Vol. 6, No. 1, 2024

Олександр Манюк[1], Ярослав Соколовський[2]

[1] Кафедра систем автоматизованого проектування, Національний університет "Львівська політехніка", вул. С. Бандери, Львів, Україна, E-mail:oleksandr.m.maniuk@lpnu.ua, ORCID 0009-0007-5673-1522
[2] Кафедра систем автоматизованого проектування, Національний університет "Львівська політехніка", вул. С. Бандери, Львів, Україна, E-mail: yaroslav.i.sokolovskyi@lpnu.ua, ORCID 0000-0003-4866-2575

ПОБУДОВА АЛГОРИТМУ ТА СТРУКТУРИ ДАНИХ ДЛЯ ПРИШВИДШЕНОГО ПОШУКУ ЕЛЕМЕНТІВ ДИСКРЕТИЗАЦІЇ 2D-ОБЛАСТЕЙ

Отримано: березень 04, 2024 / Переглянуто: березень 20, 2024 / Прийнято: квітень 01, 2024

https://doi.org/

Анотація. Дана робота присвячена покращенню швидкодії обробки пошукових запитів до планарних дискретних моделей, що застосовуються в інженерному програмному забезпеченні. Розроблено структуру даних для пришвидшення пошуку елементів дискретизації на основі ієрархічної триангуляційної сітки. Розроблена індексаційна структура будується висхідним ітеративним алгоритмом, який конструює кожен новий рівень ієрархії на основі попереднього або індексованої триангуляції шляхом його спрощення, що забезпечує наскрізне збереження морфології триангуляційної сітки у всіх рівнях ієрархічної індексаційної структури. Розроблений алгоритм побудови забезпечує наявність деревоподібних зв'язків між рівнями ієрархічної триангуляційної сітки, що дозволяє низхідну навігацію між геометрично близькими трикутниками. Пришвидшення пошуку досягається завдяки виконанню направленого пошуку в верхньому рівні індексаційної структури та подальшої навігації між рівнями з використанням низхідних зв'язків, доки не буде знайдено трикутник індексованої дискретизації. Здійснена програмна реалізація засобами C++17, візуалізація триангуляційних сіток та ізоліній здійснена засобами бібліотеки ObjectARX. На основі програмної реалізації створено виконавчу бібліотеку.

Ключові слова: дискретизація 2D-області**,** пришвидшений пошук, дискретна модель, нерегулярна триангуляційна сітка, UML-діаграми, бібліотека Object ARX

Вступ та постановка задачі

Актуальність.дослідження. Проблеми пошуку елементів дискретизації просторових даних активно опрацьовуються науковцями вже понад половину століття та, попри розробку значної кількості можливих рішень, не втрачає актуальності до тепер. Будь-який з напрямків інженерії працює з просторовими об'єктами, що, залежно від спрямування, відрізняються масштабом та розмірами: від нанометрів в мікроелектроніці до десятків або й сотень кілометрів в галузі будівництва. Просторові дані використовуються як для опису проектованих об'єктів так і середовища, в якому цей об'єкт існуватиме, якщо це матиме вплив на нього. Складно уявити процес проектування житлового комплексу без прив'язки до рельєфу та місцевості, а мостового переходу без врахування гідрологічних та геологічних умов. Описана проблематика лежить на стику систем автоматизованого проектування (САПР) та геоінформаційними систем (ГІС), що обумовлює потребу в інтеграції, в першу чергу, в контексті механізмів опису та опрацювання поверхонь. В обох галузях існує консенсус, який полягає в широкому застосуванні дискретних чи дискретизованих моделей поверхонь попри те, що використання кривих (криві Без'є, B-сплайни (B-Splines), неоднорідні раціональні B-сплайни (Non-UniformRational B-Splines), тощо) та поверхонь вищих порядків активніше використовується в задачах генерації поверхонь, а не опису існуючих.

Аналіз результатів.досліджень. Застосування дискретних моделей вважається природним та раціональним підходом до опису поверхонь, оскільки в його основі лежать саме точкові (дискретні) виміри, виконані вручну або автоматизовано; також він дозволяє дотриматись оптимального балансу між розміром даних та точністю відтворення поверхні [7]. Найпопулярнішими серед способів дискретного представлення поверхонь є розміщення точок (PointClouds), регулярні (Mashes) та нерегулярні сітки (TIN – Triangulatedirregularnetwork); кожен з цих способів, базується на різних елементах дискретизації (точки, грані, трикутники), а також дозволяє трансляцію моделі між різними представленнями. Розміщення точок – відносно новий спосіб який набуває популярності з поширенням технологій лазерного сканування LIDaR (Light Identification, Detectionand Ranging), яка дозволила значно здешевити вартість вимірювання: проблемою методу є надмірність даних, що породжує потребу в їх фільтрації. Регулярні сітки – зручний спосіб опису поверхні, проте, на відміну від нерегулярних, не має змоги адаптуватись до флуктуацій. Саме нерегулярні тріангуляційні сітки, є предметом розгляду даної роботи.

Задача пошуку елементів дискретизації TIN полягає в локалізації трикутника, який містить задану точку. Найпростіший спосіб пошуку трикутника – пошук перебором. Він полягає в послідовній оцінці положення точки відносно кожного трикутника, доки не буде знайдено той, що містить точку; такий підхід має лінійну алгоритмічна складність O(n), що на практиці означає значне зниження продуктивності при збільшенні розміру задачі. Альтернативний спосіб – пошук по напрямку, який полягає в використанні зв'язків між елементами дискретизації, якщо такі наявні, для поступового скорочення відстані між центром деякого поточного трикутником та шуканою точкою, завдяки пошуку нового трикутника серед тих, що мають спільне ребро або вершину з поточним і покращують результат; застосування значно скорочує кількістьопрацьованих трикутників;застосування алгоритму доцільне лише для невеликих дискретизацій.

Покращення швидкодії екстенсивним способом полягає в т.з. індексуванні пошуку, суть якого, згідно з Чжаном (Zhang, X) [1], полягає в використанні спеціальної структури даних, яка дозволяє ефективніший доступ до індексованих елементів. В широкому розумінні, індексаційні структури відображає окремі аспекти об'єктів, але не містить їх точного відображення, та, в загальному випадку, він покращує пошук завдяки зберіганню референтних значень в лінійній послідовності, що не є оптимальним для просторових даних. Згідно з Ріго (Rigaux) [6], ця проблема обумовила розвиток двох гілок структур для індексації просторових даних: керовані простором (space-driven) та керовані даними (data-driven), різниця між якими полягає як різному підході до структур даних та способів доступу; керовані даними базуються на використанні ідеї просторових зв'язків (Spatial Containments), замість дискретних комірок, як в керованих простором. До керованих простором відносять регулярні сіткові структури (Fixedgridindex), дерева квадрантів (Quadtree), k-вимірні дерева (KD-tree) та, так зване,геокодування (Geohash). Керовані даними структури це R-дерево (R-tree) та його модифікації. Перевагою просторово орієнтованих структур є можливість попередньої підготовки структури, натомість, перевагою структур орієнтованих на дані є їх ефективність ташвидке виконання пошуку.

Дюоном (Dutton) та ін. [5] було запропоновано модифікацію моделі дерева квадрантів з ієрархічною структурою; Кунстом (Kunszt) та ін. [9] було запропоновано модифікацію моделі дерева квадрантів з ієрархічною структурою, що була вперше застосована як спосіб каталогізації БД астрономічних об'єктів. Метод отримав назву ієрархічної тріангуляційної сітки – HTM (Hierarchical Triangular Mesh), оскільки, його суть полягає в побудові тріангуляційної сітки, в якій кожен трикутник можна розділити на менші і продовжувати цей процес, доки не буде досягнуто необхідної роздільної здатності тріангуляції; пізніше ідеї було розвиненоКунстом (Kunszt) та ін. [10], Салаєм (Szalay) та ін. [11]. Горскі (Gorski) та ін. [12] було запропоновано модель дискретизації HEAL Pix (Hierarchical Equal Areaiso Latitude Pixelisation), яку також було розроблено для обробки та впорядкування великих об'ємів даних мікрохвильових астрономічних експериментів. Суть методу полягає в розділення поверхні сфери з допомогою ромбовидних пікселів, однакової площі, що підтримує побудову ієрархії через розділення пікселів на менші; описані також розбиття на

пікселі інших форм [13]. Обидва методи були порівняні Омуланом (O'Mullane) та ін. [13]; було здійснено спробу комбінованого використання моделей, та визнано, що обидва підходи надто різні для цього. Також автори визнають перевагу HEALPix в збалансованості структури для пошуку, що гарантує кращу швидкодію в найгіршому випадку, завдяки використанню більш однорідного поділу просторо; натомість, можливість локальної деталізації дозволяє HTM-структурі показувати кращі результати з нерівномірно розподіленими даними. Пізніші роботи описують застосування структур даних HTMта HEALPix в сфері геоінформатики та управління даними тривалих наукових проектів; Райлі (Rilee) та ін. [14], [15] наводять застосування HTM-структур для систематизації просторо-часових даних спектрографії проектів MODIS (Moderate Resolution Imaging Spectroradiometer) та GOES(Geostationary Operative Environmental Satellite), та загалом, т.з. Big Earth Data. Есен (Esen) та ін. [16] описують застосування HEALPix-структури для аналізу картографічних проекцій, та доводять наявність спотворень площі при застосуванні методу.

Постановка задачі. Згадані дослідження об'єднує застосування ієрархічних індексаційних структур для глобальних задач; індексовані ними дані стосуються земного геоїда або ж небесної сфери, що обумовлює фіксовану кількість елементів у верхніх рівнях індексаційної структури, та, відповідно, низхідну побудову. Наша робота сконцентрована на побудові індексаційної структури для дискретизації, обмеженою визначеною областю та заданою в прямокутних координатах, та передбачає висхідний напрям побудови. Оскільки, індексована дискретизація є нерегулярною триангуляційною сіткою, доцільним є застосування HTM-подібної структури.

Об'єктом дослідження є процес пошуку елементів дискретизації 2D-областей.

Предметом дослідження є алгоритми та структури даних для пришвидшення пошуку елементів дискретизації 2D-областей на основі ієрархічно триангуляційної сітки.

Метою дослідження є розробка індексаційної структури даних та алгоритму її опрацювання задля пришвидшення просторового пошуку елементів дискретизації.

Для досягнення поставленої мети визначено наступні основні завдання дослідження:

- Розробка тестової структури даних дискретної моделі, до якої необхідно застосувати індексацію;
- Розробка загальних механізмів функціонування тестової моделі, включно з простими механізмами пошуку її елементів;
- Аналіз переваг та недоліків відомих індексаційних структур та синтез власної структури даних для індексації, механізмів її побудови та функціонування на основі ієрархічної триангуляційної сітки.

Наукова новизна. Розроблено індексаційну структуру даних на основі ієрархічної триангуляційної сітки (HTM), кожен з рівнів якоїє нерегулярною триангуляційною сіткою, морфологія якого наближена до попереднього рівня.Розроблена структура застосована до задачі обробки пошукових запитів до дискретної моделі.Особливістю структури є використання в ній елементів, подібних до індексованого об'єкта (TIN), що забезпечує можливість використання загальних механізмів пошуку (пошук в напрямку), а також, можливість навігації в деревовидній структурі зв'язків між трикутниками індексних та індексованих дискретизацій.

Практична значимість. Реалізовано тестову модель дискретизації 2D-області, що базується на нерегулярній триангуляційній сітці на основі критерію Делоне. Також реалізовано індексаційну структуру для дискретизації. На основі реалізації сформовано файл бібліотеки dll (DynamicLinkLibrary), що забезпечить можливість повторного використання в подальших дослідженнях, створення надбудов та використання в сторонньому програмному забезпеченні.

Виклад основного матеріалу

Для тестової моделі дискретизації розроблено нерегулярну триангуляційну мережу TIN (Triangulated irregular network), яка на практиці широко застосовується в інженерному програмному забезпеченні в якості ядра інтерполяційних моделей поверхонь. На даний момент описана значна кількість підходів до внутрішньої організації неоднорідних триангуляційних мереж а також

модифікацій структур даних та алгоритмів, що забезпечують їх функціонування [8]. Було прийнято рішення реалізувати тестову модель на основі найбільш дослідженої та описаної моделі – тріангуляції Делоне [9,10].

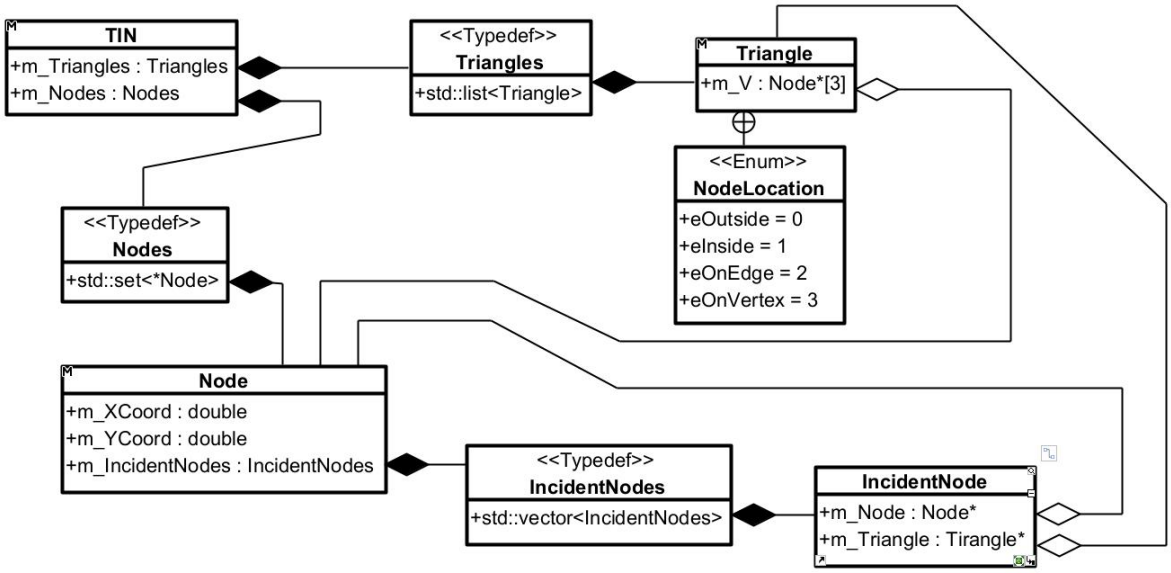


Рис. 1.UML-діаграма класів реалізованої дискретної моделі з відображеними атрибутами сутностей

Серед усіх популярних структур тріангуляції Делоне обрано структуру «Вузли та трикутники», оскільки вона найбільш очевидно ілюструє сутність дискретизації поверхні та дозволяє прямий доступ до трикутників, які і є цільовим елементом дискретизації. Розроблена структура нерегулярної сітки містить контейнери з трикутниками та вузлами (рис. 1.). Вона подана у вигляді UML-діаграми класів з відображеними атрибутами сутностей .В якості контейнера для трикутників обрано список, оскільки він забезпечує відсутність втручання в пам'ять, виділену під збережені об'єкти та їх копіювання при модифікації контейнера, а також зберігає актуальність ітераторів на елементи [10]. Для вузлів обрано множину з тих же причин, що і список для трикутників, але важливою особливістю множини є можливість підтримки впорядкованості контейнера та умовно легкого пошуку елементів [11].

```
class TIN {
public:
        // Метод виконує направлений пошук трикутника по точці, починаючи з заданого, або
  останнього в контейнері
        boolFind(const Point2d&pnt, constTriangle*&pFindedTriangle, constTriangle*pHintTriangle
  = nullptr);
private:
        Nodes  m_Nodes                  ; // Контейнер вузлів
        Trianglesm_Triangles    ; // Контейнер трикутників
};
```

Структура трикутника володіє масивом з трьох вказівників на вузли, які в свою чергу визначають геометричні параметри трикутника. Також присутні методи для швидкого доступу до вершини по індексу, перевірки належності вузла до трикутника та локалізації довільної точки.

```
classTriangle{
public:
        Triangle(Node* pV1, Node* pV2, Node* pV3);
        Node* operator[](intIndex) const;                    // доступ до вузла по індексу
```

```
        boolHasNode(constNode* pNode) const;          // перевірити наявність вузла
        enumNodeLocation{     // випадки розміщення довільної точки щодо трикутника
                eOutside        // ззовні
                , eInside                // всередині
                , eOnEdge       // На ребрі
                , eOnVertex     // У вершині};
        NodeLocationGetPointLocation(double x, double y); // перевірка розміщення
        boolContain(double X, double Y) const;  перевірка входження//
private:
        Node* m_V[3];
}
typedef std::list<Triangle>Triangles;
```

Опорним елементом триангуляційної сітки є вузол – точка на плані або в просторі на яку спираються трикутники. Структура для опису вузла містить двійку раціональних чисел, які відповідають координатам точки на площині. Параметра для опису висотної відмітки вузла не додано через надлишковість, оскільки предметом інтересу дослідження є пошук на площині.

```
classNode{
public:
        Node(double dX, double dY);
        IncidentNode* GetEdgeTo(const Node* pOther);
private:
        doublem_XCoord;
        doublem_YCoord;
        std::vector<IncidentNode>m_IncidentNodes;
};
typedef std::set<Node*> Nodes;
```

Структура «Вузли та трикутники» забезпечує швидкий доступ від трикутника до вузла але ускладнює зворотній доступ, оскільки в найгіршому випадку доведеться перевірити всі трикутники. Для покращення зв'язності моделі базова структура розширена додатковим контейнером зв'язків вузла з суміжними, який перебуває в структурі вузла. Кожен такий зв'язок складається з вказівника на вузол, зв'язок з яким описується та ітератор на суміжного трикутника.

```
classIncidentNode {
public:
        IncidentNode(Node* pNode, Triangle*pTriangle = nullptr);
        bool HasTriangle()const;
        const Triangle* GetTriangle()const;
private:
        Triangle*       m_Triangle      ; // Трикутник праворуч
        Node*   m_Node                  ; // Вузол, до якого прямує ребро
}
typedef std::vector<IncidentNode>IncidentNodes;
```

Фактично, структура суміжного вузла описує направлене ребро, яке володіє даними лише про пункт призначення та трикутник, що знаходиться праворуч від нього.

Побудова неоднорідної сітки відбувається типовим ітеративним алгоритмом. Перший етап включає наповнення контейнера вузлів без встановлення зв'язків між ними та формування топології мережі (рис. 2.а). Наступним кроком є підготовка так званої супертриангуляції – неоднорідної триангуляційної сітки, яка покриває всі вузли, що мають бути додані, але включає лише крайні (рис. 2. б). Важливою вимогою до такої мережі є випуклість, що означає можливість з'єднати будь яку пару вузлів лінією, яка не покидатиме межі триангуляції. Побудова супер-

триангуляції включає розрахунок випуклої оболонки по множині вузлів методом Ендрю [2] та наповнення триангуляції трикутниками (рис. 2. в) по отриманому полігональному контуру так званим методом відтинання вуха (Ear Clipping) [3].

Операція додавання трикутника також ініціалізує у вузлах контейнери зв'язків з суміжними вузлами. Кожна структура зв'язку ініціалізується вказівником на вузол кінця ребра та вказівником на трикутник, який утворений даним ребром. Наступний етап передбачає обхід всіх ще не застосованих в триангуляції вузлів з пошуком трикутника, в який потрапляє цей вузол. Для знайдених трикутників виконується оцінка положення вузла, та його розбиття на два (вузол потрапив на ребро) або три (вузол потрапив в середину) менших трикутники.

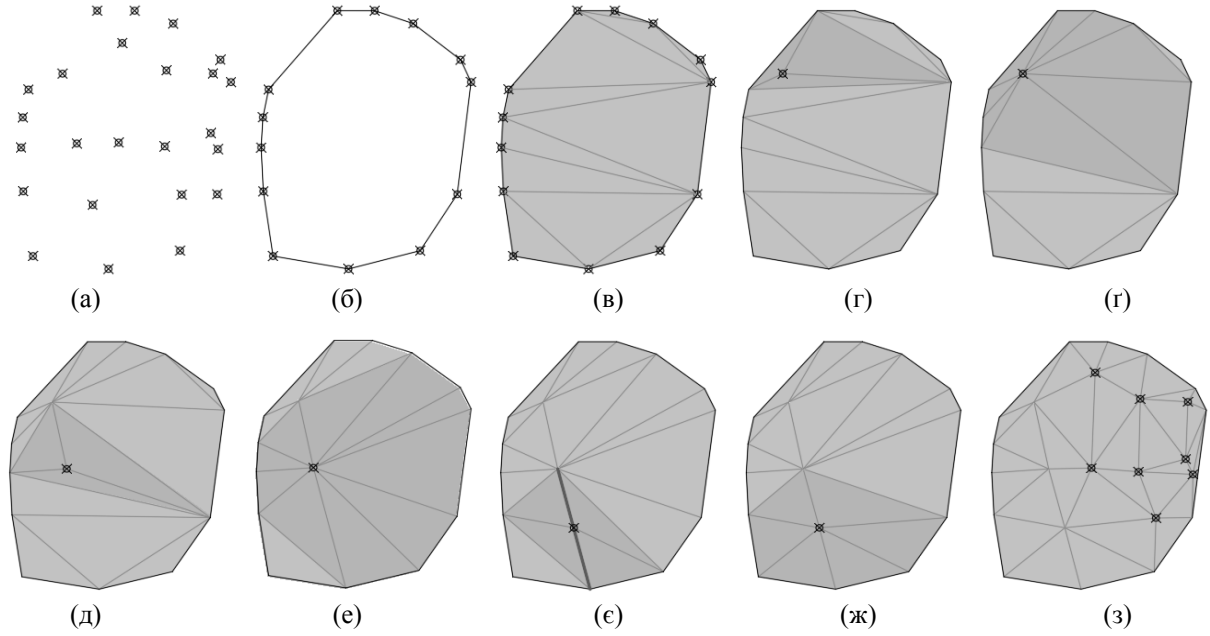


Рис. 2. Приклад виконання побудови: Множина вхідних точок(а), результат побудови випуклої оболонки методом Ендрю(б), наповнення утвореного полігона трикутниками(в) та поступове додавання в триангуляцію решти точок (г-з), з вставкою вузлів в середину трикутника (г, д) та вставкою на ребро(є). Також проілюстровані зміни топології (змінені області затінені)в результаті перевірки умови Делоне після додавання нових вузлів (г – ґ, д – е, є – ж)

Відносно знайденого трикутника необхідно оцінити положення вузла. При потраплянні вузла всередину виконується розбиття трикутника на три нових (Рис. 2.г,д), при чому, за для оптимізації алгоритму трикутник не видаляється, але відбувається його переналаштування на іншу трійку вузлів та ребер, з додаванням двох нових трикутників в контейнер, та перевірка виконання умови Делоне (Рис. 2.ґ, е) згідно [4], для всіх трьох трикутників, за потреби, їх ребра перебудовуються. При потраплянні на ребро трикутника (Рис.2.є), воно буде розбите на два, відповідно, аналогічне розбиття також виконується із суміжним ребру трикутником.

Оцінка розміщення точки в трикутнику виконується шляхом послідовної перевірки гіпотез про розміщення точки на одному з вузлів трикутника, на одному з ребер трикутника та всередині трикутника. Якщо жодна з трьох гіпотез не підтвердилась, вважається, що шукана точка знаходиться ззовні. Оскільки для представлення точок використано пару чисел з плаваючою точкою, порівняння значень виконано з точністю до 1e-9. Оцінка положення точки відносно прямої реалізовано з використанням векторного добутку між напрямком ребра та напрямком з точки початку ребра на шукану точку.

Організація пошуку в TIN

Задля забезпечення базового функціоналу пошуку в неоднорідній триангуляційній сітці було реалізовано алгоритм пошуку в напрямку. Як вже було сказано раніше, обрані структури даних забезпечують сильну зв'язність елементів дискретизації між собою, що, в свою чергу, дозволяє почати виконання алгоритму з довільного елемента дискретизації та послідовно скорочувати відстань до шуканої точки, доки поточний трикутник не міститиме шукану точку. Фактично, описаний алгоритм передбачає своєрідне «крокування» між елементами дискретизації, а скорочення відстані від визначеної точки довільного елементу дискретизації до шуканої точки буде ознакою покращення стану пошуку.

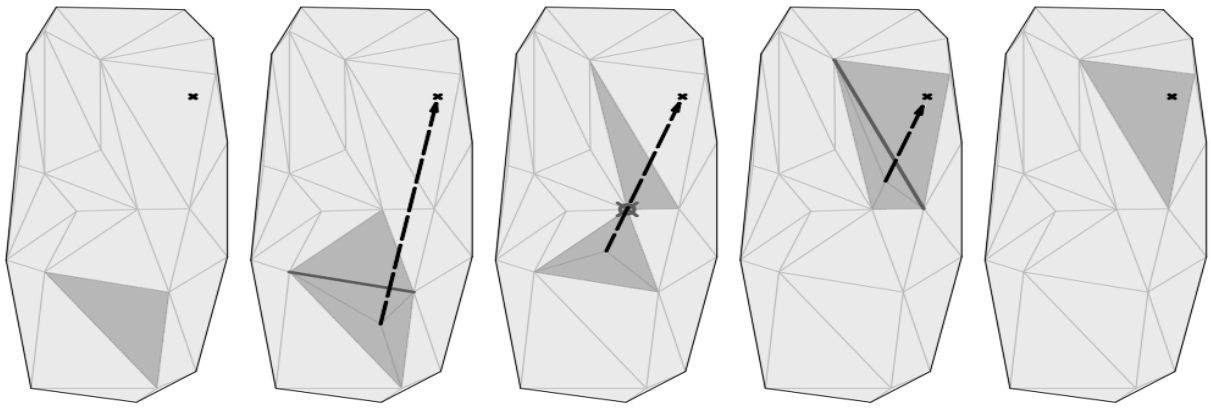


Рис. 3. Виконання направленого пошукув нерегулярній триангуляційній сітці, починаючи від довільного трикутника (а). Під час виконання застосовано перехід до суміжного трикутника через ребро (б), пошук сектору для продовження пошуку після потрапляння у вузол(в), повторний перехід через наступне ребро (г) та успішне завершення пошуку в знайденому трикутнику (ґ)

Обраний алгоритм передбачає побудову відрізка, що сполучає центр мас початкового трикутника та шукану точку. Після знаходження першого пересічення відрізка з ребром або вершиною суміжного трикутника, один с кінців відрізка підміняється цією точкою для мінімізації накопичувального ефекту обчислювальних похибок. В залежності від розміщення точки пересічення відбувається перехід через ребро або через вершину.

При переході через ребро наступним трикутником обирається суміжний(рис. 3.б, г). Випадок потрапляння у вершину передбачає вибір сектора, утвореного суміжними ребрами, що містить шукану точку, задля продовження пошуку в його напрямку. Реалізація передбачає перегляд всіх пар суміжних ребер, між якими існують трикутники та оцінку розміщення шуканої точки відносно цих ребер; якщо така оцінка співпадає, шукана точка не потрапляє в сектор. В іншому випадку, трикутник потрібно обрати в якості наступного для продовження пошуку(рис. 3. в).

```
SearchActionSearchFromTringle() {
        // … //
        Line<Node>testLine(m_LastIntersection, m_testPnt);
        Point2d intersectPnt;
        Line<Node>edge(m_LastTriangle [0], m_LastTriangle [1]);
        if (!testLine.IsIntersect(edge, &intersectPnt)
          &&!testLine.IsIntersect(edge = Line<Node>( m_LastTriangle [1], m_LastTriangle [2]),
&intersectPnt)
          && !testLine.IsIntersect(edge = Line<Node>( m_LastTriangle [2], m_LastTriangle [0],
&intersectPnt))
                  returneStop;                    // Відсутність пересічень з ребрами свідчить про
        помилку
```

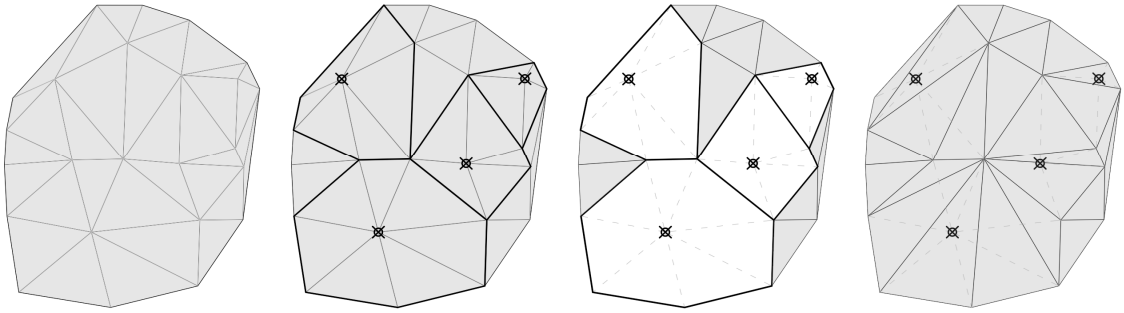
```
        m_LastIntersection = intersectPnt;                  // фіксація останньої точки пересічення
        if (edge.StartNode() == intersectPnt) {             // Перевірка потрапляння у вузол
                m_pLastNode = edge.StartNode();
                returneSearchFromNode;                      // потрапляння в початкову вершину ребра
        } elseif (edge.EndNode() == intersectPnt){
                m_pLastNode = edge.EndNode();
                returneSearchFromNode;                      //потрапляння в кінцеву вершину ребра
        }
        // … //
        constIncidentNode&backEdge = edge.EndNode()->GetEdgeTo(edge.StartNode());
        m_LastTriangle = backEdge.GetTriangle();
        returneSearchFromTriangle;                          // В іншому випадку перехід через суміжне ребро
}
SearchActionSearchFromNode() {
        // … //
        foreach(constIncidentNode&edgeinm_pLastNode->G

        etIncidentNodes())
        {
          if (!edge.HasTriangle())
                continue;                   // Перевірити вихід за межі триангуляції
          constTriangle*pTriangleInSector =*edge.GetTriangle();
          constNode* pRemainNode = pTriangleInSector->GetRemainNode(m_pLastNode,
edge.m_pNode);
          if(Order(edge.m_Node, m_pLastNode, m_testPnt)==Order(pRemainNode, m_pLastNode,
m_testPnt))
                continue;                   // Пропустити трикутникякщо шукана точка не потрапляє в його
сектор
          m_LastTriangleIt = incidentTriangleIt;
          returneSearchFromTriangle;
        }
        // … //    }
```

Вище наведено фрагменти реалізації опрацювання випадків виконання пошуку від довільного трикутника та від вузла.

Індексація пошуку. Ідея пришвидшення пошуку, якій присвячена дана робота, полягає в побудові нерегулярної триангуляційної мережі (індексаційної) на основі оригінальної (індексованої), таким чином, щоб нова модель повністю покривала індексовану та містила значно меншу кількість трикутників.Це забезпечить швидше виконання направленого пошуку в ній, а після його завершення забезпечить перехід до трикутника індексованої моделі, що є шуканим або близьким до нього.Перехід від індексаційної триангуляції до індексованої досягається побудовою зв'язків між їх трикутниками. Як було зазначено вище, триангуляції значно відрізняються розміром, тому, в загальному випадку, кожен трикутник індексаційної сітки матиме зв'язок з кількома трикутниками індексованої, що визначає деревовидну топологію цих зв'язків, таким чином, що кожен трикутник індексаційної сітки буде виконувати роль кореня дерева а окремі трикутники індексованої – листків. Фактично, описана структура відповідає множині дерев, або «лісу» в термінах теорії структур даних.

Очевидно, що наявність зв'язку між трикутниками індексованої триангуляціїта оригінальної передбачає їх геометричну близькість, а точніше «перекривання» індексаційним трикутником оригінальних, що і є ключем до скорочення кількості переглянутих трикутників під час пошуку. Відповідно, побудова індексаційної триангуляції повинна виконуватись по скороченій множині вузлів оригінальної, що актуалізує проблему відсіювання непотрібних та ефективної фіксації зв'язків між трикутниками оригінальної та індексаційної триангуляцій.

Вирішення потреби скорочення кількості вузлів полягає в побудові індексаційної триангуляції шляхом спрощення оригінальної, що передбачає ітеративне застосування (рис. 4), що в свою чергу виявило потребу в збереженні результату проміжних ітерацій. Застосування спрощення також дозволить досягнути збереження загальної морфології триангуляційної моделі при створенні індексаційної триангуляції.



(а) (б) (в) (г)

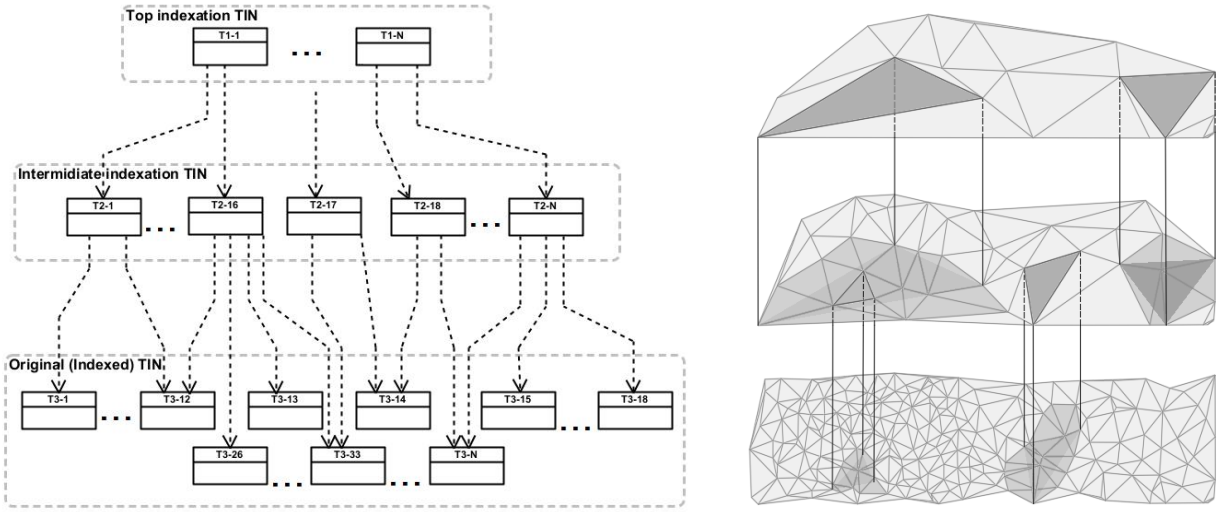
Рис. 4. Виконання однієї ітерації побудови індексаційної триангуляції(г) по оригінальній-індексованій(а), показано вибір вузлів, для виключення(б) та звільнення областей, обмежених суміжними вузлами до обраних(в) з наповненням звільнених областей новими трикутниками (г).

Очевидним вирішенням проблеми збереження зв’язків між рівнями є використання проміжних індексаційних триангуляцій, таким чином, щоб трикутники-корені множини дерев містили зв’язки з трикутниками проміжної індексаційної триангуляції. Останні, в свою чергу, містять зв’язки з трикутниками оригінальної триангуляції, або ж, за потреби, іншої проміжної індексаційної сітки. Таким чином забезпечується можливість розбудови дерев в глибину, відповідно до кількості ітерацій побудови.

Потребу в ефективній фіксації зв’язків між трикутниками різних рівнів дерева вирішено завдяки оцінці перекривання трикутників індексованого рівня індексаційними під час побудови нового рівня. Таким чином, побудова індексаційноїтриангуляції виконується шляхом копіювання індексованої, поступового виключення з неї вузлів та наповнення звільненої області новими трикутниками. Вибір вузла для виключення обмежений загальною вимогою про те, що всі трикутники індексованої повинні бути перекриті індексаційною, відповідно, заборонено виключати вузли, які лежать на границі та можуть вплинути на випуклість триангуляції. Після виключення, звільнена область, обмежена суміжними з виключеним вузлами, наповнюється трикутниками алгоритмом відтинання вуха, що вже був загаданий вище. Для кожного з нових трикутників потрібно провести оцінку перекривання індексованих та зафіксувати факти перекриття в якості зв’язків.

На рис.5(а) наведено схему внутрішніх зв’язків, як вертикальних – в рамках деревовидних зв’язків між рівнями індексаційних нерегулярних сіток, так і горизонтальних в межах внутрішніх зв’язків між дискретними елементами цих сіток. На рис.5(б) наведено приклад зв’язків між трикутниками індексаційної, проміжної індексаційної, та оригінальної триангуляцій; на кожній з триангуляцій показано трикутники індексованої триангуляції (середнє затінення), які перекриваються трикутником індексаційної (сильніше затінення), що, як вже було сказано вище, відповідає наявності зв’язку між трикутниками різних рівнів. Наведений приклад також демонструє зберігання положення окремих вузлів між різними рівнями ієрархічної триангуляційної сітки, що забезпечує зберігання морфології оригінальної триангуляції.

Реалізація індексаційної триангуляції передбачає виділення нових сутностей (рис.6.), які хоч і подібні до застосованих в звичайній реалізації триангуляційної сітки (рис.1.), мають окремі особливості, що забезпечують доступ до оригінальних вузлів та зберігання зв’язків між індексаційними та індексованими трикутниками.



(а) (б)

Рис. 5.Схема внутрішніх зв'язків індексаційної структури (а) та наочне представлення суті окремих зв'язків на прикладі побудованої індексаційної структури (б)

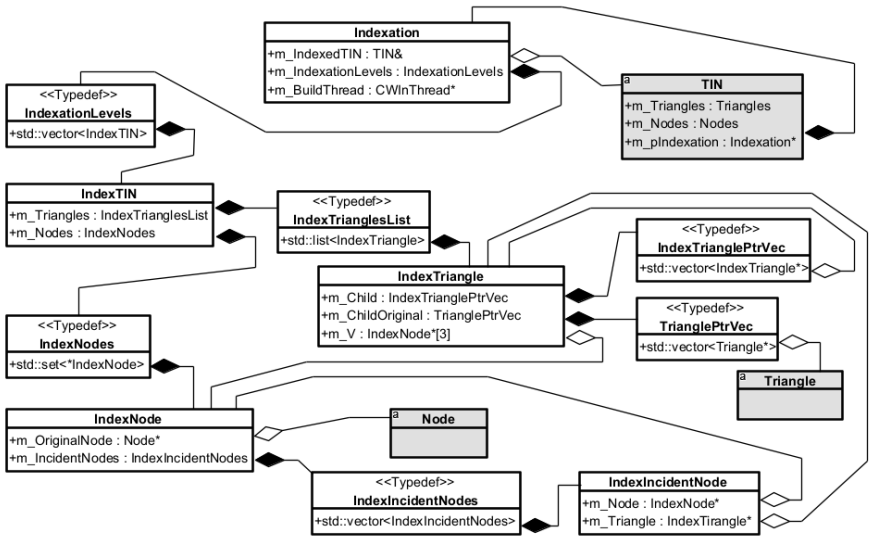


Рис. 6. UML-діаграма класів реалізованої індексаційної структури з відображеними атрибутами сутностей

На рис.6 сірим кольором позначено сутності індексованої дискретної моделі. Індексаційна тріангуляція по структурі ідентична до оригінальної, та складається з контейнерів еквівалентних об'єктів.

```
classIndexTIN {
public:
        // Виконує направлений пошук індексаційноготрикутника по точці, починаючи з
заданого або довільного
        boolFind(const Point2d&pnt, constIndexTriangle*&pFindedTriangle,
  constIndexTriangle*pHintTriangle = nullptr);
        // Перевірити чи побудова завершена
        boolIsBuilded()const;
        // Встановити прапорець завершення побудови
        boolBuildFinished();
private:
        IndexNodes      m_Nodes                 ; // Контейнер вузлів
```

```
        IndexTrianglesList      m_Triangles     ; // Контейнер трикутників
        bool                    m_bIsBuilded    ; // прапорець завершення побудови
};
typedef std::vector<IndexTIN*>IndexationLevelsPtrs;
```

Індексаційний трикутник відрізняється від звичайного тим, що він спирається на власний тип вузлів та містить два контейнери для фіксації зв'язків з індексованими трикутниками. Використовуватись може лише один з контейнерів, в залежності від типу індексованої триангуляції (оригінальної чи проміжної).

```
typedef std::vector<Triangle*>TrianglePtrVec;
classIndexTriangle{
public:
        IndexTriangle(IndexNode* pV1, IndexNode* pV2, IndexNode* pV3);
        // Методи для отримання проіндексованих трикутників (це можуть трикутники з
  індексації або оригінальні)
        IndexTriangle* GetChildByPoint(const Point2d&pnt);
        Triangle* GetChildOriginalByPoint(const Point2d&pnt);
public:
        IndexNode*              m_V[3]          ; // Трійка вузлів
        IndexTrianglePtrVec     m_Child                 ; // Індексовані трикутники з нижчого
  рівня
        TrianglePtrVec          m_ChildOriginal         ; // Індексовані трикутники з
  ориганольноїтриангуляції
};
typedef std::vector<IndexTriangle*>IndexTrianglePtrVec;
typedef std::list<IndexTriangle>IndexTrianglesList;
```

Індексаційний вузол відрізняється від звичайного тим, що він містить власний тип зв'язків з суміжними вузлами. Його особливістю є те, що він не містить власної геометрії, а опирається на геометрію оригінального вузла.Також він містить контейнер з власним типом суміжного ребра, який є еквівалентним звичайному.

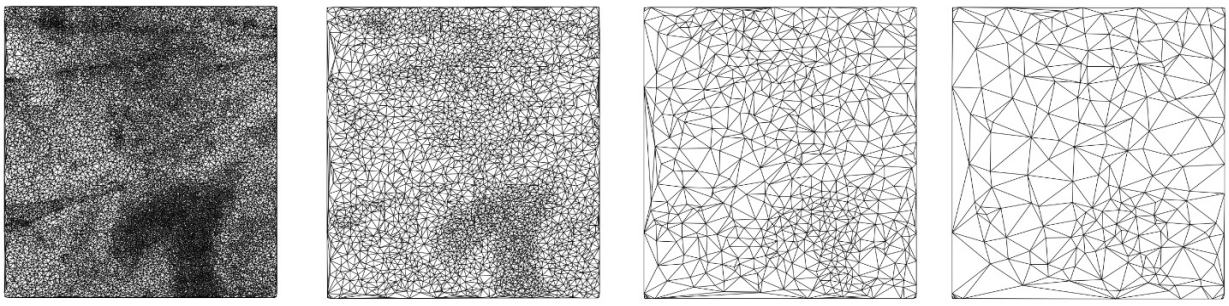
```
classIndexIncidentNode {
public:
        IncidentNode(Node* pNode, Triangle* pTriangle = nullptr);
private:
        IndexTriangle*  m_Triangle      ; // Трикутник, праворуч від ребра
        IndexNode*      m_Node                  ; // Вузол, до якого прямує ребро
};
typedef std::vector<IndexIncidentNodes>IndexIncidentNodes;
classIndexNode{
public:
        IndexNode(Node* pOriginal);
private:
        Node*                   m_OriginalNode          ; // Вузол з оригінальної триангуляції
        IndexIncidentNodes      m_IncidentNodes         ; // Ребра, що виходять з вузла
};
typedef std::set<IndexNode*>IndexNodes;
```

Для організації роботи з індексацією було створено відповідний клас, що маршрутизує пошук між різними рівнями індексаційноїтриангуляції, та, якщо він вдало відбувся, повертає елемент індексованої дискретної моделі.

```
classIndexation{
public:
        Indexation(const TIN&triangulation);
        // Метод виконує пошук оригінального трикутника по точці
        boolFind(const Point2d&pnt, constTriangle*&pFindedTriangle)const;
private:
        static UINT Build(Indexation* idx);         \\ функція для виклику побудови для підтримки
```

```
    роботи з afxwin в потоці
        constTIN&       m_OriginalTriangulation;// Індексована тріангуляції
        CWinThread*     m_BuildThread           ;// Потік для виконання побудови
        IndexationLevels        m_IndexationLevels      ;// Контейнер індексаційнихтріангуляцій
};
Indexation::Indexation(const TIN&triangulation)
  : m_OriginalTriangulation(triangulation)
  ,m_Thread(AfxBeginThread(&Build,this,THREAD_PRIORITY_NORMAL,0,CREATE_SUSPENDE
D))
{}
bool Indexation::Find(const Point2d&pnt, constTriangle*&pTriangle,
constIndexTriangle*pHintTriangle)const{
        IndexationLevels::const_iteratorindexLevelIt = m_IndexationLevels.begin()
        IndexTriangle* pFindedIndexTriangle= nullptr, * pHintTriangle = nullptr;
        while (indexLevelIt != m_IndexationLevels.end()){
                const IndexTIN&indexLevel = **indexLevelIt;
                if(!indexLevel.IsBuilded()){
                        // Якщо для поточного рівня ще не завершена побудова, перейти до
                наступного
                        ++indexLevelIt;
                        continue;
                }
                if(!indexLevel.Find(pnt, pFindedIndexTriangle, pHintTriangle) // Запуск пошуку
                        return false;
                pFindedIndexTriangle= pFindedIndexTriangle->GetChildByPoint(pnt);
                if(!pFindedIndexTriangle)
                        return false;
                pHintTriangle = pFindedIndexTriangle;  // Якщо трикутник знайти вдалось,
зберегти
                ++indexLevelIt;                         // його в якості зерна пошуку в наступному рівні
        }
        if(!pFindedIndexTriangle)
        returnfalse;
        pTriangle = pFindedIndexTriangle->GetChildOriginalByPoint(pnt);
        return true;
}
```

Як видно з прикладу коду вище, використано інтерфейс MFC та його стандартну функцію (Afx Begin Thread), що дозволяє винести виконання статичних функцій в окремий потік, щоб забезпечити можливість роботи з дискретною моделлю поки індексаційна структура не побудована.

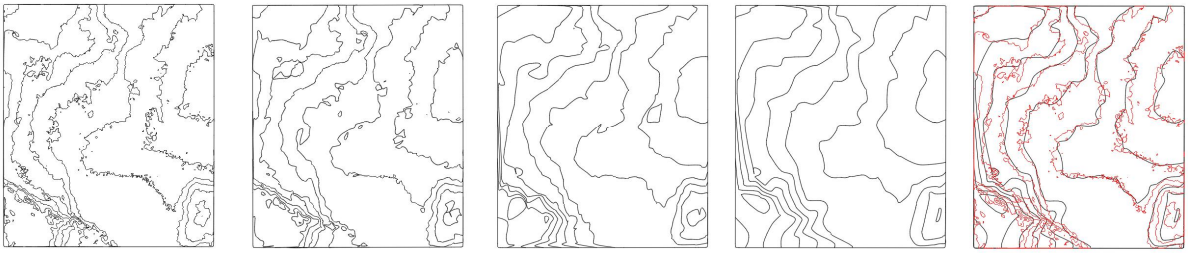


(а) (б) (в) (г)

Рис. 7. Результат побудови основної індексацій них тріангуляцій для генерованого набору точок. В результаті обробки оригінальної (а) побудовано проміжні (б, в) та основну (г) індексаційні тріангуляції.

Розглянемо результат побудови основної та проміжних індексаційних тріангуляцій на основі оригінальної (рис. 7). Наведена оригінальна тріангуляція (рис. 7.а) розміром 39912 трикутників. Також наведено топологію зв’язків індексаційної тріангуляції після виконання 5 ітерацій (рис. 7.б)

– 6848 трикутників; після 10 ітерацій (рис. 7.в) – 1212 трикутників; після 13 ітерацій (рис.7.г) – 426 трикутників. Продуктивність алгоритму побудови поки не була належно оцінена, але, з наявних даних відомо, що поточна реалізація забезпечує скорочення кількості трикутників в індексацій ній триангуляції відносно індексованої на 45-50% за ітерацію при розмірі індексованої від однієї тисячі та значно падає (до 10-15%) при зменшенні до 300-400.

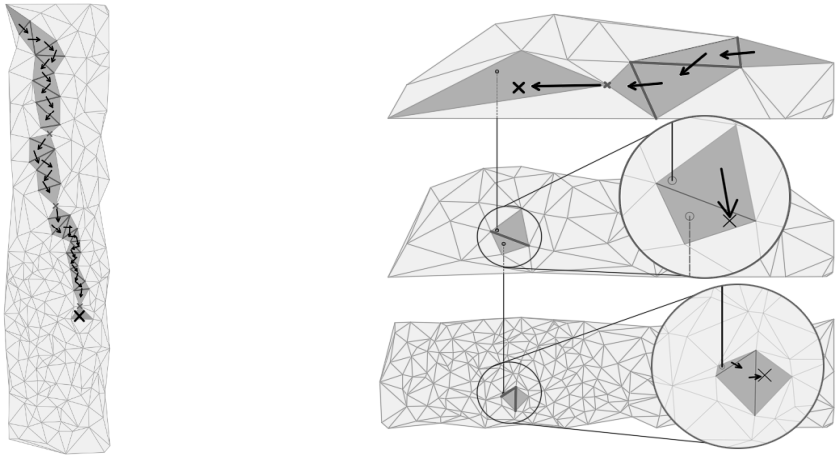


(а) (б) (в) (г) (г)

Рис. 8. Ізолінії для кожної з триангуляційних сіток (рис.7): оригінальна триангуляційна сітка (а) проміжні рівні ієрархічної триангуляції (б, в) та основний рівень ієрархічної триангуляційної сітки (г). Також наведено порівняння ізоліній (г) оригінальної (червоні) та основної в індексаційній структурі (чорні)

На рис.8 наведено ізолінії для оригінальної, проміжних рівнівпобудованої ієрархічної триангуляційної сітки та основної. Як видно на рисунку (рис.8.г), ізолінії оригінальної та верхньої індексаційної тріангуляції відрізняються не значно, з врахуванням рівня спрощення з 39912 до 426 трикутників, що свідчить про вдале збереження морфології триангуляційної моделі.

На рис.9 наведено приклад виконання пошукового запиту засобами реалізованої TIN. Пошуковий запит успішно виконано завдяки послідовному перегляду понад 32-х трикутників, переходу через 28 ребер та 3 вузли. Для порівняння, застосування індексаційної структури (рис.9б) дозволило виконати аналогічний запит з переглядом 9-ти трикутників, 5-ти переходів через ребра, одного переходу через вузол та 2 переходів в деревовидній структурі зв’язків між індексаційними трикутниками до індексованих. Очевидно, використання розробленої структури скорочує кількість звернень до елементів дискретизації, що, в свою чергу, дозволяє висунути припущення про вплив застосування структури на швидкодію опрацювання запитів на пошук елементів дискретизації.



(а) (б)

Рис. 9. Ілюстрація виконання пошуку в рамках області оригінальної триангуляційної сітки (а) та виконання того ж пошукового запиту з використанням індексаційної структури (б)

Зібраних даних про результати роботи розробленої індексаційної структури не достатньо для проведення її обґрунтованої оцінки. Спроби використання структури показують її перспективність, але і демонструють окремі проблеми. Розроблений ітеративний алгоритм побудови структури

потребує дослідження і вдосконалення способів оцінки поточного стану основної індексаційної триангуляції та розробки критеріїв зупинки побудови. Також, розроблений алгоритм потребує аналізу оптимальності побудови та оцінки його алгоритмічної складності, згідно з методологією нотації Ландау, оскільки, подібна оцінка не була проведена в рамках даної роботи. Згадані проблеми, окреслюють можливі напрямки для продовження досліджень розробленої індексаційної структури, та перспективні напрямки для її розвитку.

Висновки

В даній роботі реалізовано застосування індексаційної структури на основі ієрархічної триангуляційної сітки (HTM) до задачі індексації пошуку елементів планарної дискретній моделі (триангуляційної сітки). Розроблена індексаційна структура базується на дискретних елементах, подібних до оригінального (індексованого) об'єкта, що, завдяки наявності горизонтальних зв'язків між дискретними елементами індексаційної структури, забезпечує використання загальних механізмів пошуку в триангуляції. Застосування спрощення триангуляційної сітки в механізмі побудови рівнів індексаційної структури дозволило зберігати морфологію індексаційних сіток, що дасть можливість диференціювання деталізації відображення триангуляції в залежності від масштабу, шляхом використання для візуалізації різних рівнів ієрархічної триангуляції. Реалізований алгоритм спрощення забезпечує стабільне ~40% скорочення кількості трикутників між рівнями індексаційної структури при кількості понад 1000 шт., та падаєдо <20% при кількості трикутників ~500шт, відповідно, падіння збіжності нижче 20% прийнято за умову завершення побудови індексаційної структури.

Перелік використаних джерел

[1] Zhang, X and Du, Z.,"Spatial Indexing",The Geographic Information Science & Technology Body of Knowledge (4th Quarter 2017 Edition), John P. Wilson (ed). https://doi.org/10.22224/gistbok/2017.4.12

[2] M. Andrew, "Another Efficient Algorithm for Convex Hullsin Two Dimensions", Info. Proc. Letters 9, 216-219 (1979). https://doi.org/10.1016/0020-0190(79)90072-3

[3] D. H. Eberly, "Triangulation by Ear Clipping", Geometric Tools, LLC.,www.geometrictools.com, 1998 https://doi.org/10.1145/282918.282923

[4] Guibas L, Stolfi J,"Primitives for the manipulation of general subdivision sand the computation of Voronoi",ACM Transactionson Graphics 1985 Volume 4, Issue 2, p 107,doi:10.1145/282918.282923

[5] Dutton, G. "Encoding and handling geospatial data with hierarchical triangular meshes",Advancesin GIS Research II. London: Taylor&Francis, 1996, p 505-518

[6] Rigaux, P., Scholl, M., &Voisard, A. "Spatial Databases – with application to GIS",Morgan Kaufmann, San Francisco 2002, p 410.

[7] M. Bern, D. Eppstein, "Mesh generation and optimal triangulation", Computing in Euclidean geometry 1992, 1, p 23–90. https://doi.org/10.1142/9789814355858_0002

[8] de Berg, M., Cheong, O., van Kreveld, M. and Overmars, M. "Computational Ge-ometry: Algorithms and Applications", Springer, 3rd edition, ISBN 9783540847441(2008). https://doi.org/10.1007/978-3-540-77974-2

[9] Kunszt, P. Z.. Szalay. A. S., Csabai, I., Thakar. A. R. 2000, in ASP Conf.Ser.. Vol. 216. Astronomical Data Analysis Software and Systems IX. cds. N.Manset, C. Veillet, D. Crabtree (SanP'raucisco:ASP), 141

[10] P. Z. Kunszt, A. S. Szalay, and A. R. Thakar. The hierarchical triangular mesh. In Miningthesky, pages 631–637. Springer,2001. https://doi.org/10.1007/10849171_83

[11] S. Szalay, J. Gray, G. Fekete, P.Z. Kunszt, P. Kukol, and A. Thakar, "Indexing the Sphere with the Hierarchical Triangular Mesh," Micr. Res. Tech. Rpt.,MSR-TR-2005-123, 2005

[12] K.M. Górski, E. Hivon, A.J. Banday, B.D. Wandelt, F.K. Hansen, M. Reinecke, M. Bartelmann, HEALPix: a framework for high-resolution discretization and fast analysis of data distributed on the sphere, Astrophys. J. 622 (2) (2005) 759–771. https://doi.org/10.1086/427976

[13] O'Mullane, A. Banday, K. Gorski, P. Kunszt, and A. Szalay, "Splitting the sky-htm and healpix," in Miningthe Sky. Springer, 2000, pp. 638–648

[14] Michael Rilee, Niklas Griessbaum, Kwo-SenKuo, James Frew, and Robert Wolfe. 2020. STARE-based Integrative Analysis of Diverse Data Using Dask Parallel Programming Demo Paper. In Proceedings of ACMSIGSPATIAL conference (SIGSPATIAL'20). ACM, NewYork, NY, USA, 4pages. https://doi.org/10.1145/3397536.3422346

[15] Rilee M, Kuo K, Clune T, etal. Addressing the big-earth-data variety challenge with the hierarchical triangular mesh. 2016 IEEE International Conferenceon Big Data (Big Data); 2016 Dec 5-8; Washington, DC,USA: IEEE; p. 1006–1011. https://doi.org/10.1109/BigData.2016.7840700

[16] O. Esen, V. Tongur, I. B. Gundogdu HEALPix Mapping Technique and Cartographical Application May 2013 Conference: 26th International Cartographic Conference At: Dresden, Germany

[17] list class |Microsoft Learn.Retrieved from:https://learn.microsoft.com/en-us/cpp/standard-library/list-class

[18] set class | Microsoft Learn. Retrieved from:https://learn.microsoft.com/en-us/cpp/standard-library/set

Oleksandr Manyuk[1], Yaroslav Sokolovskyy[2]

[1] Department Computer Design System, Lviv Polytechnic National University, Ukraine, Lviv, S. Bandery street 12, E-mail: oleksandr.m.maniuk@lpnu.ua, ORCID 0009-0007-5673-1522

[2] Department Computer Design System, Lviv Polytechnic National University, Ukraine, Lviv, S. Bandery street 12, E-mail: yaroslav.i.sokolovskyi@lpnu.ua, ORCID 0000-0003-4866-2575

DEVELOPMENT OF ALGORITHM AND DATA STRUCTURE FOR 2D REGIONS DISCRETE MODEL ELEMENTS ACCELERATED SEARCH

Received: March 04, 2024 / Revised: March 20, 2024 / Accepted: April 01, 2024

Abstract. This paper is devoted to improving the search request processing productivity for planar discrete models used in engineering software. A data structure has been developed to accelerate the search for discretization elements based on a hierarchical triangular mesh. The developed indexing structure is built by a downward iterative algorithm, which constructs each new level of the hierarchy based on the previous level or indexed triangulation by simplifying it, which ensures that the morphology of the triangulation mesh is preserved throughout all levels of the hierarchical indexing structure. The developed building algorithm ensures the presence of tree-like connections between the levels of the hierarchical triangulation mesh, which allows downward navigation between geometrically close triangles. Search acceleration is achieved by performing a directed search in the top level of the indexing structure and then navigating between levels using downward links until the indexed triangle is found. The program implementation was carried out using C++17, and visualization of triangulation grids and isolines was carried out using the ObjectARX library. Based on the software implementation, an executive library was created.

Keywords: 2D regions discretization**,** accelerated search, discrete model, triangulated irregular network, UML diagram, ObjectARX library.