УДК 004.925.2

К.т.н., доцент Соколова Н.А., магістрант Барабанов М.Ю.

Національний технічний університету України
«Київський політехнічний інститут»

ВИКОРИСТАННЯ ПОПЕРЕДНЬОГО ВІДСІВУ ЯВНО НЕВИДИМИХ ОБ'ЄКТІВ В АЛГОРИТМАХ ВИДАЛЕННЯ НЕВИДИМИХ ПОВЕРХОНЬ

Вступ

Традиційні алгоритми видалення невидимих поверхонь зосереджуються на генерації зображень з точною взаємною видимістю між об'єктами. Вони неефективні для сцен з великою кількістю об'єктів тому, що необхідно опрацювати абсолютно всі многокутники. Алгоритми відсіву намагаються швидко виявити і позбавитися від великих частин сцени, що приховані від спостерігача. Алгоритми відсіву не замінюють традиційні алгоритми видалення невидимих поверхонь тому, що вони не розв'язують задачу видимості на детальному рівні (наприклад, на рівні пікселя для растрової графіки). Ці два види алгоритмів повинні працювати разом: спочатку алгоритм відсіву швидко позбавляється від великих груп невидимих многокутників, а потім традиційний алгоритм видалення невидимих поверхонь обробляє многокутники, що залишилися, і генерує фінальне зображення.

Постійно зростаючі потужності персональних комп'ютерів, а також нові версії Adobe Flash Player дають можливість візуалізації тривимірних об'єктів у веб-застосуваннях. Але ряд обмежень Flash платформи не дає можливості керувати всіма етапами візуалізації. Тому єдиний алгоритм видалення невидимих поверхонь, який більш-менш ефективно реалізовано, ґрунтується на використанні списку пріоритетів.

Постановка задачі

При візуалізації сцен з великою кількістю об'єктів досить часто буває, що більшу їх частину не видно. Вони повністю закриваються об'єктами, що

знаходяться на передньому плані. Але алгоритм, який використовує список пріоритетів, повинен зобразити їх усіх. А це призводить до того, що одна й та сама область зображення перемальовується по кілька разів.

Тому необхідно розробити спосіб попереднього відсіву явно невидимих об'єктів, який би задовольняв наступним вимогам:

1. Не повинен відкидати об'єкти, які є видимими у фінальному зображенні.
2. Попередній відсів не повинен займати більше часу, ніж візуалізація всіх об'єктів.

Аналіз існуючого методу

Алгоритми, які використовують список пріоритетів ґрунтується на використанні попереднього сортування за глибиною або пріоритетом [1]. Мета такого сортування – отримати список елементів сцени, впорядкований за пріоритетом глибини, заснованому на відстані від точки до спостерігача. Коли такий список остаточний, то ніякі два елемента не будуть взаємно перекривати один одного. Тоді можна записати всі елементи в буфер зображення по черзі, починаючи з елемента найбільш віддаленого від спостерігача. Більш близькі елементи будуть затирати інформацію про більш віддалені елементи. В сценах з великою кількістю об'єктів віддалені елементи можуть повністю затиратися декілька (іноді – десятки) разів.

Якби була можливість на попередньому етапі оцінити ймовірність повного затирання елемента, з метою виключення такого елемента, то це б зменшило кількість зайвих обчислень і збільшило б швидкість та ефективність алгоритму.

Таким чином, виникла необхідність розробки способу попереднього відсіву явно невидимих об'єктів.

Опис запропонованого методу

Нехай задано сцену з деякою множиною об'єктів. Наша мета – це швидке знаходження підмножини об'єктів X, які перекриваються деяким вибраним об'єктом y. Ідея алгоритму полягає в тому, щоб під час визначення області сцени, яку закриває об'єкт y, використовувати інший – більш простий об'єкт z або множину простих об'єктів Z. В ролі таких об'єктів можуть виступати випуклі многокутники та/або многогранники. Будемо називати множину Z внутрішньою оболонкою. Якщо їх розмістити всередині об'єкта y, то можна стверджувати, що область сцени, яку вони собою закривають, є

підобластю сцени, яку закриває *y*. Таким чином, знайшовши множину об'єктів, що закривається внутрішньою оболонкою, ми отримаємо деяку підмножину *X*.

Перевірка на перекриття проводиться в просторі зображення шляхом порівняння проекцій. Для того, щоб зберігати інформацію про те які фрагменти зображення вже були перекриті, поділимо зображення на рівні прямокутні сегменти. Коли проекція внутрішньої оболонки буде повністю покривати такий сегмент, будемо відмічати його як перекритий. Потім множина перекритих сегментів порівнюватиметься з проекцією прямокутної оболонки для визначення видимості об'єкта. Будемо називати множину таких сегментів буфером видимості. Оскільки в якості допоміжних об'єктів ми використовуємо випуклі многокутники та многогранники, то їх проекції також будуть випуклими. Таким чином, задача перекриття зводиться до задачі взаємного розташування прямокутника і випуклого многокутника (можна визначити, наприклад, за методом обчислення кутів [1]).

Пропонується наступний алгоритм:

1. Складається список об'єктів, для яких потрібно оцінити видимість.
2. Виконуються геометричні перетворення лише над внутрішніми та прямокутними оболонками.
3. Об'єкти сортуються за відстанню до спостерігача.
4. Якщо у списку є два об'єкти, прямокутні оболонки яких перетинаються, то слід перевірити чи перетинає внутрішня оболонка ближнього об'єкта прямокутну оболонку більш віддаленого. Якщо перетинаються, то слід видалити зі списку один із об'єктів.
5. Зі списку вибирається наступний найближчий до спостерігача об'єкт. Якщо дійшли до кінця списку, то переходимо до п.9.
6. Знаходимо проекцію його прямокутної оболонки, і на основі буфера видимості визначаємо чи закривається він деяким попереднім об'єктом. Якщо так, то видаляємо його зі списку і переходимо до п. 5.
7. Вибираємо наступний об'єкт з внутрішньої оболонки і знаходимо його проекцію. Якщо це був останній, то переходимо до п.5.
8. Якщо проекція перекриває якісь сегменти буфера видимості, то позначаємо відповідні сегменти як перекриті. Переходимо до п.7.
9. Візуалізація за алгоритмом, що використовує список пріоритетів.

Слід зазначити, що на 4-му етапі, коли треба видалити один із двох об'єктів доцільніше залишати той, що потенційно може закрити собою більше простору. Для цього можна порівнювати розміри прямокутних оболонок, внутрішніх оболонок або їх проекцій.

Замість звичайного буфера видимості, можна використати багаторівневий. Коли, наприклад, на найнижчому (більш детальному) рівні

заповнюються чотири сусідні сегменти, то з цього рівня вони зникають, а замість того з'являється один сегмент на наступному рівні, що покриває чотири зниклі. Тоді перевірка на покриття об'єкта має починатися з найвищого рівня і спускатись до більш детального.

Оскільки використання методу не впливає на якість зображення, то ефективність можна перевірити порівнюючи швидкість візуалізації з ним і без нього. Також можна оцінити міру консервативності методу [2] – відношення множини потенційно видимих об'єктів, яку залишає алгоритм після опрацювання, до точної множини видимих об'єктів.

Висновки

Не існує одночасно ефективного і універсального алгоритму відсіву явно невидимих об'єктів. Розроблений метод також має свої переваги і недоліки. Серед переваг слід відзначити:

1. Можливість використання як для архітектурних, так і для загальних візуалізацій.
2. Можливість роботи як зі статичними, так і з динамічними об'єктами.
3. Відсів проводиться перед геометричними перетвореннями об'єктів, тому не витрачається час на перетворення невидимих об'єктів.

Головний недолік методу – те, що на даний момент не відомий спосіб автоматичної побудови внутрішньої оболонки. Це може бути напрямком для подальших досліджень.

Завдяки розробленому методу, з'являється можливість візуалізації набагато складніших тривимірних моделей в реальному часі. Це може розширити використання інтерактивної тривимірної графіки у веб-застосуваннях при створенні дизайну та інтерфейсу, візуалізації інформаційних даних або ілюстративного матеріалу.

Література

1. *Д. Роджерс.* Алгоритмические основы машинной графики. – М.:«Мир», 1989. – 230 с.
2. *S. Gummerus.* Conservative From-Point Visibility // University of Tampere – 2003. – P. 10.
3. *H. Zhang.* Effective Occlusion Culling for the Interactive Display of Arbitrary Models // The University of North Carolina. – 1998. – P. 11.
4. *M. Abrash.* Graphics Programming Black Book // An International Thomson Publishing Company – 1997. – P. 1095.