Науково-практичний звіт

ЗАДАЧА ТРІАНГУЛЯЦІЇ ПОВЕРХНІ ЗІРКОВОГО МНОГОГРАННИКА

А.К. Александрова, ММФ КНУ

Анотація. У роботі запропоновано узагальнений метод тріангуляції граней 3D - многогранників на основі алгоритму замітальної прямої. Розроблено підхід, що поєднує точність методу замітання зі стійкістю віялової тріангуляції, що дозволило вирішити проблему топологічної некоректності при проектуванні непланарних граней. Застосування методу Ньюелла для проекції та обробка вироджених випадків забезпечують ефективність алгоритму на рівні $O(N \log N)$.

Abstract. In the paper, we propose a generalized method for triangulating faces of 3D polyhedra based on the sweep-line algorithm. In this case, a hybrid approach has been developed combining the precision of the sweep-line method with the robustness of fan triangulation, which made it possible to solve the problem of topological incorrectness when projecting non-planar faces. The application of Newell's method for projection and handling of degenerate cases ensure the algorithm's efficiency at $O(N \log N)$.

1 Вступ

Постановка проблеми. У роботі розглядається проблема розбиття складних просторових многокутників (граней зіркових многогранників) на набір трикутників. Тріангуляція поверхонь є фундаментальною задачею комп'ютерної графіки, оскільки сучасні графічні прискорювачі (GPU) оптимізовані для обробки саме трикутних сіток. Особливої актуальності набуває задача обробки даних, що містять геометричні шуми — «вироджених» або непланарних граней, які часто виникають при процедурній генерації об'єктів, 3D-скануванні або фізичному моделюванні. Пошук ефективних алгоритмів, здатних коректно обробляти такі дані без структурних дефектів (дірок, самоперетинів) та зі збереженням топологічної цілісності моделі, є важливою науково-практичною задачею [1]. Це також стосується задач автоматизованого проектування (CAD) та геоінформаційних систем (GIS), де точність представлення поверхні є критичною [2, 3]. Тому виникає необхідність розробки методів, які поєднують теоретичну ефективність з практичною стійкістю до похибок вхідних даних.

Аналіз останніх досліджень. На сьогоднішній день для розв'язання задачі тріангуляції полігонів використовують низку підходів, які класифікують за алгоритмічною складністю та вимогами до вхідних даних.

Фундаментальні дослідження у галузі обчислювальної геометрії, присвячені проблемі декомпозиції полігональних областей на симплекси, базуються на конструктивній теоремі про існування двох неперетинних діагоналей у довільному простому N-кутнику. Класичний ітеративний метод, відомий як відрізання вух, спирається на пошук локально опуклої вершини, діагональ якої повністю належить внутрішній області полігона. Алгоритмічна складність наївної реалізації цього підходу визначається необхідністю перевірки перетину потенційної діагоналі з множиною всіх інших ребер графа, що призводить до асимптотичної оцінки $O(N^3)$. Застосування оптимізованих структур даних для запитів видимості дозволяє знизити цю оцінку до $O(N^2)$ у найгіршому випадку, а використання методів геометричного хешування та локалізації на сітці дає середній час виконання $O(N)$. Проте, теоретична верхня межа складності залишається квадратичною, що робить даний клас

алгоритмів обчислювально неприйнятним для обробки моделей у системах реального часу, де кількість вершин може перевищувати 10^4 [4, 5]. Теоретично можлива детермінована тріангуляція за лінійний час $O(N)$ завдяки алгоритму Чазелла, проте його практична реалізація є надзвичайно складною через велику кількість крайніх випадків [19].

Асимптотично оптимальним рішенням для загального випадку є методи, засновані на декомпозиції області замітальною прямою, що досягають нижньої межі складності $O(N \log N)$. Процес регуляризації полігона передбачає його розбиття на множину y - монотонних підполігонів шляхом введення діагоналей, що виходять із вершин, в яких змінюється напрямок монотонності реберного ланцюга. Часова складність цього етапу визначається операцією сортування вершин, яка вимагає $O(N \log N)$ операцій, та підтримкою динамічної структури (збалансованого бінарного дерева пошуку), операції вставки та видалення в якій виконуються за час $O(\log K)$, де K — поточна кількість активних ребер.

Альтернативою детермінованим методам є рандомізований підхід Зейделя, який використовує трапецієподібну декомпозицію для досягнення аналогічної середньої складності [20]. Другий етап алгоритму — тріангуляція отриманих монотонних частин — виконується за лінійний час $O(N)$ за допомогою жадібного алгоритму з використанням стекової пам'яті, оскільки вершини монотонного ланцюга вже є впорядкованими. Таким чином, загальна часова складність методу $T(N) = O(N \log N) + O(N) = O(N \log N)$ є оптимальною [6, 7, 8].

Пошук алгоритмів з детермінованою лінійною складністю $O(N)$ призвів до розробки алгоритму Чазелла, який базується на теоремі про розбиття планарного графа, проте через надзвичайно високу константу при асимптотиці та складність імплементації він рідко застосовується на практиці. Натомість для специфічних топологічних класів полігонів широко використовуються евристичні методи, зокрема віялова тріангуляція, що будує множину симплексів, інцидентних одній фіксованій вершині. Обчислювальна вартість такої процедури є строго лінійною $O(N)$ і не вимагає допоміжних структур даних. Однак область коректності цього алгоритму обмежується класом зіркових полігонів, для яких перетин ядра та множини вершин є непорожнім [9, 10]. Проблеми генерації якісних сіток детально розглянуті в огляді Берна, де наголошується на важливості вибору правильної стратегії тріангуляції [21].

Окремий клас задач становить тріангуляція граней, заданих у тривимірному евклідовому просторі $\mathbb{R}^3$, що вимагає побудови гомеоморфного відображення поверхні на площину $\mathbb{R}^2$. Для відновлення поверхонь зі складною топологією (що є узагальненням нашої задачі) часто застосовують методи на основі діаграм Вороного, запропоновані Аментою та Берном, які дозволяють гарантувати топологічну коректність [22]. Стандартні методи ортогональної проекції шляхом відкидання однієї з координат є чисельно нестабільними для граней, нормаль яких майже ортогональна вектору проектування, що призводить до виродження координат і втрати інформації про орієнтацію. Більш робастним підходом є застосування методу Ньюелла, що мінімізує метричні спотворення. Крім того, як показано в роботах Кеттнера та Шевчука, використання стандартної арифметики з плаваючою комою (стандарт IEEE 754) при обчисленні геометричних предикатів орієнтації призводить до порушення аксіом евклідової геометрії та виникнення топологічних сингулярностей. Це обґрунтовує необхідність застосування точних геометричних обчислень [11, 12, 13].

2 Основна частина

2.1 Постановка задачі тріангуляції поверхні зіркового многогранника

Нехай задано многогранник P в евклідовому просторі $(\mathbb{R}^3, d)$, де d - евклідова метрика. Границя многогранника ∂P представлена скінченною множиною граней $F = \{f_1, f_2, \ldots, f_m\}$. Кожна грань $f_i \in F$ визначається циклічно впорядкованою послідовністю вершин $V_i = \{v_1, v_2, \ldots, v_{k_i}\}$, де $v_j \in \mathbb{R}^3$, а $k_i \geq 3$ — кількість вершин у грані.

Зауважимо, що внаслідок похибок обчислень, особливостей генерації або моделювання, вершини V_i можуть не належати одній площині $\alpha \subset \mathbb{R}^3$, тобто $\exists v_j \in V_i : \text{dist}(v_j, \alpha) > \epsilon$.

Задача полягає у знаходженні множини трикутників $\mathcal{T} = \{\Delta_1, \Delta_2, \ldots, \Delta_n\}$, що задовольняє наступним умовам:

1. **Умова покриття:** Об'єднання всіх трикутників має повністю покривати поверхню многогранника:
$$\bigcup_{\Delta \in \mathcal{T}} \Delta = \bigcup_{f \in F} f = \partial P \tag{1}$$

2. **Умова узгодженості:** Для будь-яких двох трикутників $\Delta_a, \Delta_b \in \mathcal{T}$, де $a \neq b$, їх перетин $\Delta_a \cap \Delta_b$ є або порожньою множиною, або спільною вершиною, або спільним ребром. Перетин внутрішніх областей трикутників порожній:
$$\text{int}(\Delta_a) \cap \text{int}(\Delta_b) = \emptyset \tag{2}$$

3. **Умова вершин:** Множина вершин тріангуляції співпадає з множиною вершин многогранника P. Введення додаткових точок (точок Штейнера) не допускається.

Оскільки класичні алгоритми тріангуляції визначені для площини $\mathbb{R}^2$, задачу розбито на підзадачі тріангуляції кожної окремої грані f_i. Для цього необхідно побудувати відображення (проекцію) $\pi : \mathbb{R}^3 \to \mathbb{R}^2$, таке що проекція грані $f_i' = \pi(f_i)$ залишається простим многокутником (тобто границя f_i' не містить самоперетинів), після чого знайти тріангуляцію $\mathcal{T}_i$ для f_i' та відобразити її назад у простір $\mathbb{R}^3$.

2.2 Постановка задачі генерації зіркового многогранника

Нехай $N \in \mathbb{N}$ визначає кількість вершин. Необхідно побудувати замкнену поверхню $\mathcal{M} = (V, F)$, де $V = \{v_1, \ldots, v_N\} \subset \mathbb{R}^3$ — множина вершин, а F — множина граней. При цьому має виконуватися умова зірковості: $\exists$ точка O(центр), така що будь-який промінь, що виходить з O, перетинає поверхню $\mathcal{M}$ рівно в одній точці:

$$\forall \mathbf{u} \in S^2, \quad |\{O + t\mathbf{u} \mid t > 0\} \cap \mathcal{M}| = 1$$

2.3 Основні означення, теореми, леми, твердження

Введемо основні об'єкти дослідження та їх властивості.

Означення 1 (Простий многокутник). Многокутник P називається простим, якщо його границя ∂P не має самоперетинів, а суміжні ребра перетинаються лише у спільних вершинах.

Означення 2 (y-монотонний многокутник). Простий многокутник P називається монотонним відносно осі y, якщо перетин будь-якої прямої, перпендикулярної до осі y, з многокутником P є зв'язним (тобто є відрізком або точкою, або порожньою множиною).

Означення 3 (Зірковий многокутник). Многокутник P називається зірковим (star-shaped), якщо існує хоча б одна точка z у його внутрішній області ($z \in P$), така що для будь-якої точки $p \in P$ відрізок $[z, p]$ повністю належить P. Множина всіх таких точок z утворює ядро (kernel) многокутника.

Означення 4 (Зірковий многогранник). Многогранник $P \subset \mathbb{R}^3$ називається зірковим, якщо

$$\exists C \in P \forall A \in P : [C, A] \subseteq P \tag{3}$$

Множина всіх таких точок C називається ядром многогранника P.

Теорема 1 (Існування тріангуляції). Будь-який простий многокутник з N вершинами можна розбити на $N - 2$ трикутники за допомогою $N - 3$ діагоналей, що не перетинаються.

Лема 1 (Критерій монотонності). Простий многокутник P є y-монотонним тоді і тільки тоді, коли він не має вершин типу «розбиття» (split) або «злиття» (merge).

Коментар: Саме на цій лемі базується перший етап алгоритму тріангуляції — усунення вершин split/merge шляхом додавання діагоналей, що перетворює довільний многокутник на набір монотонних.

Теорема 2 (Тріангуляція монотонного многокутника). Простий y-монотонний многокутник з N вершинами може бути тріангульований за час $O(N)$ за допомогою жадібного алгоритму.

Теорема 3 (Коректність віялової тріангуляції) Якщо многокутник P є опуклим або зіркоподібним відносно першої вершини v_0, то множина трикутників, утворена з'єднанням v_0 з усіма іншими вершинами v_i (де $i = 1 \dots N - 2$), утворює коректну тріангуляцію P без накладань.

Коментар: Це обґрунтовує корекність використання Fallback-механізму. Оскільки дані (многогранники) генеруються як двоїсті до опуклих, вони локально зберігають зіркоподібну структуру, що робить метод Fan Triangulation допустимим.

2.4 Побудова розв'язку задачі тріангуляції

Нехай задано грань f як циклічно впорядковану множину вершин $V = \{v_1, v_2, \dots, v_N\}$ у просторі $\mathbb{R}^3$, де $v_i = (x_i, y_i, z_i)$. Оскільки класичні алгоритми тріангуляції визначені для евклідової площини, першим етапом розв'язку є побудова відображення $\pi : \mathbb{R}^3 \to \mathbb{R}^2$, яке мінімізує метричні спотворення.

Для визначення площини проекції обчислюється вектор нормалі $\vec{n} = (n_x, n_y, n_z)$ до апроксимуючої площини грані за методом Ньюелла:

$$\begin{aligned} n_x &= \sum_{i=1}^{N} (y_i - y_{i+1})(z_i + z_{i+1}), \\ n_y &= \sum_{i=1}^{N} (z_i - z_{i+1})(x_i + x_{i+1}), \\ n_z &= \sum_{i=1}^{N} (x_i - x_{i+1})(y_i + y_{i+1}), \end{aligned} \tag{4}$$

де $v_{N+1} = v_1$. Площина проекції обирається шляхом відкидання координати, що відповідає $\max(|n_x|, |n_y|, |n_z|)$. Отримані 2D-координати вершин позначимо як $p_i = (u_i, w_i)$.

Задача розбиття неопуклого многокутника P на множину трикутників зводиться до задачі його декомпозиції на монотонні частини. Многокутник є w-монотонним, якщо перетин його з будь-якою прямою $w = const$ є зв'язним. Для реалізації декомпозиції вершини

класифікуються на основі локальної геометрії. Нехай p_{i-1} та p_{i+1} — сусіди вершини p_i. Введемо предикат орієнтації $\mathrm{Or}(A, B, C)$. Позначимо $\varphi :=$ кут між точками A, B, C.

$$\mathrm{Or}(A,B,C) = \begin{cases} +1, & \text{якщо } 0 < \varphi < \pi, \\ -1, & \text{якщо } \pi < \varphi < 2\pi, \\ 0, & \text{якщо } \varphi = 0 \text{ або } \varphi = \pi. \end{cases} \tag{5}$$

Вершини, що порушують монотонність, - це вершини типу Split та Merge. Вони визначаються наступними умовами:

1) **Вершина розбиття (Split vertex):**

$$w_{i-1} > w_i \quad \text{та} \quad w_{i+1} > w_i \quad \text{та} \quad \mathrm{Or}(p_{i-1}, p_i, p_{i+1}) = -1 \tag{6}$$

(внутрішній кут перевищує π, обидва сусіди лежать вище).

2) **Вершина злиття (Merge vertex):**

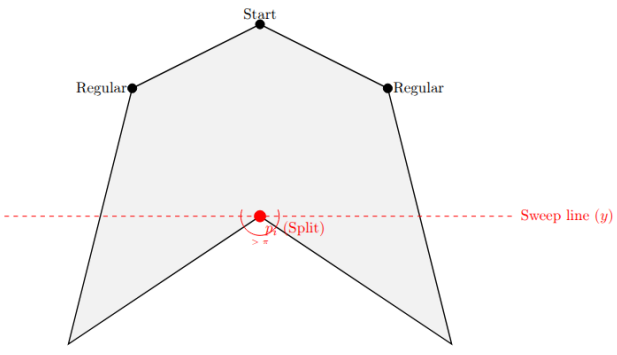
$$w_{i-1} < w_i \quad \text{та} \quad w_{i+1} < w_i \quad \text{та} \quad \mathrm{Or}(p_{i-1}, p_i, p_{i+1}) = -1 \tag{7}$$

(внутрішній кут перевищує π, обидва сусіди лежать нижче).

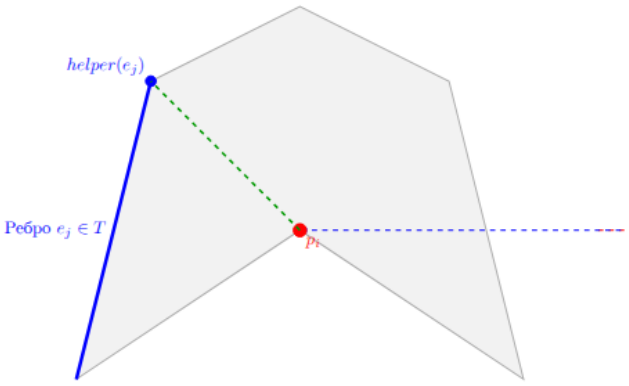
Алгоритм побудови тріангуляції реалізується таким чином:

1. Виконується сортування вершин за спаданням координати w (метод замітальної прямої).

2. Під час проходження замітальної прямої підтримується впорядкована структура статусів T (активних ребер). Для кожної вершини типу *Split* знаходиться найближче ребро $e_j \in T$, що лежить ліворуч від p_i, і додається діагональ $(p_i, \mathrm{helper}(e_j))$.

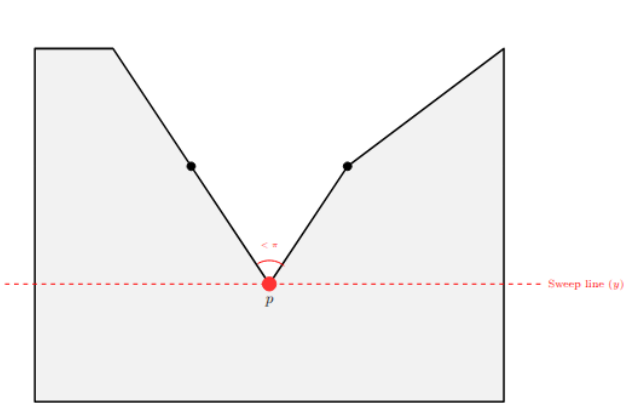


(а) Вершина типу Split

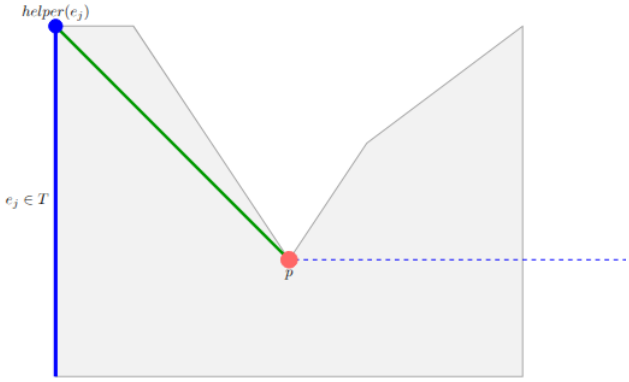


(б) Додавання діагоналі

3. Аналогічно для вершин типу *Merge* додається діагональ до відповідної вершини, що лежить вище.



(а) Вершина типу Merge



(б) Додавання діагоналі

4. Після додавання всіх діагоналей многокутник P розпадається на множину монотонних багатокутників $\{M_1, \ldots, M_k\}$.

5. Кожен M_j тріангулюється за один прохід.

Fallback algorithm. У випадку, якщо внаслідок похибок вхідних даних проекція $\pi(f)$ містить самоперетини (тобто не є простим багатокутником), коректність алгоритму замітання порушується. Тоді розв'язок будується альтернативним методом віялової тріангуляції:

$$\mathcal{T}_f = \bigcup_{i=2}^{N-1} \triangle(v_1, v_i, v_{i+1}) \tag{8}$$

Геометрично такий підхід гарантує топологічну замкненість поверхні (відсутність "дірок"), навіть якщо грань f є непланарною.

2.5 Побудова розв'язку задачі генерації

Побудова розв'язку базується на властивостях полярної двоїстості опуклих тіл, описаних у класичних працях з обчислювальної геометрії [6]. Нехай S^2 — одинична сфера з центром у початку координат O.

Етап 1. Дискретизація сферичної поверхні. Генерується множина $P = \{p_1, p_2, \ldots, p_N\}$ з N випадкових точок, рівномірно розподілених на S^2. Координати точок визначаються у сферичній системі:

$$\begin{aligned} p_i &= (r\sin\theta_i\cos\phi_i,\ r\sin\theta_i\sin\phi_i,\ r\cos\theta_i), \\ \theta_i &= \arccos(1-2u_i), \quad \phi_i = 2\pi v_i, \end{aligned} \tag{9}$$

де u_i, v_i — незалежні випадкові величини з рівномірним розподілом на $[0,1]$, $r = 1$.

Етап 2. Побудова первинної сітки (Primary Mesh). Будується опукла оболонка $\mathcal{H} = \text{ConvexHull}(P)$. Оскільки $p_i \in S^2$, всі точки належать границі $\partial\mathcal{H}$. Для побудови оболонки використовується алгоритм *QuickHull*, складність якого для сферичного розподілу становить $O(N \log N)$ [14]. Поверхня $\partial\mathcal{H}$ є тріангуляцією, що складається з множини граней $F_{\mathcal{H}} = \{t_1, \ldots, t_M\}$, де кожна грань t_j є трикутником. Згідно з формулою Ейлера для планарних графів, кількість граней $M \approx 2N$ [15].

Етап 3. Побудова двоїстого многогранника (Dual Mesh). Виконується перетворення $\mathcal{H}$ для побудови многогранника $\mathcal{H}^*$ [3].

1. Вершини двоїстого многогранника $V^* = \{v_1^*, \ldots, v_M^*\}$ визначаються як центри граней первинної сітки. Для забезпечення планарності граней у випадку сферичної генерації використовується центр описаного кола (circumcenter) трикутника $t_j = \triangle(A, B, C)$:

$$v_j^* = \text{Circumcenter}(A, B, C) \tag{10}$$

2. Грані двоїстого многогранника F^* будуються навколо вершин первинної сітки. Для кожної вершини $p_k \in P$ формується грань f_k^*, вершинами якої є центри трикутників, інцидентних p_k.

Програмна реалізація базується на структурах даних бібліотеки CGAL [16].

Етап 4. Стохастична деформація (Perturbation). Для моделювання складної геометрії до кожної вершини $v \in V^*$ застосовується варіація радіус-вектора:

$$v' = v \cdot (1 + \delta(\vec{v})), \quad \delta(\vec{v}) > -1 \tag{11}$$

Функція збурення $\delta(\vec{v})$ задається як суперпозиція гармонічних функцій (шум Перліна), що широко використовується у процедурному моделюванні [1]:

$$\delta(x, y, z) = A \cdot \sin(\omega x + \psi_x) \cos(\omega y + \psi_y) \sin(\omega z + \psi_z) \tag{12}$$

де A — амплітуда, ω — частота, ψ — випадкова фаза. Таке перетворення зберігає топологічну зв'язність та властивість зірковості відносно O, але порушує планарність граней, створюючи необхідні умови для тестування алгоритму тріангуляції.

3 Обґрунтування оцінки складності алгоритму

Теорема 1. Часова складність запропонованого алгоритму тріангуляції для грані, що складається з N вершин, становить $O(N \log N)$.

Доведення. Розглянемо алгоритмічну складність кожного етапу окремо.

1. **Проекція (Метод Ньюелла).** Для обчислення вектора нормалі $\vec{n}$ виконується один прохід по границі грані, під час якого обчислюються суми добутків координат суміжних вершин:

$$T_{proj}(N) = \sum_{i=1}^{N} O(1) = O(N) \tag{13}$$

2. **Сортування подій (Event Queue).** Для ініціалізації черги подій алгоритму замітальної прямої необхідно впорядкувати N вершин за y-координатою.

$$T_{sort}(N) = O(N \log N) \tag{14}$$

3. **Обробка подій (Sweep-line Process).** Алгоритм послідовно обробляє N подій (вершин). На кожному кроці виконуються операції зі структурою статусів T (активних ребер), яка реалізована як збалансоване бінарне дерево пошуку (червоно-чорне дерево у `std::set`).

   Основні операції (пошук найближчого лівого ребра, вставка нового ребра та видалення старого) мають складність $O(\log K)$, де K — поточна кількість активних ребер. $K \leq N \implies$ вартість обробки однієї події не перевищує $O(\log N)$.

$$T_{sweep}(N) = \sum_{i=1}^{N} O(\log N) = O(N \log N) \tag{15}$$

4. **Тріангуляція монотонних частин.** Після додавання діагоналей многокутник розпадається на набір y-монотонних многокутників. Тріангуляція кожного з них виконується жадібним алгоритмом (стековий метод), який обробляє кожну вершину рівно один раз (додавання в стек та видалення зі стека):

$$T_{mono}(N) = O(N) \tag{16}$$

5. **Віялова тріангуляція.** У випадку виявлення топологічної некоректності (самоперетину проекції), основний алгоритм переривається, і запускається метод віяла. Він виконує один прохід по вершинах, з'єднуючи опорну вершину з іншими:

$$T_{fan}(N) = O(N) \tag{17}$$

Загальна оцінка. Асимптотична складність алгоритму визначається наступним чином:

$$T(N) = T_{proj} + T_{sort} + T_{sweep} + T_{mono} = O(N \log N) \tag{18}$$

Просторова складність. Для зберігання структури DCEL (вершини, напівребра, грані) та черги подій потрібно виділити пам'ять, пропорційну кількості вершин. Кількість доданих діагоналей не перевищує $N - 3$.

$$S(N) = O(N) \tag{19}$$

Теорема 2. Нехай P — многогранник із загальною кількістю вершин N. Тоді часова складність тріангуляції всієї поверхні ∂P запропонованим методом становить $O(N \log N)$.

Доведення. Нехай $V, F = \{f_1, f_2, \ldots, f_m\}$ - множини вершин і граней многогранника відповідно. Позначимо $k_i := \#\{v \in V | v \in f_i\}$. Оскільки кожне ребро належить рівно двом граням (для замкненої поверхні), маємо співвідношення:

$$\sum_{i=1}^{m} k_i = 2E \tag{20}$$

За формулою Ейлера для поверхонь роду 0, $E \approx 3N$ (асимптотично $E = O(N)$). Час тріангуляції однієї грані f_i запропонованим методом становить $T(f_i) = O(k_i \log k_i)$ (за теоремою 1). Тоді час для всього многогранника :

$$T_{total} = \sum_{i=1}^{m} O(k_i \log k_i) \tag{21}$$

Оскільки для будь-якої грані $k_i \leq N$, то $\log k_i \leq \log N$. Тоді

$$T_{total} \leq O(\log N) \cdot \sum_{i=1}^{m} k_i \tag{22}$$

Врахувавши (13) та $E = O(N)$, отримаємо:

$$T_{total} \leq O(\log N) \cdot O(N) = O(N \log N) \tag{23}$$

Просторова складність. Алгоритм обробляє грані послідовно. Максимальна пам'ять, необхідна в один момент часу, залежить від розміру найбільшої грані k_{max}. Для зберігання самого многогранника (структура DCEL або HalfedgeDS) потрібно $O(N)$ пам'яті. Тоді

$$S(N) = O(N) + O(k_{max}) = O(N) \tag{24}$$

4 Практична частина

Особливості програмної реалізації. Програму розроблено за допомогою мови C++ (стандарт C++17). Для забезпечення надійності геометричних обчислень та роботи з числами довільної точності використано бібліотеку **CGAL** [16]. Однією з особливостей реалізації є використання ядра `Exact_predicates_exact_constructions_kernel`. Це дозволяє

уникнути помилок округлення, що є критичним для коректної роботи алгоритму замітальної прямої. Графічний інтерфейс користувача (GUI) реалізовано за допомогою бібліотеки **Qt5** [17].

Опис основних функцій (модулів) програмної реалізації. Архітектура програми складається з трьох логічних модулів:

- `generate_star_mesh(int num_points, Polyhedron& mesh)`: генерація тестових даних. Реалізована відповідно до опису розв'язку задачі генерації.
- `triangulate_facet_sweepline(Facet_handle f, Polyhedron& mesh)`: основна функція алгоритму. Виконує проекцію грані на 2D-площину методом Ньюелла, будує локальну структуру DCEL, сортує події та запускає процес замітання для декомпозиції полігона на монотонні частини. У разі виявлення неправильної топології повертає `false`, ініціюючи запуск алгоритму Fan Triangulation.
- `full_sweep_triangulation(Polyhedron& mesh)`: Ітерує по всіх гранях моделі. Будує тріангуляцію всього многогранника.

Характеризація вводу-виводу даних. Програма підтримує два режими роботи з вхідними даними:

1. **Автоматична генерація:** Користувач задає кількість опорних точок N (наприклад, $N = 10^4$). Програма генерує зірковий многогранник із N вершинами. Це дозволяє тестувати ефективність алгоритму на великих масивах даних.
2. **Завантаження з файлу:** Підтримується стандартний формат **.OFF** (Object File Format) [18], що дозволяє завантажувати довільні поліедральні моделі (наприклад, правильний додекаедр або куб).

Результат роботи (тріангульована поверхня) візуалізується та автоматично зберігається у файл `result.off`.

Можливості, програмне та технічне забезпечення програмної реалізації. Програма дозволяє:

- Тріангулювати поверхні зіркових многогранників.
- Візуалізувати початкову (полігональну) та кінцеву (тріангульовану) сітки.
- Обробляти моделі розміром до 10^5 вершин за прийнятний час (< 5 сек).

Технічні вимоги: Операційна система Linux/Windows/macOS, підтримка OpenGL 3.3+.

Необхідне ПЗ: Компілятор C++ (GCC/Clang/MSVC), бібліотеки CGAL 5.x, Qt 5.15, CMake 3.10+.

5 Висновки

У ході виконання роботи отримано наступні результати:

1. Експериментально підтверджено теоретичну складність $O(N \log N)$ та встановлено критичну роль точної арифметики (Exact Kernel) для уникнення помилок округлення на щільних сітках.

2. Доведено, що для тріангуляції стохастичних 3D-об'єктів необхідним є гібридний підхід. Застосування віялової тріангуляції як механізму відновлення для алгоритму замітальної прямої гарантує цілісність тріангуляції навіть при наявності непланарних граней.

Проблеми та перспективи. Основним обмеженням поточного підходу є залежність від якості 2D-проекції. Метод Ньюелла мінімізує, але не усуває спотворення для сильно викривлених граней, що може призводити до появи вироджених трикутників. Перспективним напрямком є відмова від застосування проєкції на користь алгоритмів, що працюють безпосередньо на поверхні сфери (сферична тріангуляція Делоне) або використання об'ємних методів (тетраедралізація з подальшим вилученням поверхні).

Шляхи покращення ефективності. Для оптимізації програмного комплексу для роботи з дуже великими даними ($N > 10^6$) доцільно:

1. Впровадити паралельну обробку незалежних граней (наприклад, за допомогою OpenMP), оскільки алгоритм тріангуляції є паралелізованим.

2. Замінити динамічні структури на основі вузлів (`std::set`, `std::list`) на кеш-ефективні структури (наприклад, плоскі масиви або B-дерева), що дозволить зменшити фрагментацію пам'яті.

Література

[1] Rogers D. F. Procedural Elements for Computer Graphics. 2nd ed. McGraw-Hill, 1998.

[2] Botsch M. et al. Polygon Mesh Processing. AK Peters/CRC Press, 2010.

[3] Edelsbrunner H. Geometry and Topology for Mesh Generation. Cambridge University Press, 2001.

[4] Meisters G. H. Polygons have ears. *American Mathematical Monthly*. 1975. Vol. 82.

[5] ElGindy H. et al. Slicing an ear in linear time. *Pattern Recognition Letters*. 1993. Vol. 14.

[6] Preparata F. P., Shamos M. I. Computational Geometry: An Introduction. Springer-Verlag, 1985.

[7] Garey M. R. et al. Triangulating a simple polygon. *Information Processing Letters*. 1978. Vol. 7.

[8] Fournier A., Montuno D. Y. Triangulating simple polygons. *ACM Transactions on Graphics*. 1984. Vol. 3.

[9] O'Rourke J. Computational Geometry in C. 2nd ed. Cambridge University Press, 1998.

[10] Held M. FIST: Fast Industrial-Strength Triangulation of Polygons. *Algorithmica*. 2001. Vol. 30.

[11] Sutherland I. E., Hodgman G. W. Reentrant Polygon Clipping. *Communications of the ACM*. 1974. Vol. 17.

[12] Shewchuk J. R. Adaptive Precision Floating-Point Arithmetic. *Discrete & Computational Geometry*. 1997. Vol. 18.

[13] Kettner L. et al. Classroom examples of robustness problems. *Computational Geometry*. 2008. Vol. 40.

[14] Barber C. B., Dobkin D. P., Huhdanpaa H. The Quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*. 1996. Vol. 22, No. 4. pp. 469–483.

[15] de Berg M., Cheong O., van Kreveld M., Overmars M. Computational Geometry: Algorithms and Applications. Springer-Verlag, Berlin, 2008. 386 p.

[16] The CGAL Project. CGAL User and Reference Manual. 5.6 ed. CGAL Editorial Board, 2023. URL: https://doc.cgal.org/

[17] Blanchette J., Summerfield M. C++ GUI Programming with Qt 4. Prentice Hall, Upper Saddle River, 2006. 560 p.

[18] Phillips M. Object File Format (OFF). Geomview Documentation. 1996. URL: http://www.geomview.org/docs/html/OFF.html

[19] Chazelle B. Triangulating a simple polygon in linear time. *Discrete & Computational Geometry*. 1991. Vol. 6. pp. 485–524.

[20] Seidel R. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Computational Geometry*. 1991. Vol. 1, No. 1. pp. 51–64.

[21] Bern M., Eppstein D. Mesh generation and optimal triangulation. *Computing in Euclidean geometry*. World Scientific, 1992. pp. 23–90.

[22] Amenta N., Bern M., Kamvysselis M. A new Voronoi-based surface reconstruction algorithm. *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. 1998. pp. 415–421.