УДК 004.932, 004.04

О.І. Акимишин
Національний університет “Львівська політехніка”,
кафедра електронних обчислювальних машин

АЛГОРИТМ ПОТОКОВОГО ОПРАЦЮВАННЯ ТРІАНГУЛЯЦІЇ У ТРИВИМІРНОМУ ПРОСТОРІ

Розроблено алгоритм потокового опрацювання тріангуляційних сіток, що базується на розбитті сітки на незалежні елементи опрацювання.

The algorithm of triangular mesh streaming processing, based on mesh dividing into independent processing units, is developed.

Вступ. Розвиток обчислювальної техніки призвів до впровадження систем візуального подання інформації в різноманітні галузі людської діяльності, зокрема телебачення, космічна галузь, важка промисловість, медицина, індустрія розваг тощо. Крім того, дедалі частіше вимагається подання даних у вигляді тривимірних зображень. Сьогодні найпоширенішим способом представлення тривимірних зображень в комп’ютерній графіці є тріангуляційні сітки (ТС). Це зумовлено простотою їх математичного апарату та можливістю описувати тривимірні поверхні будь-якої складності із заданою точністю [1]. Проте реальні ТС, відтворені за даними пристроїв об’ємного сканування, містять десятки мільйонів точок та трикутників [2, 3], через що їхня візуалізація засобами універсальної обчислювальної техніки є неможливою. В останні роки стрімко розвиваються методи спрощення тріангуляції до заданої точності або заданої кількості елементів. Огляд відомих методів спрощення тріангуляції свідчить про їх виключно програмну реалізацію [4, 5]. Тому опрацювання сіток, що містять десятки мільйонів елементів, вимагає великих затрат часу при виконанні цієї задачі засобами універсальних ЕОМ. Для пришвидшення опрацювання ТС доцільно є розробляти апаратні засоби спрощення ТС.

Огляд літератури.Головною особливістю методів спрощення моделей згаданих розмірів є опрацювання даних поза ядром системи (out-of-core). Ідея полягає в тому, що в основній пам’яті комп’ютера по черзі розміщуються неперервні фрагменти сітки, які обробляють, тоді як значно більша частина даних знаходиться на диску. Методи цього класу можна розділити на такі групи:

1. **Вибірка областей сіток.** Розбиття загальної сітки на окремі фрагменти, які можуть опрацьовуватись в основній пам’яті, та їхнє послідовне опрацювання [2]. Для забезпечення об’єднання опрацьованих фрагментів між собою елементи тріангуляції на границі кожного із фрагментів повинні опрацьовуються за певним правилом, що загалом знижує ефективність методів цього класу.

2. **Опрацювання пакетами.** Послідовне опрацювання елементів тріангуляції у пам’яті за один або декілька проходів (ітерацій) [6]. Проміжні результати опрацювання зберігаються у файлах, які на наступних ітераціях є вхідними даними. Алгоритми цього класу забезпечують високу ефективність обчислень, проте часто додатково вимагають виконання етапу попереднього опрацювання даних, що є обчислювально містким процесом. Крім того, структура вхідних даних не забезпечує повної інформації про зв’язність сітки, що зумовлює втрату якості вихідної спрощеної тріангуляції.

3. **Опрацювання сітки без розбиття.** Доступу до даних досягають послідовністю запитів [7]. Для зменшення роботи з диском при кожному запиті попередньо виконується реорганізація вхідних даних. Деякі із методів цього класу використовують віртуальну пам’ять комп’ютера з метою опрацювання всієї сітки, проте вони вимагають спеціальних структур даних для ефективної роботи з віртуальним адресним простором. Перевагою методів цього класу є збереження повної зв’язності сітки та отримання вихідної тріангуляції високої якості, проте побудова додаткових

структур для роботи з віртуальною пам'яттю часто є складним та затратним процесом, що суттєво знижує ефективність обчислень.

4. **Опрацювання послідовностей.** Суть методу полягає у обмеженні доступу до вхідної тріангуляційної сітки та повний доступ до її фрагментів в момент їх опрацювання в основній пам'яті [8]. Вхідна послідовність формується у фіксованому порядку надходження вхідних даних, що, своєю чергою, вимагає етапу попереднього опрацювання даних. Цей тип опрацювання даних може поєднуватись із буферною обробкою. Тоді виконання обчислень над фрагментом сітки здійснюється у буфері, розмір якого встановлюється відповідно до характеристик комп'ютера, на якому виконується обробка.

5. **Паралельне опрацювання даних.** Суть методу полягає у розбитті вхідної послідовності на незалежні області та їхнє паралельне опрацювання за допомогою багатопроцесорної ЕОМ [9]. Результати попереднього опрацювання вхідної послідовності використовуються як вхідні дані на наступній ітерації алгоритму. Кількість ітерацій залежить від заданого користувачем рівня, до якого виконується спрощення тріангуляції. Цей метод пропонувався як підхід до опрацювання даних в основній пам'яті багатопроцесорної ЕОМ. Проте він може бути розвинутий для опрацювання сіток поза пам'яттю комп'ютера.

Аналіз продуктивності відомих методів свідчить, що час спрощення фрагментів тріангуляції значно перевищує час розбиття вхідної сітки. Зокрема в [10] час спрощення приблизно в п'ять разів більший, ніж час розбиття тріангуляції, та становить близько 80 % загального часу опрацювання даних. Тому доцільним є розроблення апаратних засобів опрацювання тріангуляції.

Під час спрощення тріангуляції або області ТС, видалення одного з її елементів зумовлює зміну геометрії в його околі. Це накладає обмеження на можливість ефективного опрацювання даних, зокрема реалізації конвеєрного та паралельного способів спрощення тріангуляції. На вхід пристрою не можна подавати наступний елемент тріангуляції, поки повністю не опрацьовано попередній та не перераховано ціни для елементів тріангуляції в околі видаленого елемента. Графічно будь-який алгоритм спрощення області ТС можна зобразити блок-схемою, наведеною на рис. 1. Для усунення згаданого недоліку пропонується розбити вхідну ТС на незалежні області опрацювання, а отже необхідно розробити метод розбиття вхідної тріангуляції.

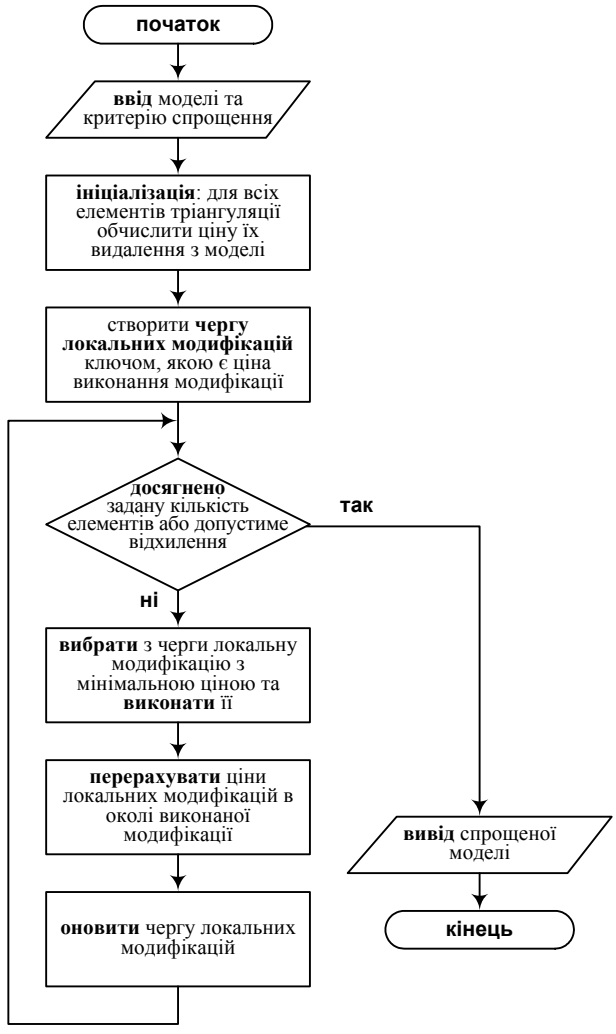

Рис. 1. Узагальнена блок-схема алгоритмів спрощення ТС

Мета досліджень. Ціллю даної роботи є розробка алгоритму потокового опрацювання ТС та реалізація на його основі спеціалізованих пристроїв опрацювання ТС. Вирішення описаної задачі доцільно розділити на такі етапи:

- Розроблення та реалізація алгоритму розбиття вхідної ТС на незалежні елементи опрацювання, тобто маючи тріангуляцію T отримати послідовність незалежних елементів опрацювання $E\{e_i\}$ таку, що e_i відповідає визначеній області вхідної тріангуляції T.
- Розробленняа алгоритму потокового опрацювання ТС та відповідних йому структур спеціалізованих пристроїв опрацювання ТС.

Розбиття ТС на незалежні елементи опрацювання. Залежно від типу локальної модифікації, елементом ТС, над яким вона виконується, є вершина, ребро або трикутник [11]. Саме ця особливість впливає на розмір областей ТС, на які треба розбивати вхідну тріангуляцію. У статті розглядаються алгоритми, локальними модифікаціями яких є колапс ребер, проте описаний підхід можна використати для будь-якого типу локальних модифікацій. Графічно взаємозв'язок між елементами тріангуляції зображено на рис. 2, *а.* Під час виконання локальної модифікації над ребром ТС змінюється геометрія ТС, внаслідок чого в околі видаленої вершини необхідно перерахувати ціни для позначених на рис. 2, *а* вершин. Відповідно до цього можна виділити незалежні області ТС, тобто області, де зміна геометрії одної із областей не впливає на іншу область (рис. 2, *б*)

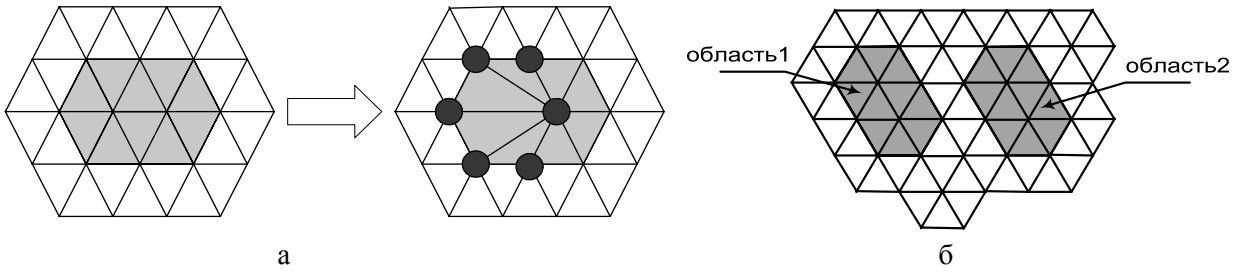

Рис. 2. Взаємозв'язок між елементами тріангуляції під час виконання локальних модифікацій

Алгоритм виділення незалежних елементів опрацювання можна описати так:

1. Початок.
2. Вичитати вхідну сітку або її фрагмент у пам'ять комп'ютера.
3. Позначити всі вершини вхідної ТС як невикористані.
4. Для кожного ребра ТС:
   – Перевірити вершини, які утворюють це ребро. Якщо вони позначені як невикористані, то перевірити вершини в їх околі.
   – Якщо всі вершини в околі ребра є позначені як невикористані, то утворити незалежний елемент опрацювання тріангуляції та позначити вказані вершини як використані.
   – Вершини, позначені як використані, пропускаються.
5. Вивести список незалежних елементів опрацювання.
6. Кінець.

Наступним кроком є розробка потокового алгоритму опрацювання тріангуляції, оскільки всі алгоритми спрощення ТС, згадані вище, є ітераційними та не дають змоги потоково обробляти дані. Крім того, якщо розроблений алгоритм подати у вигляді напрямленого потокового графу та розбити граф на незалежні за даними яруси, то на його основі можлива реалізація конвеєрного пристрою спрощення ТС, що теоретично дасть значний виграш у продуктивності порівняно із ітераційним опрацюванням даних.

Потокове опрацювання елементів ТС. Алгоритм потокового опрацювання тріангуляції наведено нижче.

Маючи на вході множину незалежних елементів опрацювання ТС, над кожним із них необхідно виконати таку послідовність операцій:

1. Вичитати елемент та розмістити його у вхідній пам'яті;

2. Обчислити відхилення, що виникне в результаті виконання локальної модифікації над елементом ТС;

3. Порівняти обчислене відхилення із величиною заданого допустимого відхилення;

4. Якщо обчислене відхилення менше/дорівнює величині заданого допустимого відхилення, виконати локальну модифікацію над поточним елементом ТС;

5. Вивести опрацьований елемент до вихідної множини елементів ТС;

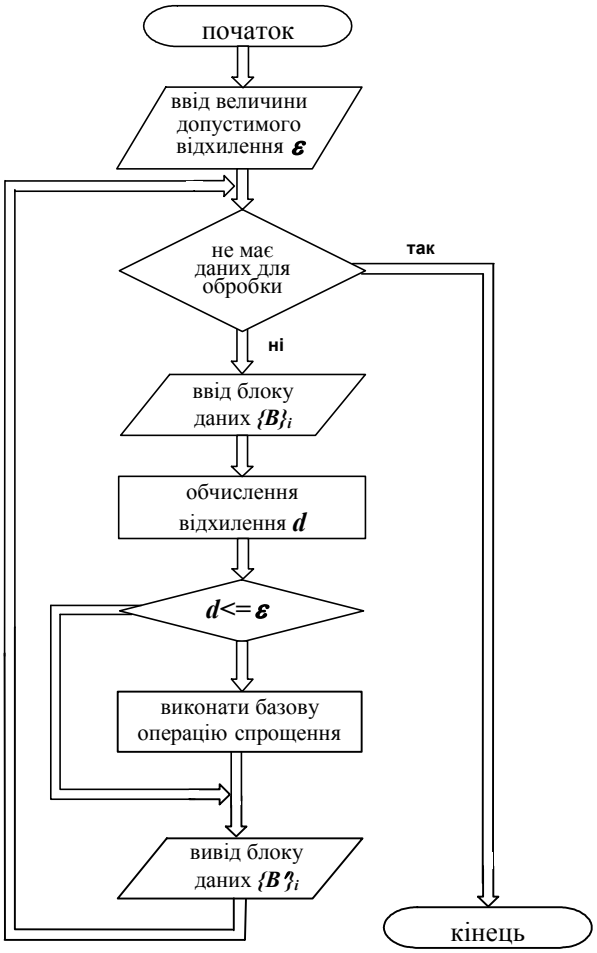

Рис. 3. Блок-схема алгоритму потокового спрощення ТС

Графічно алгоритм потокового опрацювання незалежних елементів ТС можна подати блок-схемою, зображеною на рис. 3. Обробка даних починається із введення величини заданого допустимого відхилення. Далі по черзі послідовно опрацьовуються всі елементи вхідної множини даних. Зворотний зв'язок, зображений на рис. 3, трактується тільки для позначення опрацювання послідовності даних, тобто при відображенні його у пристрій він буде відігравати роль ліній керування. У розробленому алгоритмі відсутня залежність за даними під час опрацювання будь-яких елементів вхідної послідовності.

На основі розробленого алгоритму можна визначити інтерфейс (рис. 4) та загальну структуру пристрою спрощення ТС (рис. 5).

Структуру пристрою доцільно розділити на такі основні вузли: вхідна буферна пам'ять, вузол обчислення відхилення, вузол комутації, вихідна буферна пам'ять та керівний автомат. Відносно функціональності, то вхідну та вихідну пам'ять використовують для узгодження внутрішніх та зовнішніх зв'язків пристрою; вузол обчислення відхилення виконує всі математичні операції над вхідними даними та забезпечує на виході результат порівняння величини заданого допустимого відхилення із обчисленим; вузол комутації забезпечує виконання локальної модифікації над елементом ТС; керівний автомат керує роботою всієї решти вузлів пристою.

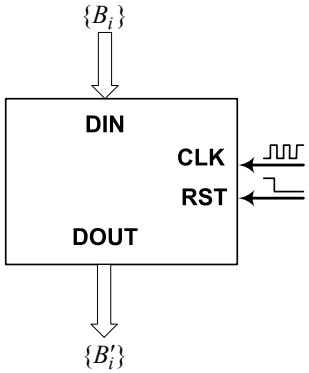

Рис. 4. Інтерфейс пристрою спрощення ТС

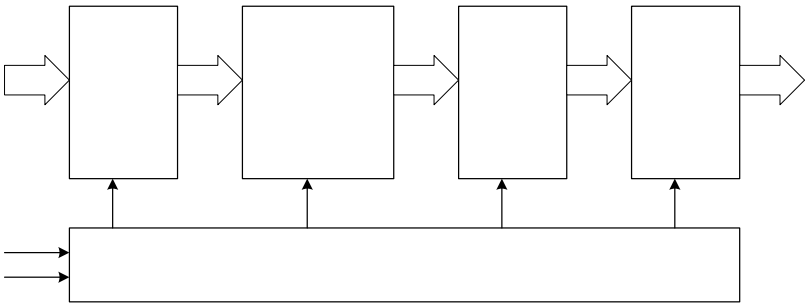

Рис. 5. Загальна структура пристрою потокового спрощення ТС

Отримані результати. Проведено моделювання роботи описаного алгоритму. Він реалізований у вигляді програми на C++ та протестований на деяких тривимірних зображеннях. На рис. 6 зображено результати виділення незалежних елементів опрацювання на тестовому зображенні.

Рис. 6. Розбиття вхідної тріангуляції на окремі області

Після розбиття вхідної ТС на незалежні елементи опрацювання описаним методом з'являється можливість потокового оброблення тріангуляції, оскільки після виконання локальної модифікації над елементом опрацювання решта елементів ТС залишаються незмінними.

На час написання статті розроблено модель пристрою потокового спрощення ТС, зокрема вхідну і вихідну пам'яті та вузол комутації. Моделі цих вузлів описані мовою опису апаратних засобів VHDL та виконане їх поведінкове моделювання. Розроблено графи алгоритмів виконання базових операцій для обчислення відхилення, що виникає внаслідок локальних модифікацій над елементами ТС. Проведено теоретичний розрахунок продуктивності опрацювання ТС, що свідчить про значний виграш у продуктивності порівняно із виконанням спрощення ТС на універсальних комп'ютерах.

Висновки: 1. Розроблено алгоритм розбиття ТС на незалежні елементи опрацювання, що дає змогу здійснити перехід від алгоритмів ітераційного спрощення ТС до потокового спрощення ТС.

2. Розроблено потоковий алгоритм спрощення ТС та на його основі запропоновано структуру пристрою спрощення ТС.

3. Виконано моделювання розробленого алгоритму розбиття ТС на незалежні елементи опрацювання та наведено результати його роботи на тестових зображеннях. Отриманий результат доводить правильність запропонованого способу розбиття ТС.

1. Luebke D. A Developer's Survey of Polygonal Simplification Algorithms // IEEE Computer Graphics & Applications. – 2001. 2. Bernardini F., Martin I., Mittleman J., Rushmeier H., Taubin, G. Building a digital model of Michelangelo's Florentine Pieta // IEEE Computer Graphics and Applications 22. – 2002. – P. 59–67. 3. Мельник А.О., Акимишин О.І. Проріджування тріангуляційних сіток тривимірних об'єктів комп'ютерної томографії // Вісн. Нац. ун-ту "Львівська політехніка". – 2006. – № 573. – С. 131–137. 4. Leng J . The IEEE Viz 2001 Conference. Trip Report. San Diego. – California, USA, 2001. 5. Jarlier S., Kim HyungSeok, Garchery S., Magnenat-Thalmann N. Reduction: State-of-the-Art // MIRALab, University of Geneva, May 2005. 6. Lindstrom P., Silva C. A memory insensitive technique for large model simplification. In Visualization'01 Proceedings. – 2001. – P. 121–126. 7. Cignoni P., Montani C., Rocchini C., Scopigno R. External memory management and simplification of huge meshes // IEEE Transactions on Visualization and Computer Graphics. – 2003. 8. Isenburg M., Lindstrom P., Gumhold S., Snoeyinklarge J. Mesh Simplification Using Processing Sequences // Proceedings of IEEE Visualization 2003, Seattle, Washington, October 19–24, 2003. 9. Franc M. , Skala V. Parallel Triangular Mesh Reduction. Proceedings of ALGORITMY 2000 // Conference on Scientific Computing. – P. 357–367. 10. Borodin P., Guthe M., Klein R. Out-of-Core Simplification with Guaranteed Error Tolerance. VMV 2003. – Munich, Germany, 2003. 11. Скворцов А.В. Триангуляция Делоне и её применение. – Томск: Изд-во Том. ун-та, 2002. – 128 с.