public float temperature = 20.0f; // °C
    public float resourceAvailability = 1.0f; // Від 0 до 1

    void Update()
    {
        SimulateDayCycle();
        SimulateResourceChange();
    }

    void SimulateDayCycle()
    {
        temperature = 15.0f + Mathf.Sin(Time.time * 0.1f) * 10.0f;
    }

    void SimulateResourceChange()
    {
        resourceAvailability = Mathf.Clamp01(Mathf.Sin(Time.time * 0.05f) * 0.5f + 0.5f);
    }
}
```

Скрипт який керує популяцією організмів

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

public class PopulationManager : MonoBehaviour
   {
      public GameObject organismPrefab;
      public int initialPopulation = 20;

      private List<GameObject> population;

      void Start()
      {
         InitializePopulation();
      }

      void InitializePopulation()
      {
         population = new List<GameObject>();
         for (int i = 0; i < initialPopulation; i++)
         {
            SpawnOrganism();
         }
      }

      void SpawnOrganism()
      {
         Vector3 spawnPosition = new Vector3(Random.Range(-10f, 10f), 0, Random.Range(-10f, 10f));
         GameObject organism = Instantiate(organismPrefab, spawnPosition, Quaternion.identity);
         population.Add(organism);
      }

      void Update()
      {
         ManagePopulation();
      }

      void ManagePopulation()
      {
         if (population.Count < initialPopulation)
         {
            SpawnOrganism();
         }
      }
   }
```

Скрипт який реалізує природній відбір в симуляції

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

public class Selection : MonoBehaviour
   {
      public GameObject organismPrefab;
      public int maxPopulation = 50;
      public float survivalRate = 0.5f; // Відсоток, що виживає

      private List<GameObject> population = new List<GameObject>();

      void Start()
      {
         InitializePopulation(20); // Початкова кількість
      }

      void Update()
      {
         if (Time.timeSinceLevelLoad % 10 < 0.1f) // Кожні 10 секунд
         {
            PerformSelection();
         }
      }

      void InitializePopulation(int count)
      {
         for (int i = 0; i < count; i++)
         {
            SpawnOrganism();
         }
      }

      void SpawnOrganism()
      {
         Vector3 spawnPosition = new Vector3(Random.Range(-10f, 10f), 0, Random.Range(-10f, 10f));
         GameObject organism = Instantiate(organismPrefab, spawnPosition, Quaternion.identity);
         organism.GetComponent<Organism>().health = Random.Range(50f, 100f);
         population.Add(organism);
      }

      void PerformSelection()
      {
         List<GameObject> survivors = new List<GameObject>();

         foreach (var organism in population)
         {
            if (Random.value < survivalRate)
            {
               survivors.Add(organism);
            }
            Else
```

```
            {
                Destroy(organism);
            }
        }

        population = survivors;

        int newOrganisms = maxPopulation - population.Count;
        for (int i = 0; i < newOrganisms; i++)
        {
            SpawnOrganism();
        }
    }
}
```

Скрипт який реалізує зміни параметрів організму в залежності від середовища

```
using UnityEngine;

public class Adaptation : MonoBehaviour
{
    public float baseSpeed = 2.0f;
    public float baseHealth = 100.0f;
    public float temperatureResistance = 0.5f; // Від 0 до 1
    public float resourceEfficiency = 1.0f;

    private Environment environment;

    void Start()
    {
        environment = FindObjectOfType<Environment>();
        RandomizeAdaptation();
    }

    void RandomizeAdaptation()
    {
        temperatureResistance = Random.Range(0.2f, 1.0f);
        resourceEfficiency = Random.Range(0.5f, 2.0f);
    }

    void Update()
    {
        AdjustParameters();
    }

    void AdjustParameters()
    {
        float temperatureFactor = Mathf.Abs(environment.temperature - 20f) * (1 - temperatureResistance);
        float resourceFactor = 1 - environment.resourceAvailability;

        baseSpeed = Mathf.Clamp(2.0f - temperatureFactor * 0.1f, 0.5f, 3.0f);
        baseHealth = Mathf.Clamp(100.0f - resourceFactor * 10.0f, 50.0f, 100.0f);
    }
}
```

Скрипт який дозволяє розмножуватися організмам

```
using UnityEngine;

public class Replication : MonoBehaviour
{
    public GameObject organismPrefab;
    public float replicationCooldown = 30.0f; // Час між розмноженнями
    public float energyRequired = 50.0f;

    private float timeSinceLastReplication = 0.0f;

    void Update()
    {
        timeSinceLastReplication += Time.deltaTime;

        if (timeSinceLastReplication >= replicationCooldown && GetComponent<Organism>().energy >= energyRequired)
        {
            Replicate();
            timeSinceLastReplication = 0.0f;
        }
    }

    void Replicate()
    {
        GameObject offspring = Instantiate(organismPrefab, transform.position + new Vector3(1f, 0, 1f), Quaternion.identity);
        Organism parent = GetComponent<Organism>();
        Organism child = offspring.GetComponent<Organism>();

        child.speed = parent.speed * Random.Range(0.9f, 1.1f);
        child.health = parent.health * Random.Range(0.8f, 1.0f);
        child.energy = 25.0f; // Початкова енергія
    }
}
```

Скрипт який відповідає за виведення статистики симуляції

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

public class Statistics : MonoBehaviour
   {
      public List<GameObject> population;
      public int totalReplications = 0;
      public int totalDeaths = 0;

      void Start()
      {
         population = new List<GameObject>();
      }

      void Update()
      {
         UpdateStatistics();
         DisplayStatistics();
      }

      void UpdateStatistics()
      {
         population.Clear();
         foreach (GameObject organism in GameObject.FindGameObjectsWithTag("Organism"))
         {
            population.Add(organism);
         }
      }

      void DisplayStatistics()
      {
         Debug.Log($"Population: {population.Count}");
         Debug.Log($"Total Replications: {totalReplications}");
         Debug.Log($"Total Deaths: {totalDeaths}");
      }

      public void IncrementReplication()
      {
         totalReplications++;
      }

      public void IncrementDeath()
      {
         totalDeaths++;
      }
   }
```

Скрипт візуалізації данних

```
   using UnityEngine;
using UnityEngine.UI;

public class GraphVisualizer : MonoBehaviour
   {
      public LineRenderer populationGraph;
      public Text populationCountText;

      private float timeStep = 1.0f;
      private float elapsedTime = 0.0f;
      private int currentPopulation = 0;

      void Start()
      {
         InitializeGraph();
      }

      void Update()
      {
         elapsedTime += Time.deltaTime;

         if (elapsedTime >= timeStep)
         {
            UpdateGraph();
            elapsedTime = 0.0f;
         }
      }

      void InitializeGraph()
      {
         populationGraph.positionCount = 0;
      }

      void UpdateGraph()
      {
         currentPopulation = GameObject.FindGameObjectsWithTag("Organism").Length;
         populationCountText.text = $"Population: {currentPopulation}";

         Vector3 newPoint = new Vector3(Time.timeSinceLevelLoad, currentPopulation, 0);
         populationGraph.positionCount++;
         populationGraph.SetPosition(populationGraph.positionCount - 1, newPoint);
      }
   }
```