Тема 3 Видалення невидимих ліній та поверхонь: методи плаваючого горизонту, трасування променів

Видалення невидимих граней

Ця задача складніша, ніж задача видалення невидимих ліній. Якщо практично всі методи, що служать для видалення невидимих ліній, працюють в об'єктному просторі і дають точний результат, то для видалення поверхонь існує велике число методів, що працюють на екрані, а також змішаних методів.

Метод трасування проміння

Найпоширенішим методом для визначення видимості граней є метод трасування проміння, при якому для кожного пікселя картинної площини визначається найближча до нього грань. Для цього через даний піксель випускається промінь, знаходяться всі точки перетину з гранями, і серед них вибирається найближча. Спостерігач бачить будь-який об'єкт який випускається якимсь джерелом світла і падає на цей об'єкт та потім якимсь шляхом доходить до спостерігача, відобразившись від поверхні, заломившись або пройшовши через неї. Якщо прослідити за променням світла, випущеним джерелом, то можна переконатися в тому, що дуже небагато з них дійдуть до спостерігача. Отже, цей процес був би неефективний. Аппель першим запропонував відстежувати (трасувати) проміння у зворотному напрямі, тобто від спостерігача до об'єкту рис. 10

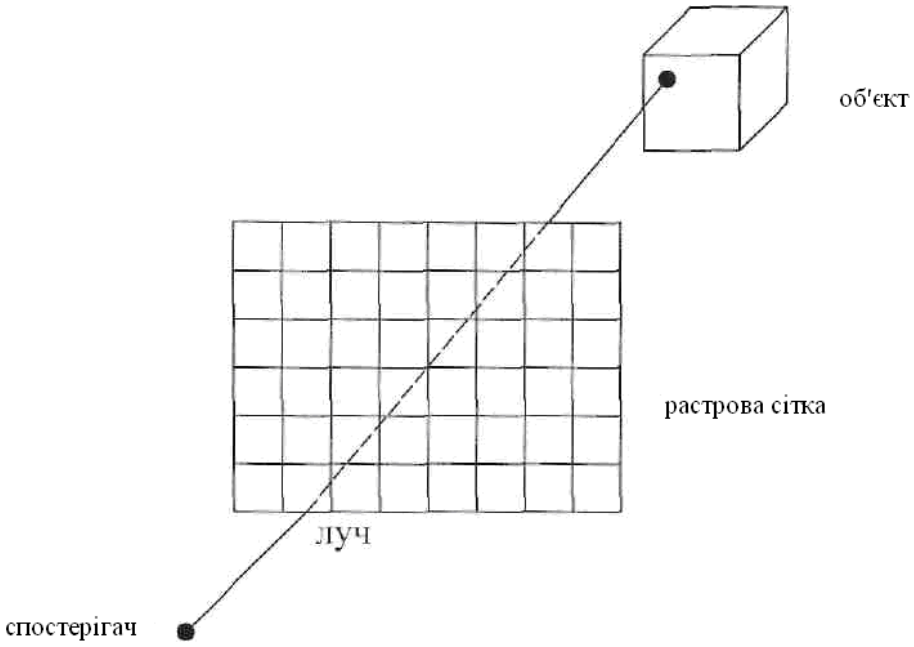


Рис. 10 Просте трасування променя

В цьому алгоритмі передбачається, що сцена вже перетворена в простір зображення. Перспективне перетворення не використовується, вважається, що точка зору або спостерігач знаходиться в нескінченності на позитивній півосі Z, тому все світлове проміння паралелі осі Z. Кожний промінь, виходящий від спостерігача, проходить через центр пікселя на растрі до сцени. Необхідно перевірити перетин кожного об'єкту сцени з кожним променем. Якщо промінь перетинає об'єкт, то визначаються всі можливі точки перетину променя і об'єкту. Ці перетини упорядковуються по глибині. Перетин з максимальним значенням Z представляють

видиму поверхню для даного пікселя. Атрибути цього об'єкту використовуються для визначення характеристик пікселя.

Якщо точка зору знаходиться не в нескінченності, алгоритм трасування лише трохи ускладнюється. Тут передбачається, що спостерігач як і раніше знаходиться на позитивній півосі Z. Картинна площина, тобто растр, перпендикулярна осі Z (як показано на рис. 11). Задача полягає в тому, щоб побудувати одноточкову центральну проекцію на картинну площину.

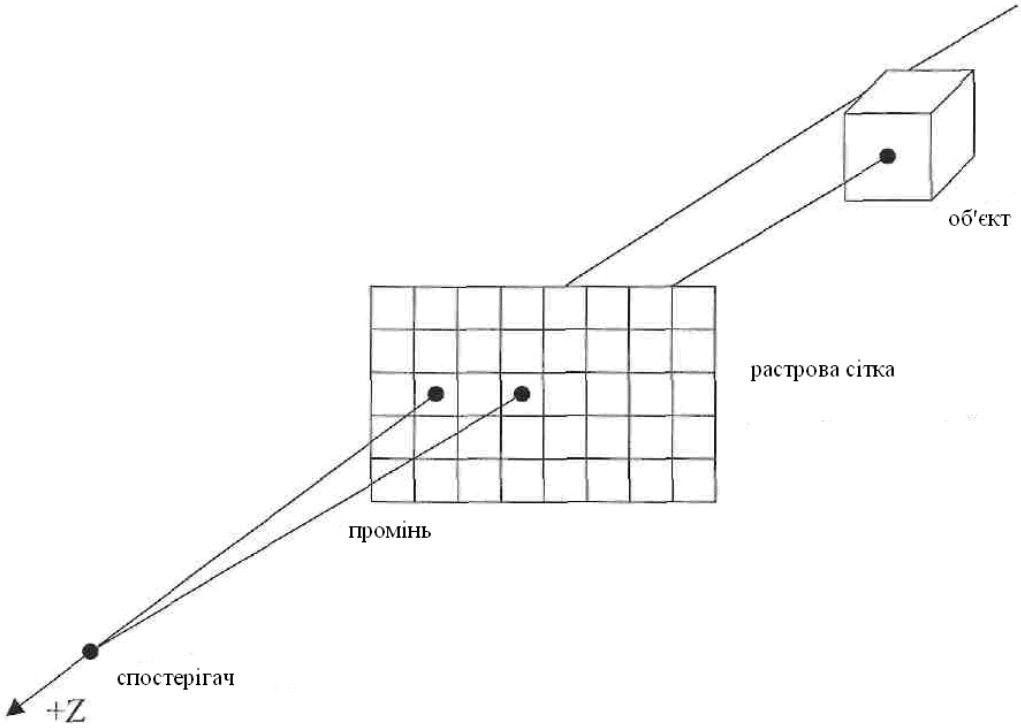


Рис. 11 Трасування променя з урахуванням перспективи.

Найважливішим елементом алгоритму визначення видимих поверхонь шляхом трасування проміння, є процедура визначення перетинів. Об'єкти сцени можуть складатися з набору плоских багатокутників. Щоб позбудеться від непотрібного пошуку перетинів, проводитися перевірка перетинів променя з об'ємною оболонкою даного об'єкту. І якщо промінь не перетинає оболонки, то не потрібно більше шукати перетинів цього об'єкту з променем. В якості оболонки можна використовувати прямокутний паралелепіпед або сферу.

Метод Z-буфера

Одним з найпростіших алгоритмів видалення невидимих граней і поверхонь є метод Z-буфера (буфера глибини), де для кожного пікселя, як і в методі трасування проміння, знаходиться грань, найближча до нього уздовж напряму проектування, проте тут цикли по пікселях і по об'єктах міняються місцями.

Поставимо у відповідність кожному пікселю (x,y) екрану, окрім кольору C(x,y) що зберігається у відеопам'яті, його відстань до екрану уздовж напряму проектування Z(x,y)(його глибину).

Масив глибин ініціалізувався $+\infty$.

Для виводу на екран довільної грані вона переводиться в растрове уявлення на екрані, а потім для кожного пікселя цієї грані знаходиться його глибина. У випадку якщо ця глибина

менше значення глибини, що зберігається в Z-буфере, піксель малюється, і глибина заноситься в Z-буфер.

Дуже ефектним є поєднання растрової розгортки грані з виводом Z-буфер. При цьому, для обчислення глибини пікселя можуть застосовуватися інкрементальні методи, що вимагають всього декілька складань на піксель Грані малюються послідовно, рядок за рядком для знаходження необхідних значень використовується лінійна інтерполяція.

Фактично метод Z-буфера здійснює порозрядне сортування по x і y, а потім сортування по Z, вимагаючи всього одного порівняння для кожного пікселя кожної грані. Даний метод працює виключно в просторі екрану, і не вимагає ніякої попередньої обробки даних. Порядок, в якому грані виводяться на екран, не грає ніякої ролі. Для економії пам'яті можна змальовувати не все зображення відразу, а малювати по частинах. Для цього екран розбивається на частини (звичайно це горизонтальні смуги) і кожна така частина обробляється незалежно. Розмір пам'яті під буфер визначається найбільшою з цих частин. Недоліком методу є не тільки великий об'єм пам'яті під буфер, але також надмірність обчислень: здійснюється вивод всіх граней незалежно від того, видимі вони чи ні. І якщо даний піксель накривається десятьма різними гранями лицьовими, то для кожного відповідного пікселя кожної з цих десяти граней необхідно провести розрахунок кольору.

Існує декілька модифікацій методу Z-буфера, що дозволяють помітно скоротити к-ть граней, що виводяться. (Ієрархічний метод)

Існує ще цілий ряд методів, що здійснюють видалення граней: метод двійкового розбиття простору, метод відрядкового сканування, алгоритм Варнока, алгоритм художника, і т.д.

Видалення невидимих ліній і площин

Аналіз видимості об'єктів можна проводити як в тривимірному просторі, так і в двомірному просторі. Це проводиться з розділенням методів на два класи:

- методи, що працюють безпосередньо в просторі самих об'єктів;
- методи, що працюють в просторі двох, тобто працюючі з проекціями об'єктів.

Одержуваний результат є або набором видимих областей або відрізків, заданих з машинною точністю (має безперервний вид), або інформацією про найближчий об'єкт для кожного пікселя екрану (має дискретний вид). Методи першого класу дають точні рішення задачі видалення невидимих ліній і поверхонь. Вони можуть працювати як з самими об'єктами, виділяючи ті їх частини, які видимі, так і з їх проекціями на площину, виділяючи на ній області, відповідні проекціям видимих частин об'єктів. Оскільки ці методи працюють з безперервними початковими даними, то їх називають безперервними. Найпростіший варіант безперервного підходу полягає в порівнянні кожного об'єкту зі всіма іншими, що дає тимчасові витрати пропорційні n^2, де n - кількість об'єктів на сцені.

Методи другого класу дають наближене рішення задачі видимості, визначаючи видимість тільки в деякому наборі точок площини. Вони дуже сильно прив'язані до растрових властивостей площини, і полягають у визначенні для кожного пікселя тієї грані, яка є найближчою до нього уздовж напряму проектування.

Зміна розширення приводить до необхідності повного перерахунку всього зображення. Найпростіший варіант дискретного методу має тимчасові витрати порядку Cn, де C- загальна кількість пікселов екрану, n - кількість об'єктів. (В цьому методі бувають помилки дискретизації, але він дуже простий).

Існує ще цілий ряд змішаних методів, що використовують роботу, як в об'єктному просторі, так і на площині.

Більшість алгоритмів видалення невидимих граней і поверхонь тісно пов'язана з різними методами сортування. Деякі методи проводять сортування явно, а в деяких вона присутня в прихованому вигляді.

Дуже поширеною структурою даних в задачах видалення невидимих ліній і поверхонь є різні типи дерев - двійкові, четвертні, вісімкові. Методи є комбінаціями ряду найпростіших алгоритмів. Дуже важлива роль в підвищенні ефективності методів видалення невидимих ліній і граней відводиться когерентності (зв'язаність). Виділяють декілька типів когерентності.

- когерентність в площині - якщо даний цикл відповідає точці грані P, то швидше за все сусідні пікселі теж відповідають точкам тієї ж грані (рис. 2)

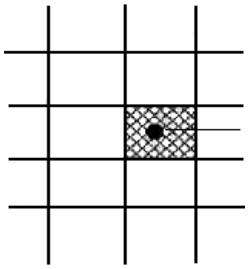


Рис 2

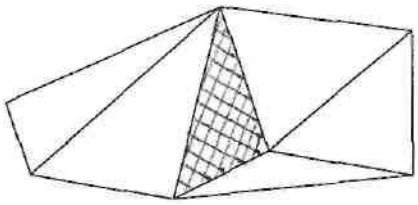


Рис 3

- когерентність в просторі об'єктів - якщо даний об'єкт (грань) бачимо (не бачимо) то розташований поряд об'єкт (грань) швидше за все також є видимим (невидимим) (рис. 3)

- у разі побудови анімації виникає третя когерентність – тимчасова:

грані, видимі в даному кадрі, швидше за все будуть видимими і в наступному; аналогічно грані, невидимі в даному кадрі, швидше за все будуть невидимими і в наступному.

Використовування когерентності дозволяє помітно скоротити кількість виникаючих перевірок, і помітно підвищити швидкодію алгоритму.

Існує декілька алгоритмів, які ілюструють видалення невидимих ліній і поверхонь. Розглянемо один з них.

Алгоритм плаваючого горизонту

Алгоритм плаваючого горизонту частіше за все використовується для видалення невидимих ліній тривимірного представлення функцій, що описують поверхню у вигляді $F(x, y, z)= 0$.

Головна ідея даного методу полягає в зведенні тривимірної задачі до двомірної, шляхом перетину початкової поверхні послідовністю паралельних січних площин, що мають постійні значення координат x, y, або z.

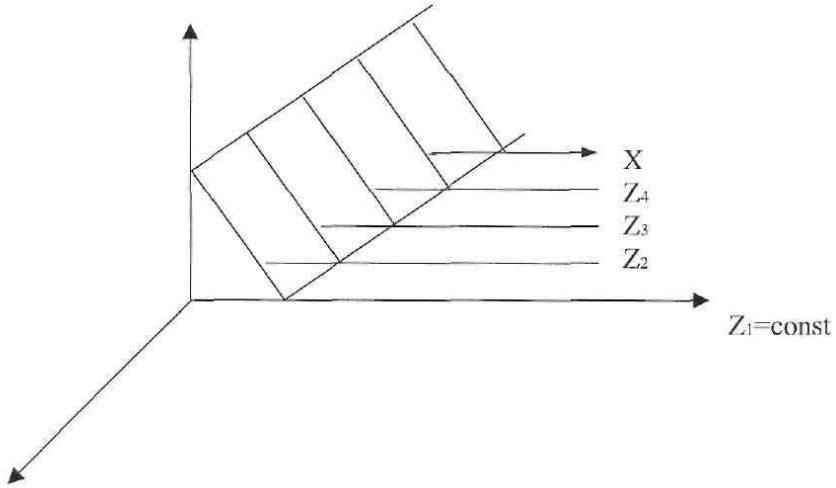


Рис. 4 Січні площини з постійною координатою

На цьому малюнку наведений приклад, де вказані паралельні площини визначаються постійним значенням Z.

Функція F(x, y, z)=0 зводиться до послідовності кривих, що лежать в кожній з цих паралельних площин, наприклад в площині y=f(x,z) *або* x = f(y, z), де z - постійно на кожній із заданих паралельних площин.

Тепер поверхня складається з послідовних кривих, що лежать в кожній з цих площин, як показано на рис.5

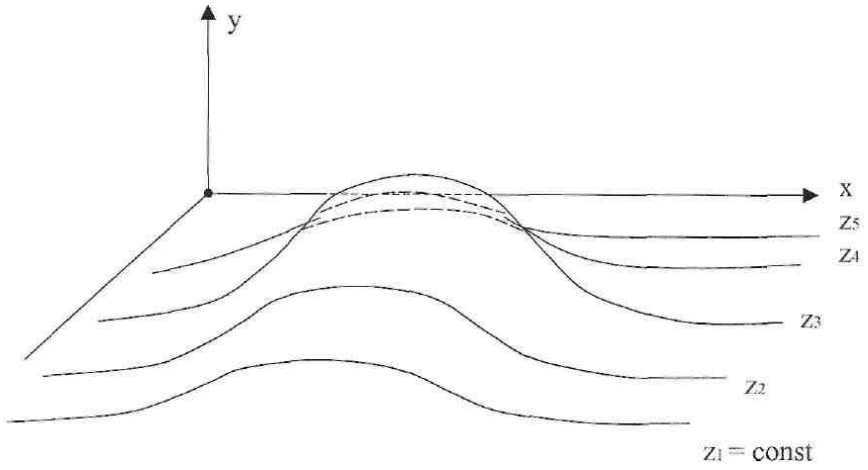


Рис. 5 Криві в січних площинах з постійною координатою.

Тут передбачається, що отримані криві є однозначними функціями незалежних змінних. Якщо спроектувати отримані криві на площину Z=0, як показано на малюнку (6) то відразу стає ясна ідея алгоритму видалення невидимих ділянок початкової поверхні. Алгоритм спочатку упорядковує площини Z=const за збільшенням відстані до них від точки спостереження. Потім, для кожної площини, починаючи з найближчою до точки спостереження, будується крива, що лежить на ній, тобто для кожного значення координати x в просторі зображення визначається відповідне значення.

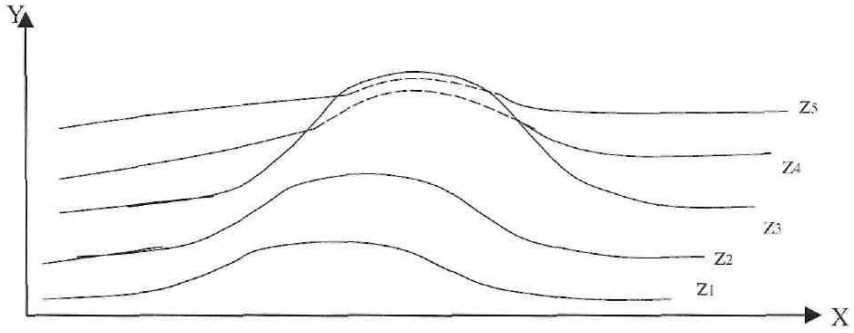
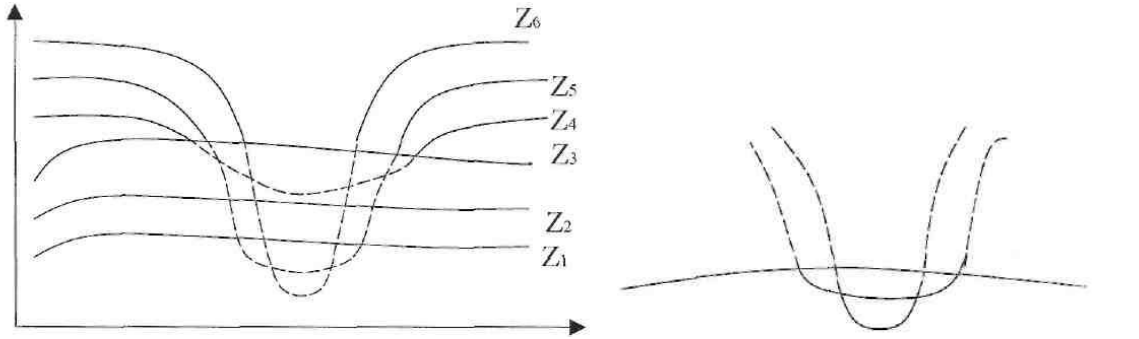


Рис.6 Проекція кривих на площині Z=0

Алгоритм видалення невидимих ліній полягає в наступним: якщо на поточній площині, при деякому заданому значенні X відповідне значення Y на цій кривій більше значення Y для всіх попередніх кривих при цьому значенні X, то поточна крива видима в цій точці; в іншому випадку вона невидима. Невидимі ділянки показані пунктиром, на рис. Реалізація даного алгоритму дуже проста. Для зберігання максимальних значень Y при кожному значенні X використовується масив, довжина якого рівна числу різних точок (дозволу) по осі X в просторі зображення. Значення, що зберігаються в цьому масиві, є поточним значенням «горизонту». Тому, у міру малювання кожній черговий кривий цей горизонт «спливає».

Фактично цей алгоритм видалення невидимих ліній працює кожного разу з однією лінією. Він дуже добрий до тих пір, поки яка-небудь чергова крива не виявиться нижче найпершої з кривих, як показано на рис. 7(а)



а) б)

Рис. 7 обробка нижньої сторони поверхні

Подібні криві видимі і представляють собою нижню сторону початкової поверхні, але, алгоритм вважатиме їх невидимими. Нижня сторона поверхні робиться видимою, якщо модифікувати цей алгоритм, включивши в нього нижній горизонт, який опускається вниз по ходу роботи алгоритму. Це реалізується за допомогою другого масиву, довжина якого рівна числу різних точок по осі X в просторі зображення. Цей масив містить якнайменше значення Y для кожного значення X. Алгоритм тепер буде таким: якщо на поточній площині при деякому заданому значенні X відповідне значення Y на кривій більше максимуму або менше мінімуму по Y для всіх попередніх кривих при цьому X, то поточна крива видима. В іншому випадку вона невидима (рис. 7б). У викладеному алгоритмі передбачається, що значення функції, тобто Y відомо для кожного значення X в просторі зображення.

Але якщо для кожного значення X не можна вказати відповідні йому значення Y, то неможливо підтримувати масиви верхнього і нижнього плаваючих горизонтів. У такому разі

використовуються лінійна інтерполяція значень Y між відомими значеннями для того, щоб заповнити масиви верхнього і нижнього плаваючих горизонтів(рис.8)

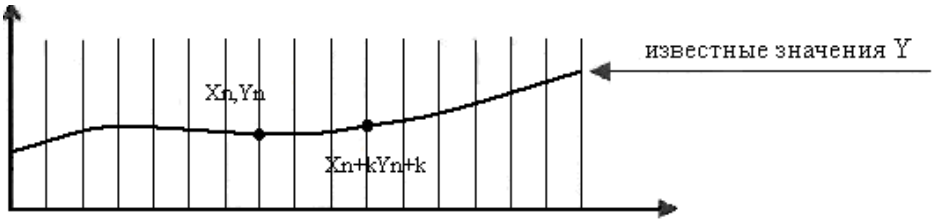
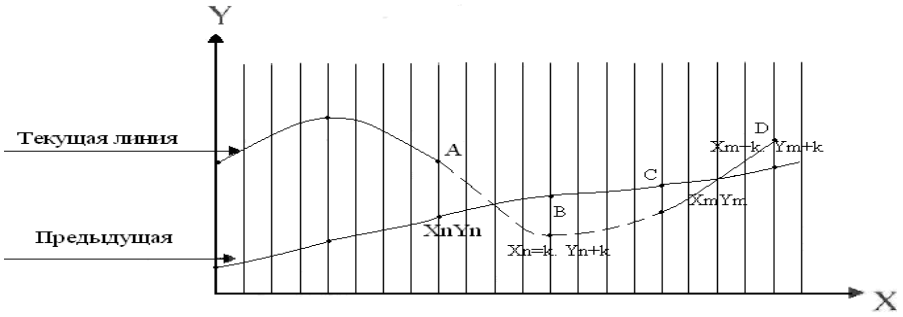
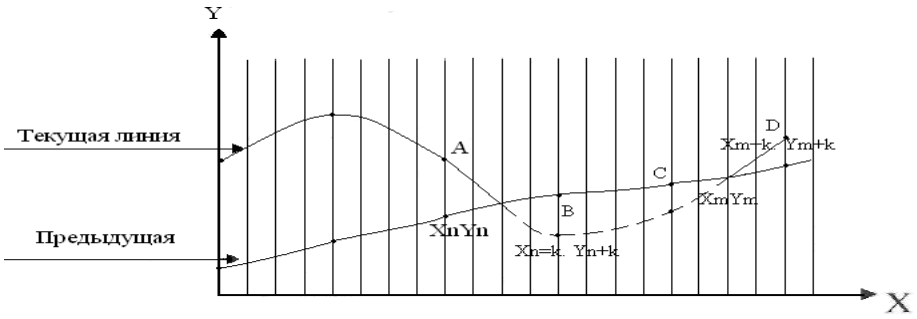


Рис. 8 Лінійна інтерполяція між заданими точками

Якщо видимість кривої міняється, то метод з такою простою інтерполяцією не дає коректного результату. Цей ефект продемонстрований на наступному малюнку. 9



9 (а)



9 (б)

Рис. 9 Ефект пересічних кривих

Припускаючи, що операція по заповненню масивів проводиться після перевірки видимості, отримаємо, що під час переходу поточної кривої від видимого до невидимого стану (сегмент AB, рис 9а) точка (Xn+k, Yn+k) оголошується невидимою. Тоді ділянка кривої між точками (Xn, Yn) і (Xn + до Yn + до) не зображається і операція по заповненню масивів не проводиться. Утворюється зазор між поточною і попередньою кривими. Якщо на ділянці поточної кривої відбувається перехід від невидимого стану до видимого (сегмент CD) рис. 9а, то точка (Xт + до, Ym+ k) оголошується видимою, а ділянка кривої між точками (Xm,Ym) і (Xm+k, Ym+k) зображається і операція по запам'ятовуванню масивів проводиться. Тому зображається і невидимий шматок сегменту CD. Крім того, масиви плаваючих горизонтів не міститимуть точних значень Y. Це може спричинити за собою додаткові небажані ефекти для подальших кривих, тому необхідно вирішувати задачу про пошук точок перетину сегментів поточної і передуючої кривих. Існує декілька методів отримання точок перетину кривих. На растрових дисплеях значення координати X можна збільшити на 1, починаючи з Xn або Xm. Значення Y, відповідні поточному значенню координати X в просторі зображення, виходить шляхом додавання до значення Y, відповідному попередньому значенню координати X, вертикального приросту ΔY вздовж заданої кривої.

Потім, визначається видимість нової точки, з координатами (X+1,Y+ΔY). Якщо ця точка видима, то активується пов'язаний з нею піксель. Якщо невидима, то піксель не активується, а X збільшується на 1. Цей процес продовжується до тих пір, поки не зустрінеться Xn+k, або Xm+k Перетин для растрових дисплеїв визначається викладеним методом з достатньою точністю. Близький і навіть більш елегантний метод визначення перетину започатковано на двійковому пошуку.

Точне значення точки перетину двох прямолінійних відрізків, які інтерполюють поточну і попередні криві, між точками (Xn, Yn) і (Xn+k,Yn+k) задається формулами:

$$X = Xn - \frac{\Delta X\,(Ynp - Ync)}{(\Delta Yp - \Delta Yc)};$$

$$Y = m(X - Xn) + Yn;$$

де $\Delta X = Xn + k - Xn;$

$$\Delta Y\rho = (\mathrm{Yn} + \mathrm{k})\rho - (\mathrm{Yn})\rho;$$

$$\Delta Yc = (Yn + k) - (Yn)c;$$

$$m = [(Yn + k) - (Yn)]/\Delta X,$$

а індекси c і ρ відповідної поточній і передуючій кривим. Отриманий результат показаний на рис. 9(б). Тепер алгоритм висловлюватиметься так:

- якщо на поточній площині при деякому заданому значенні X відповідне значення Y на кривій більше максимуму, або менше мінімуму по Y для всіх попередніх кривих при цьому X, то поточна крива видима, а в противному випадку вона невидима.

- якщо на ділянці від попередньої (Xn) до поточного (Xn+k) значення X видимість кривої змінюється, то обчислюється точка перетину (Xi).

- якщо на ділянці від Xn до Xn+k сегмент кривої повністю бачимо, то він зображається цілком; якщо він став невидимим, то зображається фрагмент від Xn до Xi, якщо ж він став видимим, то зображається фрагмент від Xi до Xn+k.

Суть: на екрані малюються тільки ті частини лінії, які знаходяться вище за лінію Ymax і ті, які знаходяться нижче Ymin.

В комп'ютерній графіці існує цілий ряд алгоритмів, що реалізовують видалення невидимих ліній: алгоритм Роберта, алгоритм Варнока і алгоритм Аппеля.