Міністерство освіти і науки України

Чернівецький національний університет імені Юрія Федьковича

Факультет математики та інформатики

Кафедра прикладної математики

Аналіз алгоритмів тріангуляції полігонів і точкових множин

Курсова робота
Рівень вищої освіти – перший (бакалаврський)

Виконав:
студент 3 курсу, 312 групи
Мацьопа Кирило Ростиславович

Керівник:
кандидат фізико-математичних наук,
доцент кафедри прикладної математики
Маценко Василь Григорович

Чернівці – 2025

АНОТАЦІЯ

Курсова робота зосереджена на детальному аналізі алгоритмів тріангуляції полігонів та множин точок. У ній розглянуто ключові поняття та види тріангуляції, включаючи тріангуляцію Делоне, інкрементні методи тріангуляції та підхід на основі динамічного програмування. Окрему увагу приділено практичним аспектам застосування тріангуляції в таких сферах, як комп'ютерна графіка, геоінформаційні системи та обробка даних із просторовою структурою. У роботі також представлено програмну реалізацію алгоритмів тріангуляції на мові Python із застосуванням відповідних бібліотек. Проведено експериментальне дослідження і порівняння ефективності різних підходів до тріангуляції. Отримані результати можна використовувати для розробки обчислювальних моделей, що підвищують ефективність вирішення прикладних задач в області обробки геометричних структур.

ЗМІСТ

ВСТУП

Сучасні інформаційні технології вимагають ефективної обробки та представлення геометричних даних. У таких сферах, як комп’ютерна графіка, геоінформаційні системи (ГІС), обробка зображень та інші прикладні області, часто виникає потреба в поділі простору або об'єкта на прості геометричні компоненти. Найпопулярнішим методом такого поділу є тріангуляція — процес розбиття багатокутників чи множин точок на систему неперетинних трикутників, які повністю охоплюють задану область. Тріангуляція значною мірою спрощує складні обчислення, сприяє оптимізації моделювання поверхонь, створенню сіток для чисельних методів, покращує візуалізацію та забезпечує вищу точність розрахунків. Ефективність цього процесу залежить від застосованого алгоритму. Одні алгоритми гарантують оптимальність за певними показниками, наприклад, максимізацію мінімального кута, тоді як інші орієнтовані на високошвидкісну обробку великих обсягів даних. У рамках цієї курсової роботи проведено аналіз найпоширеніших алгоритмів тріангуляції, їх переваг та недоліків, а також вивчено можливості застосування в різних практичних задачах. Особливу увагу приділено тріангуляції множин точок і полігонів, які є найбільш уживаними моделями в прикладній геометрії.

Мета роботи. Дослідити та проаналізувати наявні алгоритми тріангуляції для полігонів і точкових множин, оцінити їх продуктивність і практичну цінність, а також реалізувати окремі з них за допомогою мови програмування Python.

Завдання дослідження. Вивчити теоретичні основи процесу тріангуляції. Ознайомитися з алгоритмами, такими як тріангуляція Делоне, інкрементний метод і динамічне програмування. Реалізувати ці алгоритми за допомогою Python. Провести експериментальний аналіз щодо їхньої продуктивності та візуалізації. Зробити висновки про доречність використання кожного алгоритму.

Об’єкт дослідження. Процеси тріангуляції у контексті задач комп’ютерної геометрії. Предмет дослідження Алгоритми тріангуляції та їх практичне використання у роботі з полігонами і множинами точок.

Методи дослідження. Застосування аналізу наукової літератури, програмного моделювання, візуалізації геометричних об’єктів, а також порівняльного аналізу ефективності алгоритмів.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ТРІАНГУЛЯЦІЇ

У цьому розділі аналізуються ключові теоретичні аспекти тріангуляції, її види, особливості та практичне застосування. Розуміння цих понять є обов’язковою передумовою для глибшого дослідження алгоритмів, які розглядатимуться в подальших розділах.

1.1. Поняття тріангуляції.

Тріангуляція — це процес поділу геометричної фігури або множини точок на набір трикутників, які не перетинаються між собою та повністю покривають задану область без пропусків чи накладань. У геометричному й прикладному вимірі вона зменшує складність роботи з геометричними об'єктами, забезпечуючи ефективні інструменти для візуалізації, числових розрахунків і моделювання. У двовимірному просторі тріангуляція множини точок передбачає побудову такого набору трикутників, що усі вершини трикутників відповідають точкам із заданої множини жодні два трикутники не мають точок перетину всередині сукупність усіх трикутників повністю покриває опуклу оболонку множини точок.
Тріангуляції виступають базовими структурами обчислювальної геометрії, знаходячи застосування в таких завданнях, як створення та обробка тривимірних моделей у сфері 3D-моделювання аналіз просторових даних у інформаційних системах (ГІС) проектування сіток для чисельного моделювання візуалізація поверхонь у наукових, медичних і технічних галузях. Залежно від поставленої мети застосовуються різні підходи до тріангуляції, кожен із яких має власні характеристики, переваги й обмеження. У наступних підрозділах буде детально розглянуто класифікацію методів тріангуляції та їхні ключові властивості.

1.2. Типи тріангуляцій

Тріангуляції класифікуються за низкою параметрів залежно від поставлених завдань, структури вихідних даних та критеріїв якості побудови. Серед основних видів виділяють тріангуляцію Делоне, обмежену тріангуляцію, регулярну тріангуляцію, а також спеціалізовані методи для специфічних потреб.
Тріангуляція Делоне є одним із найпоширеніших методів у прикладній геометрії. Вона будується так, щоб жодна з точок набору не розташовувалася всередині описаного кола будь-якого трикутника. Такий підхід допомагає уникнути утворення надто вузьких трикутників і забезпечує більш рівномірний розподіл. Тріангуляція Делоне знаходить застосування у візуалізації поверхонь, моделюванні та в геоінформаційних системах.
Обмежена тріангуляція Делоне враховує наявність заданих меж або внутрішніх контурів об'єктів. Вона незамінна в задачах, де необхідно зберегти точну геометрію вихідного об'єкта, наприклад, при моделюванні рельєфу чи створенні архітектурних моделей. Регулярна тріангуляція враховує не лише координати точок, але й їх вагові параметри.

Це дозволяє моделювати більш складні просторові структури, де важливі як форма об'єктів, так і їх властивості, що визначаються положенням у просторі. Існують і інші види тріангуляцій, серед яких ієрархічна, адаптивна та конформна. Вони застосовуються в задачах зі специфічними вимогами — контроль деталізації, врахування складної топології або адаптація точності до масштабу. Отже, вибір конкретного типу тріангуляції залежить від різних факторів особливостей вихідних даних, специфіки завдання, необхідної точності, швидкості обчислень та вимог до відповідності геометрії.

1.3. Властивості тріангуляцій

Якісна тріангуляція повинна відповідати певному набору властивостей, які визначають її цінність, точність і ефективність у різних прикладних задачах. Ці характеристики є основоположними у створенні сіток, моделюванні фізичних процесів, розробці геометричних форм та інших застосуваннях. Основними серед таких властивостей є

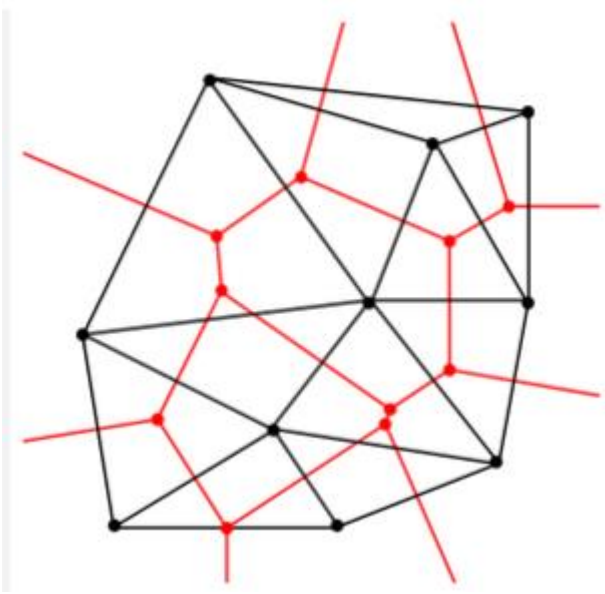
1. Повнота покриття. Тріангуляція зобов’язана повністю охоплювати задану область або множину точок, уникаючи пропущених фрагментів чи порожніх зон у межах геометричної форми. У багатокутниках це означає, що всередині фігури не залишаються невраховані частини. Недостатнє покриття в задачах 3D-моделювання або обробки рельєфу викликає помилки розрахунків і призводить до некоректного відображення поверхні.
2. Неперетинність. Усі трикутники тріангуляції мають бути неперетинними, тобто не можуть мати спільних внутрішніх точок. Дозволяється поєднання через спільні вершини або ребра, але перетинання внутрішніх областей заборонене. Це забезпечує правильну топологію та унеможливлює неоднозначність чи похибки під час подальшої обробки даних.
3. Мінімізація тонких трикутників. Наявність трикутників із надмірно малими внутрішніми кутами («вузьких» або «голчастих») може призвести до дестабілізації при чисельних розрахунках, таких як скінченно-елементний аналіз або обчислення градієнтів. Тому важливо максимізувати мінімальний кут трикутників. Ідеальним варіантом є кути близько 60°, що сприяє рівномірності й надійності обчислень. Тріангуляція Делоне є прикладом моделі, що реалізує цей принцип.
4. Структурна регулярність. Розміри трикутників у межах тріангуляції мають бути порівняними, без раптових змін масштабу. Іншими словами, розбиття повинно бути рівномірним, без появи різко великих або дрібних трикутників у близькому сусідстві. Така регулярність критично важлива для завдань візуалізації, симуляцій і обробки зображень, де великі коливання в масштабах можуть спотворити результат або вплинути на точність.
5. Локальність обчислень. Ця властивість полягає у впливі змін лише на локальну частину тріангуляції при додаванні нових точок до множини. Вона виключає необхідність перерахунку всієї тріангуляції. Локальність ключова для динамічних задач, де дані змінюються в реальному часі, наприклад.

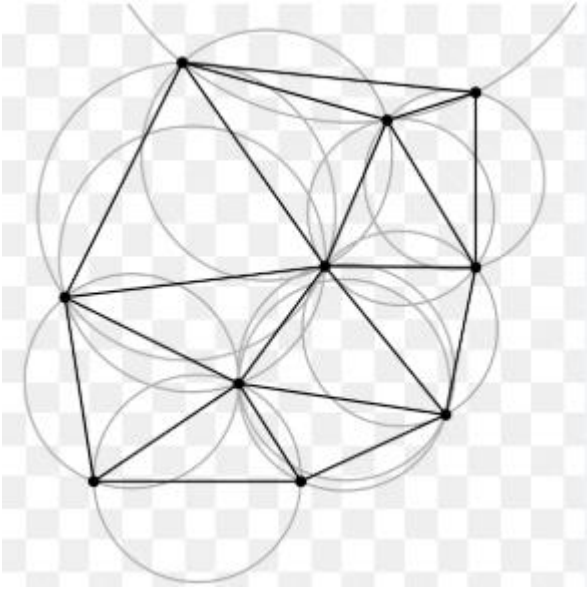
РОЗДІЛ 2. АНАЛІЗ АЛГОРИТМІВ ТРІАНГУЛЯЦІЇ

Методи тріангуляції є фундаментальним інструментом для багатьох задач, що передбачають роботу з геометричними даними. Їх застосовують у таких сферах, як комп'ютерна графіка, 3D-моделювання, геоінформаційні системи, чисельні методи та фізичне моделювання. Ефективність алгоритму тріангуляції визначається як якістю створених трикутників, так і швидкістю та стабільністю обчислень. У цьому розділі представлено найвідоміші та практичні методики тріангуляції, аналізуються їхні принципи роботи, переваги й обмеження.

2.1. Алгоритм Делоне

Він є одним із найбільш широко застосовуваних методів для тріангуляції наборів точок. Його основна концепція базується на дотриманні умови Делоне: жодна точка не повинна розміщуватися всередині описаного кола будь-якого з отриманих трикутників. Такий підхід забезпечує геометричну стабільність структури, запобігає утворенню дуже вузьких трикутників і гарантує високу якість побудованої сітки. Побудова цієї тріангуляції основана на діаграмі Вороного, яка використовується в якості подвійної структури. Кожен трикутник Делоне відповідає трьом сусіднім коміркам діаграми Вороного. У підсумку виходить рівномірна сітка, що максимально рівно заповнює простір. Серед переваг алгоритму Делоне. Висока якість сітки завдяки великому мінімальному куту трикутників. Відносно висока швидкість роботи та хороша масштабованість. Стійкість до не точностей у позиціонуванні точок. Наявність численних оптимізованих реалізацій у популярних мовах програмування (Python, C++, MATLAB тощо). Проте є й певні обмеження. Не враховує геометричних обмежень, таких як крайові межі полігонів воно ускладнюється при роботі з неопуклими багатокутниками — для таких випадків потрібна додаткова обробка. Для динамічних змін необхідно повторно перераховувати локальні структури. Часова складність у двовимірному просторі становить O(n log n), де n — кількість точок у множині.





2.2. Інкрементний алгоритм тріангуляції.

Інкрементний алгоритм тріангуляції, також відомий як метод покрокового додавання точок, базується на поступовому включенні нових точок у вже існуючу тріангуляцію. На кожному етапі відбувається локальна модифікація структури з урахуванням доданої точки, що дозволяє уникнути необхідності повного перерахунку сітки. Робота алгоритму розпочинається зі створення стартового базового трикутника, який охоплює всі інші точки всередині або на його межах. Далі точки додаються однією за одною зі списку. Після включення чергової точки проводиться локальна перебудова, видаляються трикутники, в які потрапляє нова точка, а на їхньому місці будуються нові трикутники, що з'єднують цю точку з краєм утвореного контуру.

Цей підхід є особливо корисним у випадках динамічного оновлення геометричної структури, коли дані поступово змінюються або надходять у реальному часі. Переваги інкрементного алгоритму, простота реалізації, зручність для поетапного оновлення сітки. Можливість адаптації для роботи з тривимірною тріангуляцією. Забезпечення можливості оптимізації локальних структур. Недоліки, залежність якості тріангуляції від порядку додавання точок. У певних ситуаціях виникає необхідність рекурсивного оновлення значної частини структури. Потреба у додаткових кроках (наприклад, виконання фліпів ребер) для забезпечення Делоне-тріангуляції. Часова складність: У гіршому випадку алгоритм працює за $O(n^2)$, проте при оптимальному додаванні точок і врахуванні локальних змін середня складність складає $O(n \log n)$. Приклад застосування, алгоритм широко використовується в ігрових рушіях і фізичних симуляціях, де об'єкти змінюють своє положення або додаються в реальному часі. Також він знаходить застосування у створенні навігаційних сіток у двовимірних і тривимірних середовищах.

2.3. Алгоритм динамічного програмування

Алгоритм динамічного програмування є одним із найбільш поширених підходів до тріангуляції опуклих багатокутників. Головним завданням цього методу є визначення такого розбиття багатокутника на трикутники, яке забезпечує мінімізацію заданої функції вартості. Цими критеріями можуть бути сума довжин внутрішніх діагоналей, площа або периметр отриманих трикутників. Основний принцип функціонування алгоритму ґрунтується на розбитті вихідної задачі на менші, взаємозалежні задачі. Кожна з задач вирішується незалежно, після чого їх результати інтегруються для отримання загального розв’язку. Для кожної можливої пари вершин багатокутника визначається мінімальна вартість тріангуляції області між ними. Процес обчислення реалізується рекурсивно і ґрунтується на використанні спеціальної матриці для зберігання проміжних результатів, що дозволяє уникнути багаторазового повторення однакових обчислень. Оскільки структура задач відповідає фіксованим межам і значно дублюється, використання динамічного програмування дає змогу суттєво зменшити обчислювальну складність порівняно з методами прямого перебору.

Переваги алгоритму , забезпечує глобально оптимальний результат. Ефективно працює для задач із чітко визначеною функцією вартості. Особливо придатний для тріангуляції опуклих багатокутників.

Обмеження та недоліки, використовується виключно для опуклих багатокутників; у випадку неопуклих фігур потрібне додаткове попереднє розбиття.Має високу вимогу до обсягу пам’яті при великій кількості вершин.

Тріангуляція полігонів, особливо **неопуклих**, вимагає попереднього розбиття фігури на опуклі частини. Це досягається за допомогою алгоритмів декомпозиції, після чого до кожної частини застосовується стандартна тріангуляція, зокрема метод динамічного програмування або Делоне..

Часова складність, обчислювальна складність алгоритму зазвичай оцінюється як $O(n^3)$, де n — число вершин багатокутника. У випадках певних модифікацій та оптимізацій можливо досягти покращеного значення $O(n^2)$, залежно від специфіки функції вартості.

Приклади застосування Алгоритм знаходить широке використання у різноманітних галузях, у комп'ютерній графіці для розбиття поверхонь об'єктів на трикутники з оптимальними характеристиками, у компіляторах для аналізу та обробки синтаксичних дерев.

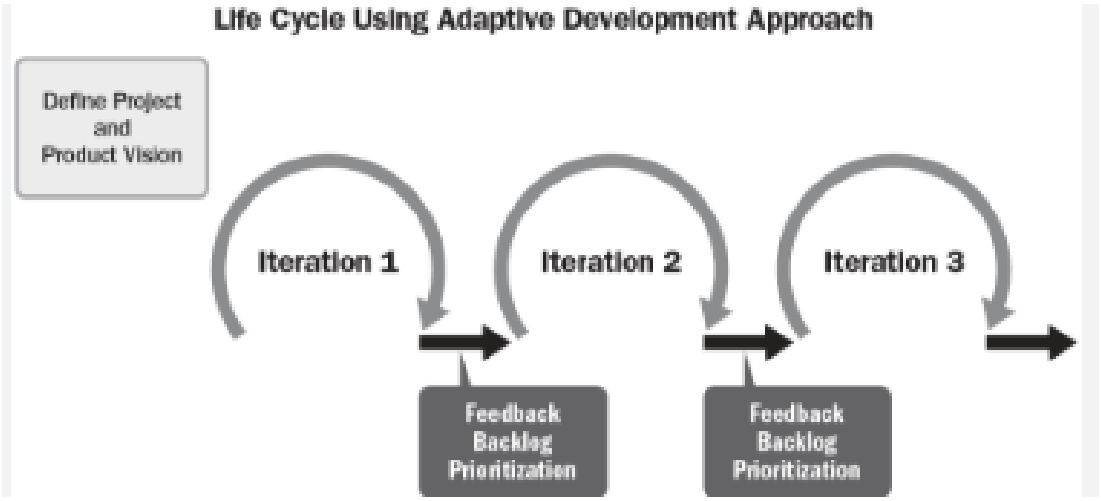
2.4. Порівняльна характеристика алгоритмів тріангуляції.

Аналіз різних алгоритмів тріангуляції дозволяє вибрати найбільш оптимальний метод залежно від особливостей завдання.
Кожен з підходів має свої унікальні переваги та обмеження, пов'язані з продуктивністю, рівнем точності, вимогами до структури даних і можливостями для подальшої обробки результатів. Нижче представлені основні характеристики трьох розглянутих алгоритмів у вигляді узагальненої порівняльної таблиці:

Критерій	**Тріангуляція Делоне**	**Інкрементний метод**	**Динамічне програмування**
Тип вхідних даних	Множина точок	Множина точок	Опуклий багатокутник
Якість трикутників	Висока	Середня	Оптимальна (залежить від мети)
Часова складність	O(n log n)	O(n log n) – $O(n^2)$	$O(n^3)$
Простота реалізації	Середня	Висока	Середня/висока
Підтримка динаміки	Середня	Висока	Низька
Підтримка обмежень	Немає	Частково	Немає
Застосування	3D-графіка, ГІС	Ігри, фізика, 2D-сцени	Оптимізація, моделювання

З аналізу таблиці видно, що алгоритм Делоне забезпечує найкращий баланс між якістю отриманих результатів і швидкодією, завдяки чому є універсальним вибором для більшості завдань. Інкрементний метод виділяється своєю гнучкістю, що особливо важливо при динамічній зміні вхідних даних. Алгоритм динамічного програмування, хоча й більш обмежений у використанні, гарантує глобальну оптимізацію і найбільш підходить для задач з високо структурованими даними, де критична висока точність. Обираючи алгоритм тріангуляції, слід враховувати такі аспекти, як тип вхідних даних, вимоги до продуктивності, необхідний рівень точності, можливі часові обмеження й специфічні вимоги до подальшого використання отриманих результатів.

Інкрементний метод



ПРАКТИЧНА РЕАЛІЗАЦІЯ АЛГОРИТМІВ ТРІАНГУЛЯЦІЇ.

У цьому розділі розглянуто практичну реалізацію алгоритмів тріангуляції множин точок із використанням мови програмування Python. Для побудови тріангуляції було залучено бібліотеки scipy.spatial, matplotlib і numpy, які забезпечують роботу з геометричними об'єктами та надають інструменти для візуалізації створених сіток. Основний акцент зроблено на тріангуляцію Делоне, яка вважається однією з найефективніших для роботи з множинами точок. У межах розробки створено програму, що генерує випадкову множину точок, будує тріангуляцію та демонструє результат у графічній формі.

Постановка задачі
Метою даного розділу є реалізація базового алгоритму тріангуляції Делоне для множини точок на площині. Передбачено також візуалізацію отриманих результатів та короткий аналіз якості створеної сітки.

Інструменти та середовище розробки
У процесі розробки використано мову програмування Python (версія 3.9+), а також наступні бібліотеки, NumPy — для генерації випадкових координат точок; SciPy (scipy.spatial.Delaunay) — для створення тріангуляції Делоне, Matplotlib — для графічного відображення побудованої сітки.

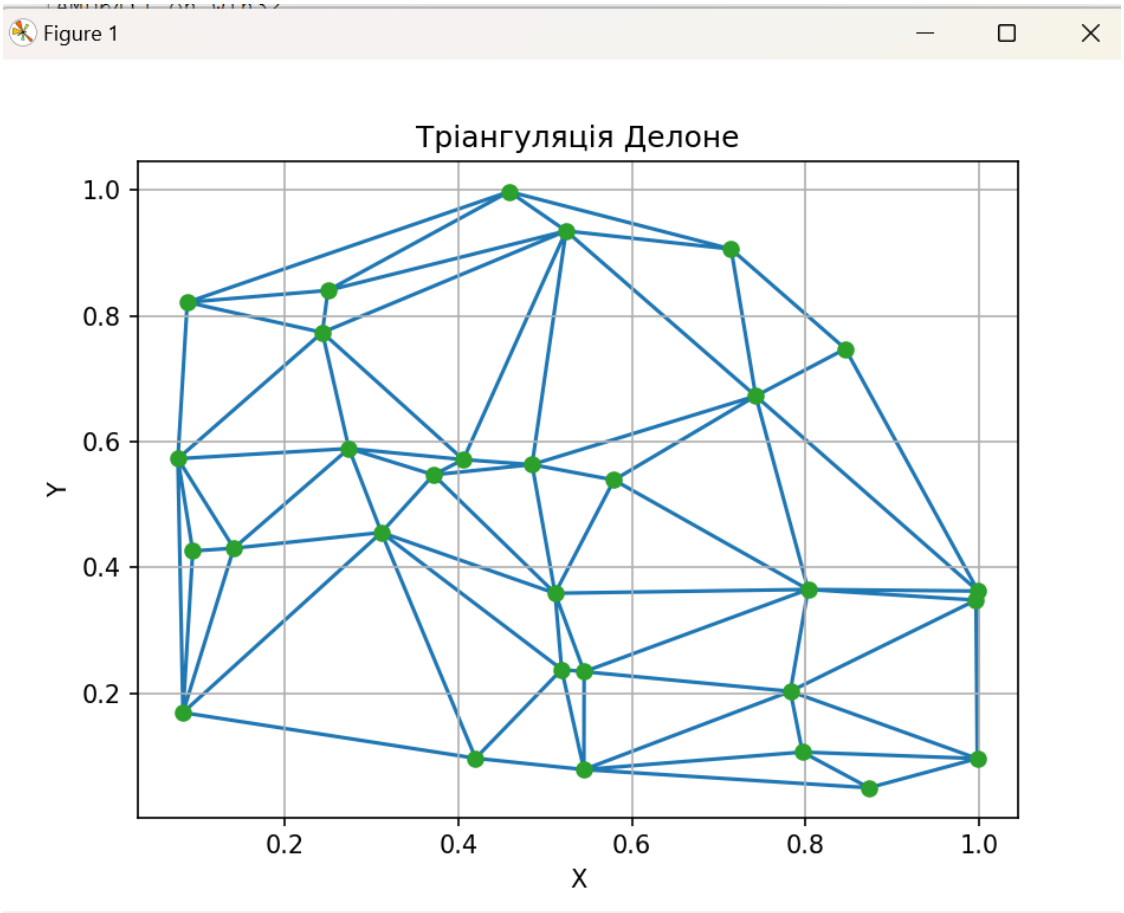
Код Тріангуляції Делоне

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial import Delaunay

# Генеруємо випадкову множину точок
points = np.random.rand(30, 2)

# Побудова тріангуляції Делоне
tri = Delaunay(points)

# Візуалізація
plt.triplot(points[:, 0], points[:, 1], tri.simplices)
plt.plot(points[:, 0], points[:, 1], 'o')
plt.title('Тріангуляція Делоне')
plt.xlabel('X')
plt.ylabel('Y')
plt.grid(True)
plt.show()
```



Аналіз результатів

На побудованому графіку показано, що множина випадково згенерованих точок була успішно розбита на трикутники згідно з принципом Делоне. Усі утворені трикутники демонструють прийнятну геометрію, оскільки не містять надмірно гострих або видовжених кутів. Це підтверджує основну теоретичну властивість алгоритму — здатність мінімізувати кількість вузьких трикутників під час тріангуляції. Застосована програма демонструє високу швидкість обчислень навіть із зростанням кількості точок, що свідчить про її ефективну реалізацію. У перспективі можливе розширення функціональних можливостей програми через впровадження альтернативних підходів до тріангуляції, наприклад, із врахуванням меж полігону. Також доцільним є розроблення користувацького інтерфейсу для зручнішого та інтерактивного використання

ДОДАТОК

ЗАДАЧА 1.
Розробка тріангуляції Делоне для множини заданих точок у двовимірному просторі.

Формулювання задачі.
Необхідно провести тріангуляцію Делоне для десяти точок із заданими координатами, розташованих у двовимірному просторі. Результат повинен бути отриманий шляхом ручного розрахунку або використання засобів програмування, зокрема бібліотеки `scipy.spatial` в середовищі Python. Додатково потрібно обґрунтувати правильність отриманого рішення.

Мета дослідження.
Ця задача має на меті продемонструвати глибоке розуміння принципів тріангуляції Делоне, що включає побудову трикутників з максимальним мінімальним кутом, а також практичне опрацювання інструментів обчислювальної геометрії, таких як бібліотека `scipy.spatial`.

Алгоритм виконання.
1. Підготовчий етап.Визначте координати десяти точок у просторі. Заздалегідь можна перевірити, чи точки не утворюють виродженої конфігурації (наприклад, коли три або більше точки лежать на одній прямій).

2. Побудова тріангуляції вручну. Використовуючи геометричний підхід, розрахуйте всі можливі трикутники між заданими точками. Перевірте виконання критерію Делоне, описане коло для кожного трикутника не повинно містити жодної іншої точки множини. В результаті збережіть тільки ті трикутники, які відповідають умовам тріангуляції Делоне.

3.Автоматизований метод з використанням Python. Імпортуйте необхідні модулі із бібліотеки `scipy.spatial`, зокрема об’єкт `Delaunay`. Передайте координати заданих точок до алгоритму для побудови тріангуляції. Збережіть отриманий результат, включаючи вершини трикутників і зв’язки між ними.

4.Обґрунтування правильності результату. Для ручного методу важливо навести геометричні доказові аргументи відповідності тріангуляції Делоне. У випадку автоматизованого методу необхідно посилатися на гарантії точності алгоритму, вбудованого в бібліотеку `scipy.spatial`.

Очікувані результати.
Отримане рішення повинно складатися з набору трикутників, які утворюють правильну тріангуляцію множини точок із дотриманням принципу Делоне. Інтуїтивно зрозумілі графічні візуалізації, створені за допомогою Python (наприклад, через бібліотеки `matplotlib`), можуть слугувати підтвердженням коректності побудови та покращувати наочність представлених даних

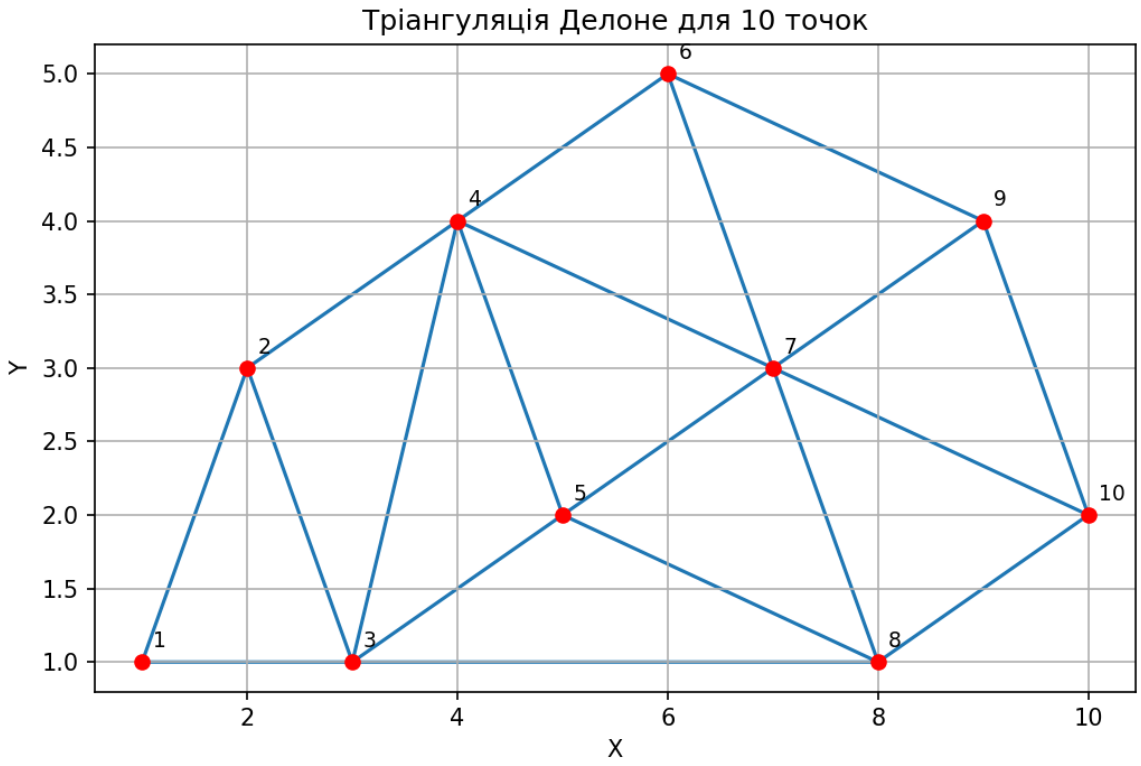
Висновки.
Успішне виконання даного завдання дозволить продемонструвати як теоретичне засвоєння концепцій обчислювальної геометрії, так і практичні навички їх використання для вирішення реальних проблем за допомогою сучасних комп'ютерних інструментів.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial import Delaunay

# Координати 10 заданих точок
points = np.array([
    [1, 1], [2, 3], [3, 1], [4, 4], [5, 2],
    [6, 5], [7, 3], [8, 1], [9, 4], [10, 2]
])

# Побудова тріангуляції Делоне
tri = Delaunay(points)

# Візуалізація
plt.figure(figsize=(8, 5))
plt.triplot(points[:, 0], points[:, 1], tri.simplices)
plt.plot(points[:, 0], points[:, 1], 'ro')  # точки
for i, (x, y) in enumerate(points):
    plt.text(x + 0.1, y + 0.1, str(i + 1), fontsize=9)
plt.title("Тріангуляція Делоне для 10 точок")
plt.xlabel("X")
plt.ylabel("Y")
plt.grid(True)
plt.show()
```



ЗАДАЧА 2.
Інкрементний алгоритм тріангуляції – наочний приклад

Умова: Початковий трикутник складається з точок A(2, 2), B(6, 2) та C(4, 5).
1. Додається точка D(4, 3), яка знаходиться всередині трикутника.
2. Потім додається точка E(1, 4), що розташована поза межами трикутника (зліва).
3. Далі додається точка F(7, 4), також поза межами (праворуч).
Принцип роботи інкрементного алгоритму.Спершу створюється початкова тріангуляція у вигляді базового трикутника. Додавання нових точок виконується за такими правилами, якщо точка потрапляє всередину існуючого трикутника, його площа розділяється на три менші трикутники. Якщо точка лежить поза поточною сіткою, визначаються ребра існуючої сітки, що утворюють контур для нової точки, після чого додаються нові трикутники.

Кроки виконання:

Крок : Початкова тріангуляція складається з одного великого трикутника A–B–C.

Крок 1: додаємо D(4, 3) Точка D знаходиться всередині базового трикутника. Сітка змінюється: початковий трикутник розбивається на три нових: A–B–D B–C–D C–A–D

Крок 2: додаємо E(1, 4) Точка E розташована поза поточною сіткою зліва. Її з'єднують із двома точками базового трикутника (A і C), які знаходяться в зоні прямої видимості. Створюється новий трикутник: A–C–E

Крок 3: додаємо F(7, 4) Аналогічно до попереднього кроку, точка F лежить поза поточною сіткою, але праворуч. Її з'єднують із двома точками базового трикутника (C і B), що утворюють видимі зв'язки. Формується новий трикутник: B–C–F

Результат: після виконаних дій на основі початкової тріангуляції формується нова сітка із зазначених трикутників.

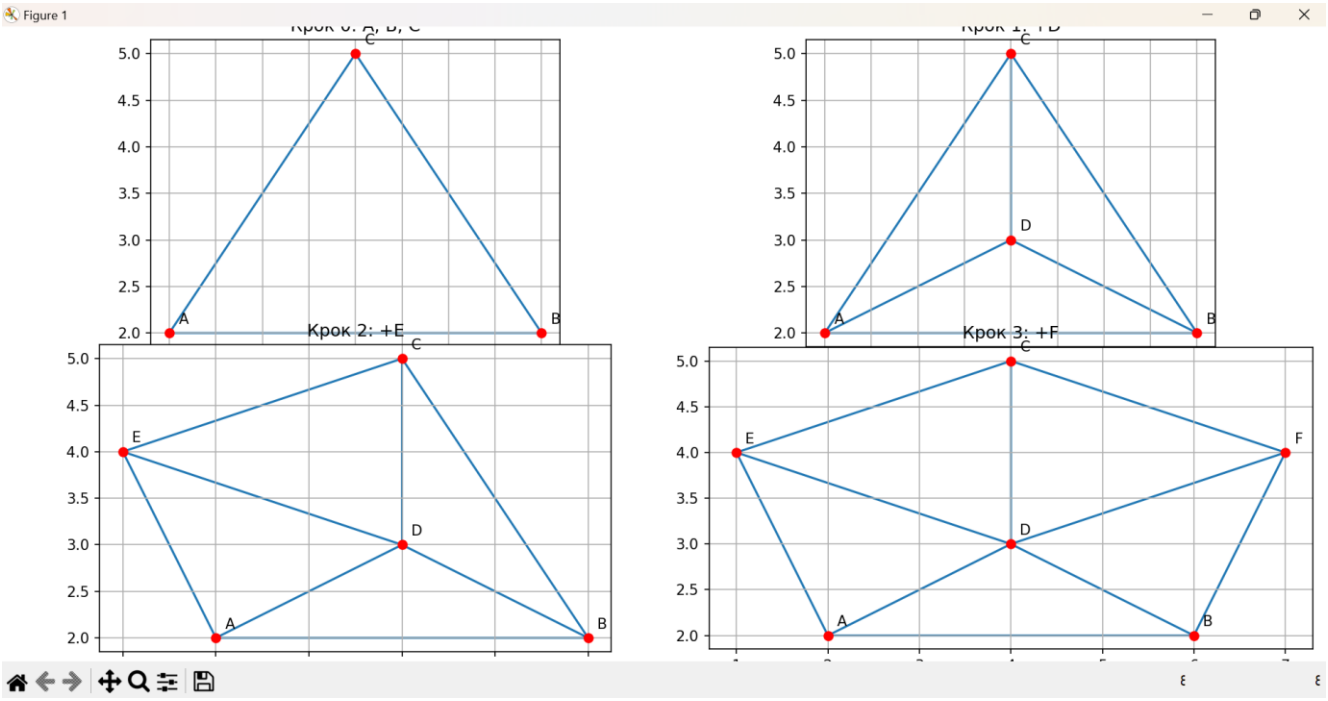
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.spatial import Delaunay

def plot_sub(ax, points, title):
    tri = Delaunay(points)
    ax.triplot(points[:, 0], points[:, 1], tri.simplices)
    ax.plot(points[:, 0], points[:, 1], 'ro')
    for i, (x, y) in enumerate(points):
        ax.text(x + 0.1, y + 0.1, f'{chr(65 + i)}', fontsize=10)
    ax.set_title(title)
    ax.set_aspect('equal')
    ax.grid(True)

# Кроки
points_0 = np.array([[2, 2], [6, 2], [4, 5]])                    # A B C
points_1 = np.vstack([points_0, [4, 3]])                         # +D
points_2 = np.vstack([points_1, [1, 4]])                         # +E
points_3 = np.vstack([points_2, [7, 4]])                         # +F

# Малюємо 4 підграфіки
fig, axs = plt.subplots(2, 2, figsize=(12, 10))
plot_sub(axs[0, 0], points_0, "Крок 0: A, B, C")
plot_sub(axs[0, 1], points_1, "Крок 1: +D")
plot_sub(axs[1, 0], points_2, "Крок 2: +E")
plot_sub(axs[1, 1], points_3, "Крок 3: +F")

plt.tight_layout()
plt.show()
```



Висновки : Я застосовую інкрементний метод побудови, при якому сітка формується поступово. З кожним додаванням нової точки алгоритм модифікує лише ту ділянку тріангуляції, яка безпосередньо зазнає змін через появу цієї точки. Такий підхід дозволяє уникнути повного перебудування всієї структури, що особливо зручно під час роботи з динамічними даних.

ЗАДАЧА 3.
Аргументація вибору алгоритму тріангуляції

Умова:
Задача А) Створення цифрової моделі рельєфу на основі GPS-даних.

Формування цифрової моделі рельєфу на основі GPS-даних Оптимальний вибір тріангуляція Делоне.

Обґрунтування:
GPS-позиції є набором випадкових точок, розкиданих на площині або у 3D-просторі, часто нерівномірно. У таких випадках тріангуляція Делоне забезпечує формування якісної сітки, уникаючи вузьких кутів, що суттєво підвищує точність при моделюванні рельєфу. Вона ідеально взаємодіє з діаграмами Вороного, які широко використовуються в геоінформаційних системах (ГІС). Завдяки цьому можна побудувати стабільну модель поверхні навіть із невпорядкованих вхідних даних.

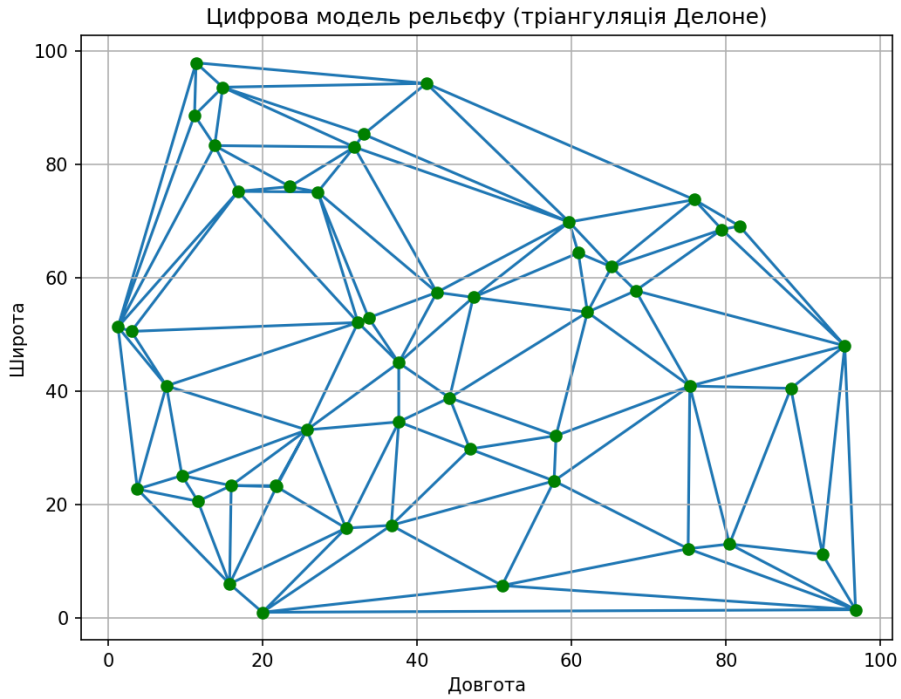
Крім того, алгоритми Делоне демонструють хорошу стійкість до похибок при роботі з реальними GPS-координатами, що робить їх особливо актуальними у геодезії та картографії.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial import Delaunay

# Симуляція GPS-даних: 50 випадкових точок на площині
gps_points = np.random.rand(50, 2) * 100  # маштабування до 100x100

# Тріангуляція Делоне
tri = Delaunay(gps_points)

# Візуалізація
plt.figure(figsize=(8, 6))
plt.triplot(gps_points[:, 0], gps_points[:, 1], tri.simplices)
plt.plot(gps_points[:, 0], gps_points[:, 1], 'o', color='green')
plt.title("Цифрова модель рельєфу (тріангуляція Делоне)")
plt.xlabel("Довгота")
plt.ylabel("Широта")
plt.grid(True)
plt.show()
```



Висновок:
Для побудови цифрової моделі рельєфу доцільно використовувати тріангуляцію Делоне як оптимальний метод через високу стабільність та якість отриманої сітки.

ЗАДАЧА 4.

Оптимізація тріангуляції для множини з 10 000 точок.

Умова:

Реалізувати тріангуляцію для 10 000 точок таким чином, щоб забезпечити ефективність обчислень без затримок та зависань. Поставлено питання щодо можливих підходів, інструментів і методів, які необхідно застосувати для успішної реалізації цього завдання

.

Розгорнута відповідь:

1. Вибір ефективного алгоритму.

Для обробки великої множини точок одним із найоптимальніших рішень є використання тріангуляції Делоне. Ця техніка забезпечує побудову доволі стабільної структури з обчислювальною складністю O(n log n), що підходить для роботи із обсягами даних такого рівня. Бібліотека `scipy.spatial.Delaunay` здійснює реалізацію тріангуляції Делоне за допомогою алгоритму Qhull. Цей алгоритм має високу швидкодію і може ефективно працювати з множинами до 50 000 точок. Однак для додаткової оптимізації продуктивності слід вдатися до кількох удосконалень у підготовці середовища та виборі супутніх підходів.

2. Застосування високопродуктивних бібліотек.

Для досягнення максимальної ефективності рекомендовано використовувати такі бібліотеки: `scipy.spatial.Delaunay`. Даний інструмент забезпечує зручну і стабільну реалізацію тріангуляції Делоне завдяки використанню алгоритму Qhull, написаного мовою C. Він не потребує значної конфігурації, що робить його ідеальним для завдань середнього масштабу. Однак при роботі із великими об'ємами даних його продуктивність може бути недостатньою. - `pygalmesh` чи `meshpy`. Обидві бібліотеки базуються на високопродуктивній. Вони забезпечують підвищену масштабованість та вищу точність розрахунків, що особливо корисно для задач із великими обсягами даних.

Для встановлення бібліотеки Pygalmesh можна скористатися командою `pip install pygalmesh`. Зазначимо, що це рішення потребує попередньо налаштованого середовища (Linux або Windows WSL).

3. Додаткові рекомендації з оптимізації

Обхід відображення у реальному часі. Для задач з великою кількістю точок візуалізація може викликати суттєві уповільнення. Якщо візуалізація не є критичною вимогою, краще виконувати її окремо після завершення основних обчислень. Використання бібліотек прискорення обчислень, інтеграція зі спеціалізованими інструментами типу Numba дозволяє значно прискорити роботу алгоритмів.

Паралельна обробкаРозподіл задач між кількома процесорами за допомогою модуля `multiprocessing` може значно покращити продуктивність при роботі із великими даними. Розподіл простору на сектори.
Для ще більшої масштабованості можна застосувати підхід розбиття множини точок на частини з подальшою незалежною обробкою секторів. Це дозволяє виконувати частину роботи паралельно та уникати надмірного використання ресурсів пам’яті.

Висновок:
Для ефективної реалізації тріангуляції множини з 10 000 точок слід використовувати алгоритми з оптимальною часовою складністю, як-от Qhull, реалізований у бібліотеці `scipy.spatial.Delaunay`.
За необхідності підвищення продуктивності є сенс перейти до більш спеціалізованих рішень на базі CGAL, як-от `pygalmesh` чи `meshpy`.
Для оптимізації застосували реалізацію з бібліотеки SciPy, що базується на C-ядрі Qhull. Це забезпечує обчислення тріангуляції зі складністю O(n log n).
Для роботи з ще більшими наборами даних можна використовувати CGAL або поділ задачі на під області з подальшою паралельною обробкою.
Для оптимізації застосували реалізацію з бібліотеки SciPy, що базується на C-ядрі Qhull. Це забезпечує обчислення тріангуляції зі складністю O(n log n).
Для роботи з ще більшими наборами даних можна використовувати CGAL або поділ задачі на під області з подальшою паралельною обробкою.

```
import numpy as np
from scipy.spatial import Delaunay
import matplotlib.pyplot as plt
import time

points = np.random.rand(10000, 2)

start = time.time()
tri = Delaunay(points)
end = time.time()

print(f"Тріангуляція виконана за {end - start:.2f} сек")

# Збереження без показу
plt.figure(figsize=(10, 10))
plt.triplot(points[:, 0], points[:, 1], tri.simplices, linewidth=0.3)
plt.axis('off')
plt.savefig("triangulation_10000.png", dpi=300)
```

```
Тріангуляція виконана за 0.04 сек
```

ВИСНОВКИ

У ході виконання курсової роботи було досліджено теоретичні основи тріангуляції множин точок і полігонів, здійснено аналіз найпоширеніших алгоритмів та реалізовано їх практичне застосування на мові програмування Python. Розглянуто тріангуляцію Делоне, інкрементний алгоритм і метод динамічного програмування, кожен із яких має свої переваги залежно від специфіки розв'язуваних задач. Тріангуляція Делоне проявила себе як найбільш оптимальний варіант за якістю та швидкодією при роботі з множинами точок. Інкрементна тріангуляція показала високу гнучкість у сценаріях із динамічними об’єктами, тоді як метод динамічного програмування виявився ефективним для побудови оптимальних розбиттів опуклих багатокутників.

Практична частина роботи успішно продемонструвала можливості побудови тріангуляції для великих наборів даних, що охоплюють до 10 000 точок, без суттєвих втрат у швидкодії. Це підтвердило ефективність застосовуваних алгоритмів та їх доцільність для опрацювання геометричних даних у таких галузях, як комп’ютерна графіка, геоінформаційні системи чи 3D-моделювання.

Одержані результати можуть слугувати базою для подальших досліджень, наприклад, в напрямках оптимізації тріангуляції для просторів в 3D, створення адаптивних сіток або використання в ігровій індустрії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дейлоне Б. Н. Геометрия числа и триангуляции. — М.: Наука, 1992.
2. Preparata F. P., Shamos M. I. Computational Geometry: An Introduction. — Springer-Verlag, 1985.
3. Berg M. de, Cheong O., van Kreveld M., Overmars M. Computational Geometry: Algorithms and Applications. — Springer, 3rd ed., 2008.
4. Qhull — Official Site: https://qhull.org
5. Scipy.spatial.Delaunay documentation — https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.Delaunay.html
6. Matplotlib — Python Plotting Library: https://matplotlib.org
7. CGAL — Computational Geometry Algorithms Library: https://www.cgal.org
8. NumPy Documentation — https://numpy.org/doc/