Тема 2 Відсікання: розгляд методів відсікання для двомірного та тривимірного відсікання Кіраса-Бека, Коєна-Сазерленда

Відсікання — це процес виділення деякої частини бази даних. Відсікання застосовується в алгоритмах видалення невидимих ліній і поверхонь, при побудові тіней, а також при формуванні фактури. Алгоритми відсікання бувають дво- або тривимірними, і застосовуються як до регулярних (стандартним) областей і об'ємів так і до не регулярних областей і об'ємів. (До стандартних відносяться прямокутники і паралелепіпеди із сторонами, паралельними осям координат). Ці алгоритми можна реалізувати апаратно, або програмно.

Двомірні відсікання

На рис. 13.1 показана плоска сцена і відсікаюче вікно регулярної форми. Вікно задається лівим (Л), правим (П), нижнім (Н), верхнім (В) двомірними ребрами. Регулярним відсікаючим вікном є прямокутник, сторони якого паралелі осям координат об'єктного простору або осям координат екрану.

Метою алгоритму відсікання є визначення тих точок, відрізків або їх частин, які лежать всередині відсікаючого вікна ці точки, відрізки або їх частини залишаються для візуалізації, а всі інші відкидаються.

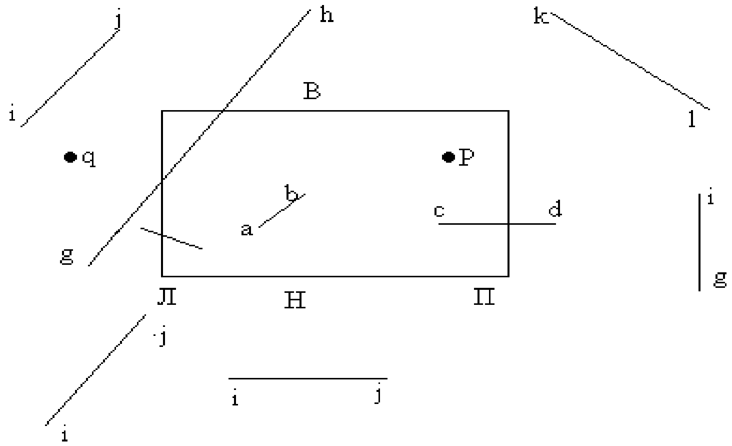

Рис. 13.1 Двомірне відсікаюче вікно.

У багатьох випадках переважне число точок або відрізків лежить цілком всередині або зовні відсікаючого вікна. Тому важливо відбирати відрізки подібні ab, або точки подібні P, і відкидати відрізки подібні ij, або точки, подібні q (рис 13.1). Точки лежачі всередині відсікаючого вікна, задовольняють умову:

$$\begin{cases} X_л \le X \le X_п \\ Y_н \le Y \le Y_в \end{cases}$$

Знак рівності говорить про те, що точки, що лежать на межі вікна, вважаються тими, що знаходяться усередині нього. Відрізок лежить усередині вікна (наприклад, ab).

Але, якщо обидва кінці відрізка лежать усередині вікна, то відрізок не обов'язково лежить цілком зовні вікна(наприклад gh). Якщо ж обидва кінці відрізка лежать справа, зліва, вище, або нижче за вікно, то цей відрізок цілком лежить зовні вікна, а значить він невидимий.

Перевірка останнього рівня усуне всі відрізки, помічені ij, але вона не усуне ні відрізка gh, який бачимо частково, ні відрізка kl, який цілком невидимий. Нехай a і b - кінці відрізка, тоді покажемо алгоритм, який визначає все повністю видимі і більшість невидимих відрізків.

Простий алгоритм визначення видимості: a і b - кінцеві точки відрізка в координатному просторі (x, y).

Для кожного відрізка треба перевірити повну видимість відрізка. Якщо одна з координат якого-небудь кінця відрізка знаходиться зовні вікна, то відрізок не є повністю видимим.

```
if(Xa<Xл)or(Xa>Xп) then GOTO 1
if(Xb<Xл)or(Xb>Xп) then GOTO 1
if(Ya<Yн)or(Ya>Yв) then GOTO 1
if(Yb<Yн)or(Yb>Yв) then GOTO 1
```

відрізок повністю видимий
візуалізувати відрізок
goto 3
Перевірити повну невидимість відрізка:
якщо обидва кінці відрізка лежать зліва, справа, зверху, знизу від вікна, то відрізок невидимий

```
1: if(Xa<Xл) and (XЬ<Xл) then GOTO 2
   if(Xa<Xп) and (Xb>Xп) then GOTO 2
   if(Ya<Yв) and (Yb>Yв) then GOTO 2
   if(Ya<Yн) and (Yb>Yн) then GOTO 2
```

Відрізок частково видимий або перетинає продовження діагоналі, залишаючись невидимим: Визначити перетин відрізка з вікном
2: відрізок невидимий
3: перехід до наступного відрізка

Тут Xл Xп Yв Yн- координати X і Y лівого, правого, верхнього, нижнього країв вікна відповідно.

Порядок проведення порівнянь при визначенні видимості або невидимості несуттєвий.

Для деяких відрізків може знадобитися проведення всіх 4-х порівнянь, перш ніж визначиться їх повна видимість або невидимість. Проте, оскільки визначення перетину відрізка з вікном вимагає великого об'єму обчислень, його слід проводити в останню чергу. В комп'ютерній графіці існують декілька методів відсікання. Одним з найпростіших є метод Коена і Сазерленда. В цьому методі для визначення тієї з 9 областей, якій належить кінець ребра, вводиться 4-розрядний бітовий код. Коди цих областей показані на рис. 13.2. Крайній правий біт коду вважається першим. У відповідний біт заноситься 1 при виконанні наступних умов:

для першого біта - якщо точка лівіше вікна
для другого біта - якщо точка правіше за вікно
для третього біта - якщо точка нижче за вікно
для четвертого біта - якщо точка вище за вікно

В противному випадку, в біт заноситься нуль. Звідси витікає, що якщо коди обох кінців ребра = 0, то обидві точки лежать усередині вікна, і відрізок видимий.

Рис. 13.2 Коди областей, яким належать кінцеві точки

Коди кінцевих точок можна використовувати також і для тривіального відкидання повністю невидимих відрізків. Розглянемо таблицю істинності, еквівалентну логічному оператору «и»

Істина і Брехня → Брехня 1і0→ 0
Брехня = 0
Брехня і Истина→Ложь 0і1→0
Брехня і Ложь→Ложь 0і0→0

Истина=1

Істина і Истина→Истина 1і1→1

Якщо побітовий логічний добуток кодових кінцевих точок відрізка не рівно нулю, то відрізок повністю невидимий, і його можна відкинути.

Таблиця 13.1 Коди кінців відрізків

Відрізок	Коди кінців	Результат логічного	Примітка
ab	0000 0000	0000	цілком бачимо
ij	0010 0110	0010	цілимо невидимий
ij	1001 1000	1000	цілимо невидимий
ij	0101 0001	0001	цілимо невидимий
lj	0100 0100	0100	цілимо невидимий
cd	0000 0010	0000	цілком видимий
ef	0001 0000	0000	цілком видимий
gh	0001 1000	0000	цілком видимий
kl	1000 0010	0000	целиком"невидим

З таблиці 13.1 видно, що якщо результат логічного множення не рівний нулю, то відрізок буде цілком невидимий. Проте, якщо логічний добуток = 0, то відрізок може виявитися цілком або частково видимим, або навіть цілком невидимим. Тому, для визначення повної видимості необхідно перевіряти значення кодів обох кінців відрізка окремо.

Перевірку значень кодів відрізків можна легко реалізувати, якщо скористатися підпрограмами, що оперують з бітами. Якщо знайдені цілком видимі і невидимі відрізки, то підпрограмі, що обчислює перетин відрізків, передаються тільки відрізки, які, можливо, частково видимі, тобто ті, для яких результат логічного множення кодів їх кінцевих точок рівний нулю. Ця підпрограма повинна правильно визначати передані їй такі повністю невидимі відрізки. Перетин 2-х відрізків можна шукати як параметричним, так і не параметричним способами. Очевидно, що рівняння нескінченної прямої, що проходить через точки.

$P_1(X_1, Y_1)$ і $P_2(X_2, Y_2)$ має вигляд

$y = m(x-x_1)+ y_1$ або

$y = т(x-x_2)+ y_2$ де

$m = (y_2-y_1)/(x_2-x_1)$ - це нахил даної прямої.

Точки перетину цієї прямої із сторонами вікна мають наступні координати:

з лівою $x_л$, $y = m(x_л-x_1)+ y_1 \quad m \neq \infty$

з правою $x_п$, $y = m(x_п-x_1)+ y_1 \quad m \neq \infty$

з верхньою $y_в$, $x = x_1 + (l/m)(y_в-y_1) \; m \neq 0$

з нижньою $y_н$, $x = x_1 + (l/m)(y_н-y_1) \; m \neq 0$

Щоб розробити схему ефективного алгоритму відсікання, необхідно спочатку розглянути декілька часткових випадків. Якщо нахил нескінченний, то відрізок паралельний лівій і правій сторонам вікна і треба шукати його перетин тільки з верхньою і нижньою сторонами. Якщо нахил = 0, то відрізок паралельний нижній і верхній сторонам вікна, а шукати його перетин треба тільки з лівою і правою сторонами. Якщо код одного з кінців відрізка рівний = 0, то цей кінець лежить усередині вікна, і тому відрізок може перетнути тільки одну сторону вікна.

В описаному вище алгоритмі передбачалося, що відсікаюче вікно є координатно-орієнтованим прямокутником. Проте, у багатьох випадках вікно не таке. Наприклад, припустимо, що прямокутне вікно повернено відносно системи координат так, як показано на малюнку 13.3

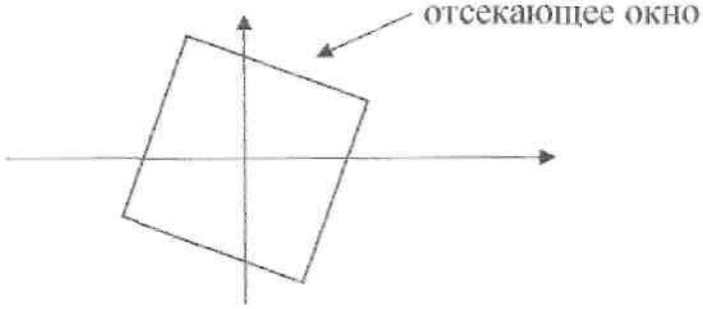

Рис. 13.3 Вікно, перевернене відносно осей

В цьому випадку не можна застосувати розглянутий алгоритм. Кирус і Бек запропонували алгоритм відсікання вікном довільної опуклої форми.

Алгоритм Кируса-Бека

Для створення надійного алгоритму відсікання потрібно мати хороший метод визначення розташування відносно вікна (всередині, на межі, зовні нього) точки, яка належить відрізку. Для цієї мети в алгоритмі Кируса-Бека використовується вектор нормалі.

Візьмемо опуклу відсікаючу область R. Вона може бути будь-яким плоским опуклим багатокутником. (Ми припускаємо, що вона двомірна, але ніяк цей багатокутник не може бути увігнутим).

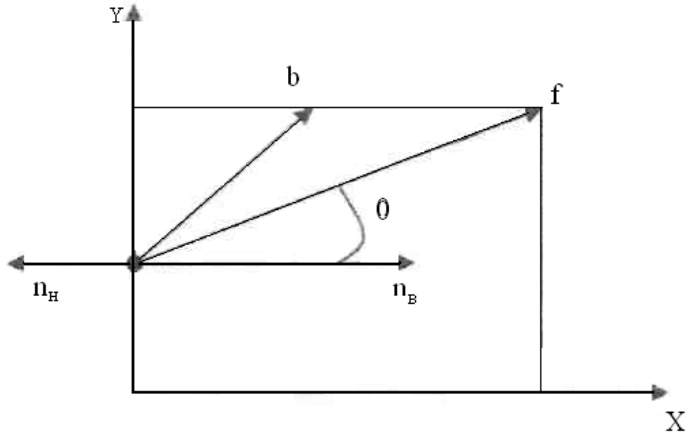

Рис 13.4 Внутрішні і зовнішні нормалі

Внутрішня нормаль n в довільній точці a, лежачої на границі R, задовольняє умову: $n*(b-a) \neq 0$, де b - будь-яка інша точка на границі R. Щоб переконатися в цьому, пригадаємо, що скалярний добуток двох векторів V_1 і V_2 дорівнює: $=*V_1*V_2*\ |V_1|\ |V_2|\ *\cos\theta$ де θ - менший з кутів, утворених V_1 і V_2.

Якщо $\theta = \pi/2$, то $\cos\theta = 0$ і $V_1*V_2=0$,тобто коли скалярний добуток пари векторів рівний нулю, то вони перпендикулярні. На рис. 13.4 показана опукла область R, тобто відсікаюче вікно. Тут же показані зовнішня $n_н$ і зовнішня $n_в$ нормалі до границі вікна, виходячи з точки a, лежачої на цій границі.

Крім того, на цьому малюнку показано ще декілька векторів, проведених з точки a в інші точки на границі вікна. Кут між $n_в$ і будь-яким з таких векторів завжди належить інтервалу - $\pi/2 \leq \theta \leq \pi/2$. При таких значеннях кута косинус його завжди позитивний, тому позитивний і скалярний добуток цих векторів. А ось кут між зовнішньою нормаллю і будь-яким з подібних векторів рівний ($\pi - \theta$) а cos ($\pi - \theta$)= -cosθ, при цьому негативний.

Якщо f — гранична точка опуклої області R, а n - внутрішня нормаль до однієї з обмежуючих цю область площин, то для будь-якої конкретної величини t, тобто для будь-якої точки відрізка P_1P_2 з умов n*[P(t)-f]<0, витікає, що вектор P(t)-f направлений зовні області R. З умови n*[P(t)-f]=0, витікає, що [P(t)-f] - належить площині, яка проходить через f і перпендикулярна нормалі. А з умови n*[P(t)-f]>0 виходить, що вектор [P(t)-f]-напрямлений всередину R. Звідси виходить, що нескінченна пряма перетинає замкнуту опуклу область (для двомірного випадку) в 2-х точках. Якщо ці дві точки не належать одній граничній площині або ребру, то рівняння n*[P(t)-f]=0 має тільки одне вирішення. Якщо точка f лежить на тій граничній площині або на тому ребрі, для яких n є внутрішньою нормаллю, то точка на відрізку P(t), яка задовольняє останньому рівнянню, буде точкою перетину цього відрізка з вказаною граничною площиною.

Для формалізації цього алгоритму приведемо параметричний опис відрізка: $P(t)=P_i+(P_2-P_1)t \quad 0 \le t \le 1$ (13.1)

А скалярний добуток внутрішньої нормалі на вектор, що починається в довільній точці відрізка і що закінчується в іншій точці, що лежить на тій же межі вікна. тобто

$n_i*[P(t)-f_i] \qquad i=1,2,3 \qquad (13.2)$

буде позитивний, рівно нулю або негативно - залежно від того, чи вибрана точка відрізка лежатиме з внутрішньої сторони границі, на самій границі або з її зовнішньої сторони. Останнє співвідношення може бути застосовано до будь-якої площини або ребра номер і, що обмежує область. Підстановка (13.1) в (13.2) дає умову перетину відрізка з границею області: $n_i*[P_1+(P_2 - P_1)t- f_i]=0$ (13.3)

В іншій формі рівняння (13.3) має вигляд: $n_i*[P_1-f_i]+n_{i*}(P_2 - P_1)t=0$ (13.4)

Вектор P_2 — P_1 визначати орієнтацію відрізка, а вектор P_1-f_i - пропорційний відстані від першого кінця відрізка до вибраної граничної точки.

Позначимо через $D = P_2 - P_1$ - діректрису або орієнтацію відрізка, а через $Wi=P_1-f_i$ - ваговий множник. Тоді рівняння (13.4) можна записати так:

$$t(n_i*D)+Wi*n_i=0 \qquad (13.5)$$

Вирішуючи останні рівняння відносно t маємо:

$$t = \frac{Wi*n_i}{D*n_i} \qquad D \neq 0 \qquad i=1,2,3 \; (13.6)$$

Тут $D*n_i$ може бути рівним нулю тільки при D=0, (а також при паралелі D вибраної границі) а це значить, що $P_2 = P_1$, тобто відрізок вироджується в точку. Якщо

$$W_i = \begin{cases} < 0 & \text{,то ця точка зовні вікна} \\ = 0 & \text{,то вона на границі вікна} \\ > 0 & \text{то вона усередині вікна} \end{cases}$$

Рівняння (13.6) використовується для отримання значень t, відповідних перетинам відрізка з кожною стороною вікна. Якщо t лежить за межами інтервалу $0 \le t \le 1$, то можна його проігнорувати. Хоча відомо, що відрізок може перетнути опукле вікно не більше ніж в двох точках, тобто при двох значеннях t рівняння (13.6) може дати велике число вирішень в інтервалі $0 \le t \le 1$. Ці вирішення слід розбити на 2 групи: нижню і верхню, залежно від того, до початку чи до кінця відрізка ближче відповідна точка. Потрібно знайти найбільшу з нижніх і якнайменшу з верхніх точок. Якщо $Di*n_i>0$, то значення *t* розглядається як можлива нижня границя. Якщо $Di*n_i<0$, то значення t розглядається як можлива верхня межа. Це і є алгоритм Кируса-Бека для двомірного відсікання.

Тривимірне відсікання

Двома найпоширенішими формами тривимірних відсікачів є: прямокутний паралелепіпед, використаний при паралельному проектуванні, а також усічена піраміда, часто називається пірамідою видимості, яка використовується при центральному проектуванні. Ці форми показані на рис. 1., у кожної з них шість граней: ліва, права, верхня, нижня, ближня, дальня. Існує необхідність відсікати і по стандартних об'ємах.

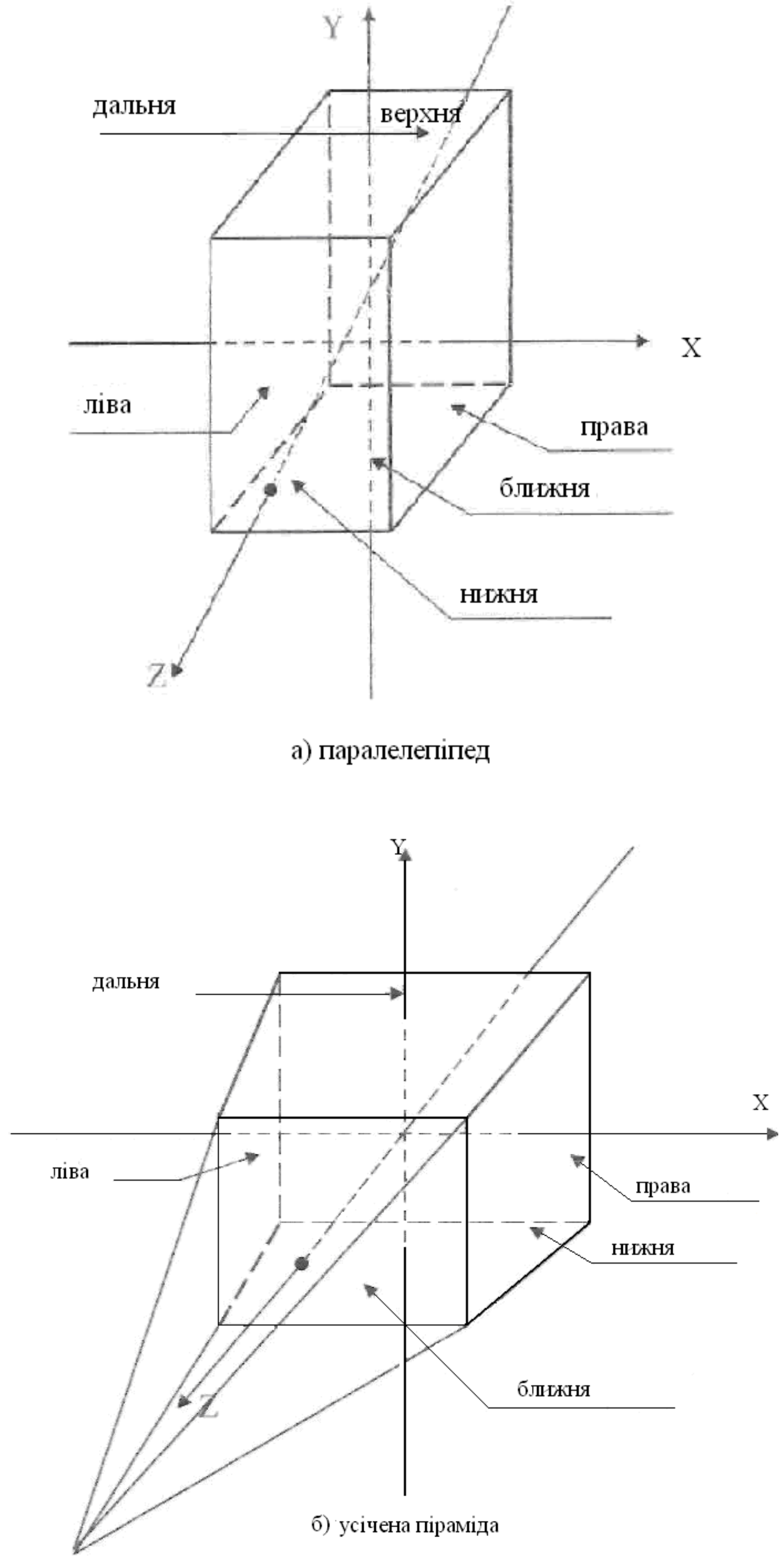

Рис 1

Як і при двомірному відсіканні, відрізки, які повністю видимі або тривіально невидимі, можна ідентифікувати з використанням узагальнення кодів кінцевих точок Коенна-Сазерленда. В тривимірному випадку використовується шестибітовий код. Самий правий біт коду вважається найпершим. В біти коду заносяться одиниці за допомогою узагальнення двомірної процедури. Конкретно, одиниця заноситься:

- в перший біт - якщо кінець ребра лівіше об'єму
- в – другий біт - якщо кінець ребра правіше за об'єм
- в третій біт - якщо кінець ребра нижче за об'єм
- в четвертий біт — якщо кінець ребра вище за об'єм
- в п'ятий біт - якщо кінець ребра ближче за об'єм
- в шостий біт - якщо кінець ребра далі за об'єм

В противному випадку у відповідні коди заносяться нулі. Якщо коди обох кінців відрізка рівні нулю, то обидва кінці видимі, і відрізок буде повністю видимий. Якщо побітовий логічний добуток кодів кінців відрізка не дорівнює нулю, то він повністю невидимий. Якщо ж цей логічний добуток рівний нулю, то відрізок може виявитися як частково видимим, так і повністю невидимим. В цьому випадку необхідно визначати перетин відрізка з гранями відсікаючого об'єму.

Пошук кодів точки відносно відсікаючого прямокутного паралелепіпеда є прямим узагальненням відповідного двомірного алгоритму. Проте, якщо відсікачем служить усічена піраміда, то тут необхідні додаткові дії.

Один з методів полягає в перетворенні відсікача в канонічну форму, де $X_{прав}=1$, $X_{лев}=-1$, $Y_{верх}=1$, $Y_{низ}=-1$,при $Z_{доп}=1$.

Якщо $Z_{ближ}=a$, де $0\leq a\leq 1$, а центр проекції співпадає з початком лівої системи координат, то перевірка кодів кінцевих точок помітно спрощується. В більш природному методі, менше викривляючи форму відсікача, відрізок, що сполучає центр проекції з центром усіченої піраміди, поєднується з віссю Z правої системи координат (рис. lb).

Вид усіченої піраміди зверху показаний на рис. 2а. Рівняння прямої на площині XZ, несучій проекцію правої грані відсікача має вигляд:

$x=(z-z_{цп})*X_п/(Z_D-Z_{цп})=za_1+a_2$
де $a_1=X_п/(Z_D-Z_{цп})$
$a_2=-a_1*Z_{цп}$

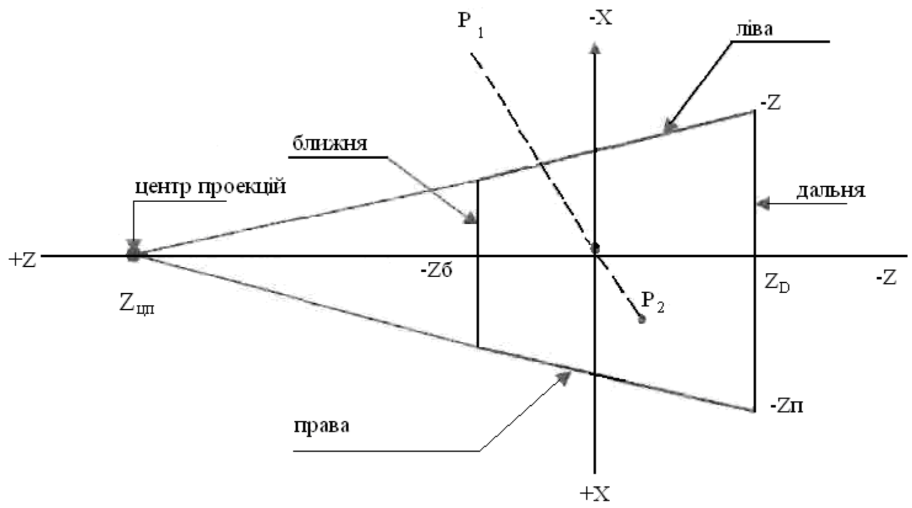

Рис 2 (а)

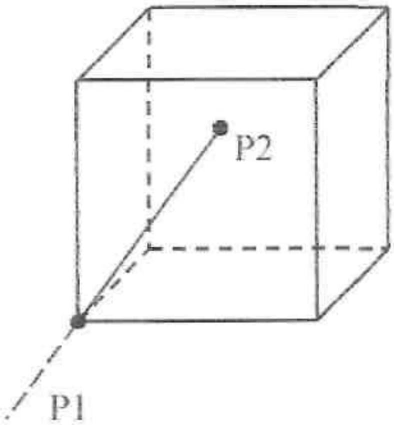

Рис. 2(b)Усічена піраміда

Рівняння цієї прямої можна використовувати для визначення розташування точки: справа, на або зліва від прямої, тобто зовні відсікача, на площині, несучій його праву грань або усередині відсікача. Підстановка координат X і Z точки P в пробну функцію правої грані дає наступний результат:

$$f_п = x - za_1 - a_2 \begin{cases} > 0, \text{ если P справа от плоскости} \\ = 0, \text{ если P на плоскости} \\ < 0, \text{ если P слева от плоскости} \end{cases}$$

Пробні функції для лівої, верхньої і нижньої граней мають вигляд:

$$f_л = z\beta_1 - \beta_2 \begin{cases} > 0, \text{ если P справа от плоскости} \\ =0, \text{ если P на плоскости} \\ <0, \text{ если P слева от плоскости} \end{cases}$$

Де $\beta_1 = X_л/(Z_D - Z_{цп}); \beta_2 = -\beta_1 Z_{цп}$

$$f_В = y - Z\gamma_1 - \gamma_2 \begin{cases} > 0, \text{ если P выше плоскости} \\ =0, \text{ если P на плоскости} \\ <0, \text{ если P ниже от плоскости} \end{cases}$$

де $\gamma_1 = \gamma_в/(Z_D - Z_{цп}); \gamma_2 = -\gamma_1 Z_{цп}$

$$f_Н = y - z\delta_1 - \delta_2 \begin{cases} < 0, \text{ если P выше плоскости} \\ =0, \text{ если P на плоскости} \\ >0, \text{ если P ниже от плоскости} \end{cases}$$

де $\delta_1 = y_н/(Z_D - Z_{цп}); = \bullet Z_{цп}$

Пробні функції для найближчої і дальньої граней:

$$f_Б = Z - Z_Б \begin{cases} > 0, \text{ если P ближе плоскости} \\ =0, \text{ если P на плоскости} \\ <0, \text{ если P дальше от плоскости} \end{cases}$$

$$f_Д = Z - Z_Д \begin{cases} <0, \text{ если P ближе плоскости} \\ =0, \text{ если P на плоскости} \\ >0, \text{ если P дальше от плоскости} \end{cases}$$

Чим ближче $Z_{ЦП}$ до нескінченності, тим більше форма відсікача наближається до прямокутного паралелепіпеда. Пробні функції при цьому теж наближаються до відповідних пробних функцій прямокутного паралелепіпеда.

Тривимірний алгоритм Кіруса- Бека

В тривимірному варіанті відсікач може бути довільним опуклим об'ємом.

Можна безпосередньо скористатися раніше розробленою двомірною версією. Тут К позначає не кількість сторін багатокутника, а число граней многогранника. Всі вектори тепер мають по 3 компоненти: x,y,z. Відсікання відносно стандартної піраміди видимості трошки складніше. Тут внутрішні нормалі до граней відсікача слід визначити формально, оскільки їх значення неочевидні. Необхідно помітити, що число операцій в алгоритмі Кіруса -Бека росте лінійно із зростанням числа сторін граней у відсікача.

Визначення опуклості тривимірного тіла і обчислення внутрішньої нормалі до його граней.

Раніше описаний двомірний алгоритм визначення опуклості багатокутника і обчислення його внутрішніх нормалей, які використовують повороти і перенесення, можна узагальнити на випадок тривимірних многогранників.

Для кожної полігональної грані тіла виконати наступне:

- Перенести тіло так, щоб одна з вершин грані виявилася на початку координат.
- Повернути тіло такої навколо вибраної осі координат, щоб вибрана грань лягла на координатну площину (наприклад на площину Z=0)
- Для всіх вершин тіла, що не належать вибраній грані, перевірити знаки координати, яка перпендикулярна цій грані (тут це буде координата Z).

Якщо тіло опукло відносно всіх граней, то воно вважається опуклим, в інших випадках неопуклим.

- Повернути тіло відносно початку координат так, щоб одна з двох суміжних вибраній вершині сторін грані співпала з однією з осей координат (наприклад з віссю X).

Якщо для всіх вершин значення координати, перпендикулярної вибраної грані, рівні нулю, то тіло вироджене, тобто воно плоске.

Вектор внутрішньої нормалі до вибраної площини, заданий в поверненій системі координат, має всі нульові компоненти, окрім тієї, яка перпендикулярна цій площині. Знак цієї компоненти для опуклої грані співпадатиме з раніше знайденим знаком.

Для визначення шуканої орієнтації внутрішньої нормалі в початковій системі координат, потрібно застосувати до неї тільки зворотне перетворення поворотів.

Послідовне відсікання багатокутника - Сазерленда-Ходжмена

В попередніх розділах ми розглядали відсікання відрізків. Багатокутник можна розглянути як набір відрізків.

Якщо замкнутий багатокутник відсікається як набір відрізків, то початкова фігура може перетворитися на один або більше відкритих багатокутників, або просто стати сукупністю розрізнених відрізків (як показано на рис. 3)

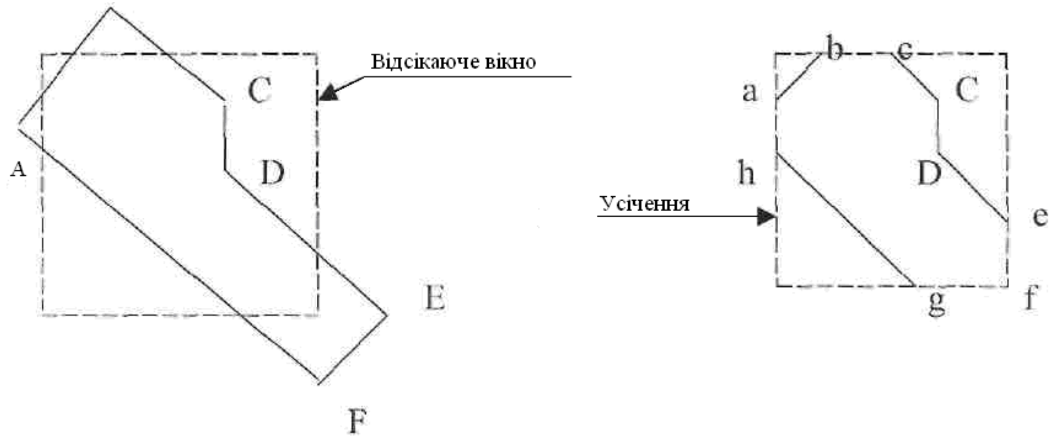

Рис. 3 Відсікання багатокутника: відкритий багатокутник

Проте, якщо багатокутники розглядаються як суцільні області, то необхідно, щоб замкнутість зберігалася і у результату.

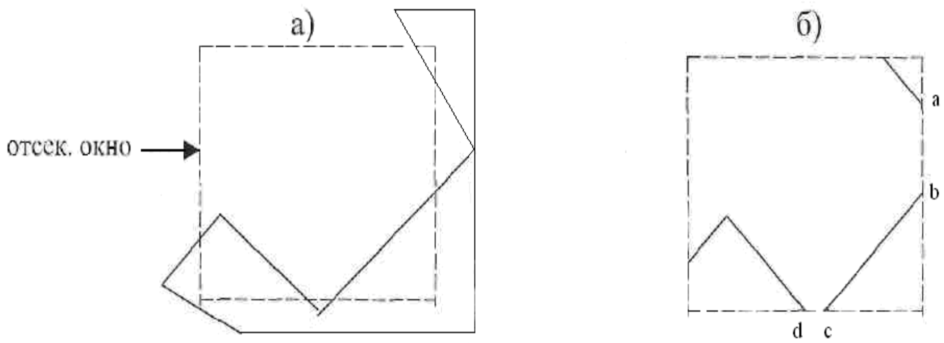

Рис. 4 Відсікання багатокутника: не зв'язані між собою багатокутники

Для прикладу рис. 3 це означає, відрізки bc, ef, fg, ha повинні бути додані до опису результуючого багатокутника. Додавання відрізків ef і fg представляють особливі труднощі. Вони зв'язані з тим, що якщо результат відсікання являє собою декілька незв'язаних між собою багатокутників менших розмірів, як показано на рис 4, то необхідно провести такі дії. Наприклад, іноді відрізки ab і cd (рис. 4) включаються в опис результату. Якщо наприклад початковий багатокутник оголошений жовтим на синьому фоні, то відрізки ab і cd теж виглядатимуть жовтими на синьому фоні. Це суперечить очікуваному результату.

Основна ідея алгоритму Сазерленда-Ходжмена полягає в тому, що відсікти багатокутник відносно однієї прямої або площини дуже легко. В цьому алгоритмі початковий і кожний з проміжних багатокутників відсікається послідовно однією прямою. Робота алгоритму для прямокутного вікна показана на рис. 5. Початковий багатокутник задається списком вершин $P_1 ... P_n$, який породжує список його ребер P_1, P_2, P_3 ...,P_n P_1 .На рис. 5 показано, що багатокутник спочатку відсікається лівою стороною вікна, внаслідок чого виходить проміжна фігура. Потім алгоритм знов відсікає цю фігуру верхньою стороною вікна. Виходить друга проміжна фігура. Далі процес відсікання продовжується із сторонами вікна, що залишилися. Всі етапи показані на рис. 5

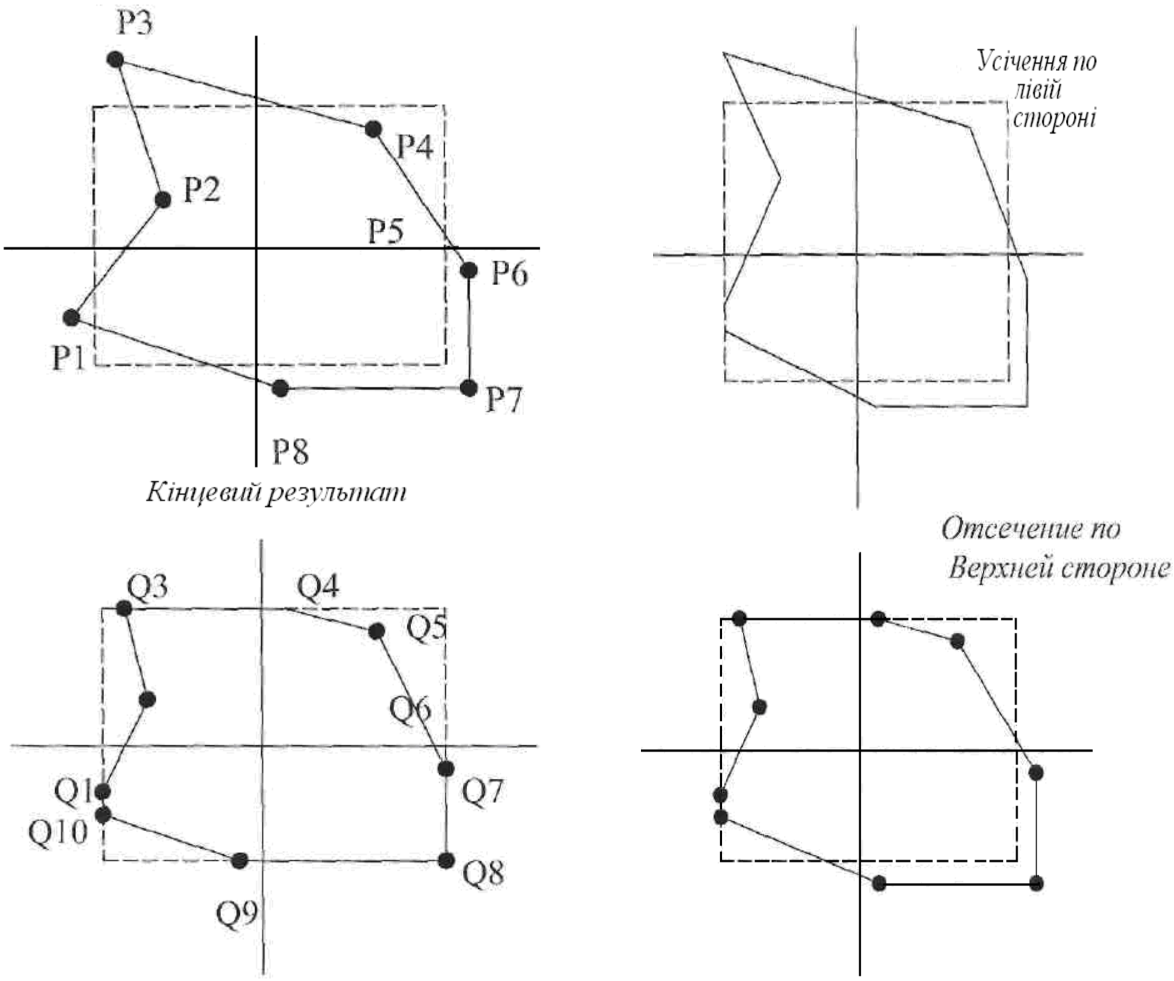

Рис. 5 Послідовне відсікання багатокутника

Помітимо, що додавання кутової точки Q8 в остаточний результат відсікання тепер стало тривіальним. Цей алгоритм здатний відсікати будь-який багатокутник, опуклий або не опуклий, плоский або неплоский, відносно будь-якого вікна, що є опуклим багатокутником. Порядок відсікання багатокутника різними сторонами вікна непринциповий.

Результатом роботи алгоритму є список вершин багатокутника, у якого всі вершини лежать по видиму сторону від чергової відсікаючої площини. Оскільки кожна сторона багатокутника відсікається незалежно від інших, то достатньо розглянути тільки можливі ситуації розташування одного відрізка відносно однієї відсікаючої площини. Розглядатимемо кожну точку P із списку вершин багатокутника за винятком першої, як кінцеву точку ребра початковою точкою S, якого є вершина, передуюча P в цьому списку. Тоді можливі тільки чотири ситуації взаємного розташування ребра і відсікаючої площини.

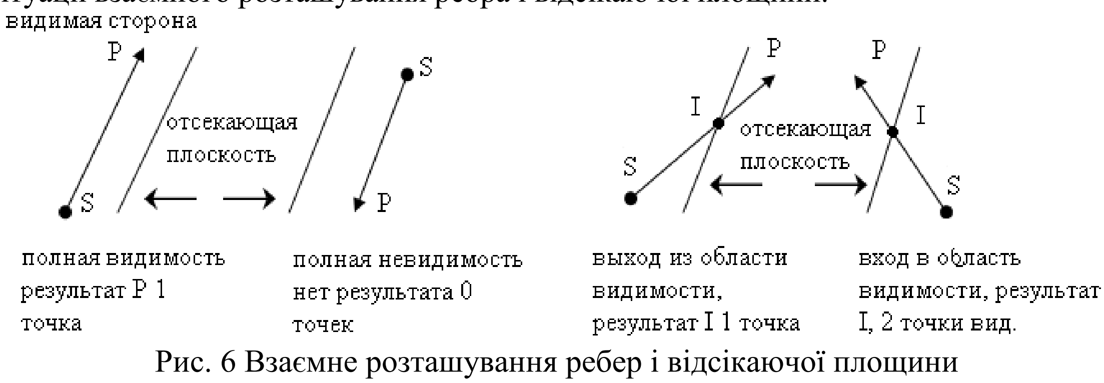

Рис. 6 Взаємне розташування ребер і відсікаючої площини

Результатом кожного зіставлення ребра багатокутника з відсікаючою площиною буде занесення в список вершин результуючого усіченого багатокутника нуля, однієї або двох вершин. Якщо дане ребро повністю видимо, то результатом буде вершина P. Заносить в результат початкову вершину S в цьому випадку не треба, оскільки якщо вершини розглядаються по черзі, то S вже була кінцевою точкою попереднього ребра і тому вже потрапила в результат. Якщо ж ребро повністю невидимо, то результат не змінюється.

Якщо ребро багатокутника видиме не повністю, то воно може або входити, або виходити з області видимості відсікаючої площини. Якщо ребро виходить з області видимості, то треба визначити і занести в результат точку перетину ребра і відсікаючої площини. Якщо ребро входить в область видимості, то слід поступити точно також. Оскільки в останньому випадку кінцева вершина P ребра видима, то вона також повинна потрапити в результат.

Для першої вершини багатокутника необхідно визначити тільки факт її видимості. Якщо вершина видима, то вона потрапляє в результат, і стає початковою точкою S. Якщо ж вершина невидима, то вона також стає початковою точкою, але в результат не потрапляє.

Останнє ребро P_nP_1 розглянемо особливо. Це реалізується шляхом запам'ятовування першої вершини багатокутника в F. Тоді останнім стає PNF I його можна обробляти точно також, як і будь-яке інше ребро.

Тут потрібно ще два додаткові міркування. Визначення видимості точки еквівалентно визначенню тієї сторони границі відсікаючої площини, по яку лежить ця точка. Якщо ребра відсікаючого багатокутника обходяться за годинниковою стрілкою, то його внутрішність лежить праворуч від границі. При протилежному порядку обходу вона лежить ліворуч. Раніше розглядалися 2 методи визначення видимості точки відносно орієнтованого відрізка або площини. Другий метод - метод підстановки координат пробної точки в рівняння орієнтованої прямої на площині, є варіантом того, що було запропоноване Сазерлендом і Ходжменом. Третій метод визначення видимості зводиться до перевірки знака координати Z у векторного добутку двох векторів, що лежать в площині. Нехай дві точки P_1 і P_2 лежать на відсікаючій площині, а P_3 - це пробна точка. Ці три точки задають якусь площину, на якій лежать два вектори: P_1P_2 і P_1P_3. Якщо цю площину вважати площиною XY, то у векторного добутку $P_1P_2 * P_1P_3$ ненульовий буде тільки компонента Z, рівна $(X_3 - X_1)(Y_2 - Y_1) - (Y_3 - Y_1)(X_2 - X_1)$.

Якщо знак цієї компоненти Z буде позитивним, нульовим або негативним, то P_3 лежатиме відповідно справа, на або зліва від прямої P_1P_2. Всі ці методи реалізуються особливо просто для випадку прямокутних відсікаючих вікон, сторони яких паралельні координатним осям.

При використовуванні цих текстів видимості, ребро багатокутника буде повністю видимим, якщо обидва його кінці видимі, і повністю невидимим, якщо обидва вони невидимі. Якщо ж один кінець ребра бачимо, а інший невидимий, то ребро перетинається з відсікаючою площиною і потрібне обчислювати точку перетину. Для цього можна використовувати будь-які викладені вище алгоритми відсікання відрізка (наприклад, Кируса-Бека). Перетин двох параметрично заданих відрізків, що лежать в одній площині, вимагає уточнення.

Два відрізки з кінцевими точками : P_1P_2 і P_4P_3 відповідно можна задати параметрично таким чином:

$P(S)= P_1 + (P_2-P_1)S \qquad 0\leq S\leq 1$

$P(t)= P_3 +(P_4 - P_3)t \qquad 0\leq t\leq 1$

В точці їх перетину P(S)= P(t).

P(S) і P(t) - є векторно-значними функціями, тобто P(S)= [X(S)Y(S)] P(t)= [X(t)Y(t)]

Звідси витікає, що з векторного рівняння виходить два скалярні рівняння з двома невідомими S і t, тобто в точці перетину X(S)= X(t)

Y(S)=Y(t)

Якщо останнє рівняння взагалі не має розв'язку, то відрізки паралельні. Якщо ж розв'язок є, тобто S або t виходять за межі допустимої області, то відрізки не мають загальних точок. Особливо зручна матрична форма запису останнього рівняння.

Сазерленд і Ходжменд запропонували новий метод формування послідовності проміжних багатокутників. В їх алгоритмі ребра багатокутника обробляються по черзі. А це вказує на те, що можна використовувати з мінімальними змінами колишні коди кінцевих точок ребер.

Сазерленд і Ходжменд показали, як можливо уникнути породження і запам'ятовування вершин проміжних багатокутників. Для цього замість відсікання кожного ребра багатокутника однією площиною, що обмежує вікно, треба відсікати кожне таке ребро послідовно всіма границями вікна.

Після відсікання чергового ребра багатокутника по одній з границь вікна, алгоритм рекурсивно звертається до самого себе, щоб відсікти результат попереднього звертання по наступній границі вікна. Ця властивість робить даний алгоритм більш зручним для апаратної реалізації.