Міністерство освіти і науки України
Запорізька державна інженерна академія

Затверджено до друку
рішенням вченої ради ЗДІА
протокол № від

КОМП'ЮТЕРНА ГРАФІКА

Навчально-методичний посібник

для студентів ЗДІА
спеціальності 8.080403, 7.080403
"Програмне забезпечення автоматизованих систем"

Рекомендовано до друку
на засіданні кафедри ПЗАС,
протокол № 18 від 26 квітня2016 р.

Комп'ютерна рафіка. Навчально-методичний

посібник для студентів ЗДІА спеціальності 8.080403,7.080403 *"Програмне забезпечення автоматизованих систем"*

/Укладач В.І. Заяц. – Запоріжжя: Вид. ЗДІА, 2016 – 81 с.

Відповідальний за випуск: **зав. кафедри ПЗАС**

проф. Шамровський О.Д.

Рецензенти:

Пазюк М.Ю. д.т.н., професор кафедри автоматизованого управління технологічними процесами Запорізької державної інженерної академії

Рисіков В.П. к.т.н., доцент кафедри програмних засобів Запорізького національного технічного університету

ЗМІСТ

Розділ 1.Сфера застосування і класифікація алгоритмів комп’ютерної графіки 5
Розділ 2. Алгоритми растрової графіки 7
2.1 Растрове розгорнення відрізка 7
2.2 Згладжування сходового дефекту 11
2.3 Растрове розгорнення кола 14
Контрольні питання 16
Лабораторна робота №1 17
2.4 Заповнення двомірних областей 17
Контрольні питання 21
Лабораторна робота №2 22
Розділ 3. Відтинання 22
3.1 Розбиття довільного багатокутника на опуклі складові 22
Контрольні питання 24
Лабораторна робота №3 24
3.2 Одновимірне відтинання 24
3.3 Відтинання багатокутників 30
Контрольні питання 33
Лабораторна робота №4 34
Розділ 4. Елементи координатної геометрії 34
4.1 Однорідні координати 34
4.1 Матриці основних перетворень 37
4.2 Визначення видових координат об'єкта 38
4.3 Проективні перетворення 40
Контрольні питання 43
Лабораторна робота №5 43
Розділ 5. Видалення невидимих ліній і поверхонь 44

5.1 Класифікація методів видалення невидимих поверхонь 44
5.2 Метод Робертса 46
5.2 Метод Z-буфера і його варіанти 47
5.3 Алгоритм Варнака 50
5.4. Метод трасування променів 54
Контрольні питання 56
Лабораторна робота №6 56
Розділ 6. Побудова реалістичних зображень 57
6.1 Моделі освітлення 57
6.2 Моделі зафарбування 62
Контрольні питання 64
Лабораторна робота №7 65
6.3 Спеціальні ефекти при побудові зображень 65
6.4 Системи представлення кольорів 69
Розділ 7 Задачі і методи когнітивної графіки 75
Контрольні питання 80
Література 81

РОЗДІЛ 1
СФЕРА ЗАСТОСУВАННЯ І КЛАСИФІКАЦІЯ АЛГОРИТМІВ КОМП'ЮТЕРНОЇ ГРАФІКИ

Будучи однієї з галузей інформаційних технологій, комп'ютерна графіка застосовується в найрізноманітніших галузях людської діяльності. Найважливішими сферами застосування комп'ютерної графіки є комп'ютерне моделювання, системи наукових досліджень, системи автоматизованого проектування, конструювання і виробництва, системи автоматизованого управління, бізнес, мистецтво, засоби масової інформації і навіть дозвілля і віртуальна реальність.

Створено велику кількість програм, для рішення задач цього класу, причому ці програми орієнтовані на дуже широкий діапазон підготовки користувача. Безсумнівно, що професійному програмістові необхідно не тільки в досконалості володіти наявним інструментарієм, але знати основні підходи до рішення задач цього класу й алгоритмічні принципи їхньої реалізації.

В даний час загальноприйнятим є ділення комп'ютерної графіки по наступним напрямках:

Комп'ютерна графіка для побудови зображень: побудова моделей об'єктів і генерація зображень; перетворення моделей і зображень; ідентифікація об'єктів і отримання необхідної інформації;

Обробка і аналіз зображень: оцінка зображень: визначення форми, місця знаходження, розмірів та інших параметрів об'єктів, розпізнавання зображень обробка знімків, введення креслень, системи навігації..

Аналіз сцен: виділення характерних особливостей, що формують графічний об'єкт

Комп'ютерна графіка для наукових досліджень і абстракцій.

Алгоритми машинної графіки можна розділити на два рівні : нижній і верхній. Група алгоритмів нижнього рівня призначена для реалізації графічних примітивів (ліній, кіл, заповнень і т.п.). Ці алгоритми або їм подібні відтво-

рені в графічних бібліотеках мов високого рівня або реалізовані апаратно в графічних процесорах робочих станцій.

Серед алгоритмів нижнього рівня можна виділити наступні групи : Найпростіші в розумінні і використовуваних математичних методів і реалізації, що відрізняються простотою. Як правило, такі алгоритми не є найкращими за обсягом виконуваних обчислень і необхідних ресурсів пам'яті.

Тому можна виділити другу групу алгоритмів, що використовують більш складні математичні передумови (але часто і евристичні) і відрізняються більшою ефективністю.

До третьої групи варто віднести алгоритми, що можуть бути без значних зусиль реалізовані апаратно (які допускають розпаралелювання, рекурсивні, реалізовані в найпростіших командах). У цю групу можуть потрапити й алгоритми, представлені в перших двох групах.

Нарешті, до четвертої групи можна віднести алгоритми з спеціальним призначенням (наприклад, для усунення сходового ефекту).

До алгоритмів верхнього рівня належать в першу чергу алгоритми видалення невидимих ліній і поверхонь. Задача видалення невидимих ліній і поверхонь продовжує залишатися центральною в машинній графіці. Від ефективності алгоритмів, що дозволяють вирішити цю задачу, залежать якість і швидкість побудови тривимірного зображення.

До задачі видалення невидимих ліній і поверхонь примикає задача побудови (зафарбовування) напівтонових (реалістичних) зображень, тобто врахування явищ, зв'язаних з кількістю і характером джерел світла, врахування властивостей поверхні тіла (прозорість, переломлення, відбиття світла). Однак при цьому не слід забувати, що виведення об'єктів в алгоритмах верхнього рівня забезпечується примітивами, що реалізують алгоритми нижнього рівня, тому не можна ігнорувати проблему вибору і розробки ефективних алгоритмів нижнього рівня.

Когнітивна (від лат. cognoscere – знати, дізнаватися, розслідувати) комп'ютерна графіка - це комп'ютерна графіка для наукових абстракцій, що

сприяє народженню нового наукового знання. Когнітивна комп'ютерна графіка є ефективним інструментом впливу на образне інтуїтивне мислення дослідника. Функція когнітивної графіки полягає в наочному зображенні сутності об’єкта або процесу, яким може бути, зокрема, будь-яке абстрактне наукове поняття, гіпотеза або теорія. Змістом когнітивної графіки є відображення (візуалізація) деяких математичних закономірностей для їх більш глибокого розуміння В сучасних інформаційних технологіях когнітивна графіка визначається як сукупність прийомів і методів образного уявлення умов завдання, яке дозволяє або відразу побачити рішення, або отримати підказку для його знаходження.

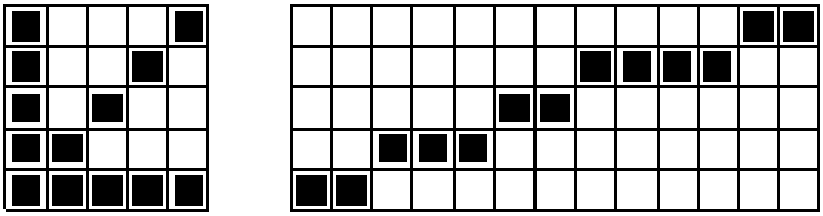
Для різних областей застосування машинної графіки на перший план можуть висуватися різні властивості алгоритмів. Для наукової графіки велике значення має універсальність алгоритму, швидкодія може відходити на другий план. Для систем моделювання, що відтворюють об'єкти, що рухаються, швидкодію стає головним критерієм, оскільки потрібно генерувати зображення практично в реальному масштабі часу.

РОЗДІЛ 2

2.1АЛГОРИТМИ РАСТРОВОЇ ГРАФІКИ.

2.1 Растрове розгорнення відрізка .

Особливості растрової графіки зв'язані з тим, що звичайні зображення, з якими зустрічаються людина у своїй діяльності (креслення, графіки, карти, художні картини і т.п.), реалізовані на площині, що складається з безкінечного набору точок. Екран же растрового дисплея представляється матрицею дискретних елементів, що мають конкретні фізичні розміри і називаються пікселями. При цьому число їх істотне обмежено. Тому не можна провести точно лінію з однієї точки в іншу, а можна виконати тільки апроксимацію цієї лінії з відображенням її на дискретній матриці (площині), рис. 1.



a) b)

Рис. 1

Таку площину також називають цілочисловою решіткою, растровою площиною або растром. Ця решітка представляються квадратною сіткою з кроком 1. Фактично пікселі мають форму прямокутника, але надалі для простоти викладу будемо припускати, що вони є квадратними. Положення пікселя визначається координатами його лівого нижнього кута.

Відображення будь-якого об'єкта на цілочислову решітку називається розкладанням його в растр або просто растровим представленням(розгорненням). Це означає, що заданої геометричної фігури слід поставити у відповідність множину на цілочисленій площині, яка в певному сенсі наближає безперервний об'єкт

Задача зображення відрізка є однієї з основних у машинній графіці, оскільки від успіху її рішення залежить якість зображення сцени в цілому. В переважній більшості випадків тіла обмежені криволінійними поверхнями представляються у вигляді багатогранників, і як наслідок, основним геометричним об’єктом комп’ютерної графіки є багатокутник. Як буде показано нижче, такий підхід дозволяє як суттєво скоротити обсяг обчислень, так і спростити необхідні геометричні перетворення.

До зображення відрізків пред'являються звичайно наступні вимоги:

- відрізок повинний починатися і закінчуватися в заданих точках;

- відрізок повинний здаватися прямим;

- яскравість відрізка повинна бути постійною по всій довжині відрізка і не залежати від коефіцієнта кутового нахилу.

Очевидно, що ці вимоги можна виконати тільки для строго горизонтальних, вертикальних і відрізків з кутовим коефіцієнтом $k = 1$, рис 1,а. Хоча в

останньому випадку яскравість відрізка буде в $\sqrt{2}$ менше ніж для вертикального і горизонтального відрізків.

При побудові зображення відрізка за допомогою дискретної множини пікселів неминуче виникає дефект, який називають драбинним або сходовим дефектом.

Найпростішим методом растрової розгортки відрізка є *цифровий диференціальний аналізатор*(ЦДА). Нехай відрізок задано його кінцевими точками (x_b, y_b) і (x_e, y_e). Тоді рівняння прямої, яка містить відрізок можна одержати як рішення диференціального рівняння

$$\frac{dy}{dx} = k, \qquad k = \frac{y_e - y_b}{x_e - x_b}$$

з початковою умовою $y(x_b) = y_b$. Рішення рівняння має вигляд

$$y = y_b + k(x - x_b),$$

а побудова відрізка зводиться до реалізації простої рекурентної формули.

$$y_{i+1} = y_i + k. \qquad (1.1)$$

Побудований таким чином відрізок має яскраво виражений сходовий дефект, постільки цілочислова координата пікселя визначається з (1.1) шляхом операції округлення, яка приводить до випадкової похибки.

Алгоритм Брезенхема дозволяє значно зменшити дефект ступінчастості. В основі методу лежить природне припущення, що зафарбувати треба той піксель, відстань від якого по відповідній координаті до відрізка менше, ніж у інших пікселів. Припустимо, що побудова ведеться у напрямку збільшення координати x і кутовий коефіцієнт в рівнянні лінії відрізка $0 < k \le 1$. Тоді проблема вибору наступної точки зведеться до аналізу проходження відрізку через два сусідні пікселі з ординатами y_i і y_{i+1}. Оскільки в межах пікселя координата може змінитися не більше ніж на одиницю, то у випадку коли відстань $\Delta = y - y_i \le \frac{1}{2}$, то зафарбовується нижній піксель, якщо $\Delta > \frac{1}{2}$

– верхній y_{i+1}. Головною незручністю у цьому підході є обчислення величини Δ.

На рис. 2 показані два перших кроки побудови відрізка прямої з коефіцієнтом нахилу $k = \frac{2}{5}$, у яких відстань Δ обчислювалась безпосередньо. Тоді похибка між точною ординатою і зображенням складає відповідно для першого і другого кроку $e = k$ і $e = 1 - 2k$. Брезенхем запропонував знак цієї похибки взяти в якості керуючого параметр алгоритму.

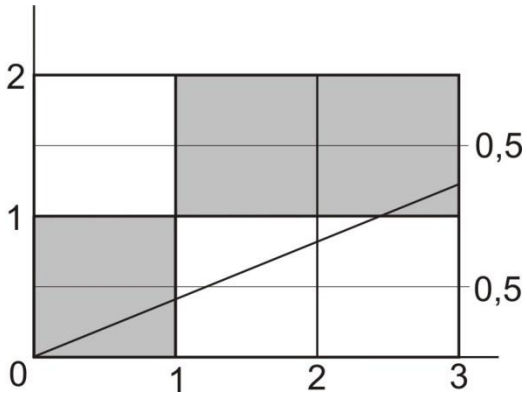


Рис.2

Координата x в обох випадках одержує одиничне збільшення: $x_{i+1} = x_i + 1$. Початкове значення похибки покладається рівним $e = -0{,}5 + k$ і відрізок починається в точці (x_b, y_b). Алгоритм подальшого вибору точок для візуалізації складається з наступних кроків:

1. Збільшити координату x на одиницю $x_{i+1} = x_i + 1$;

2. Якщо $e > 0$ то $y_{i+1} = y_i + 1$ і $e = e - 1$

у противному випадку покладається $y_{i+1} = y_i$ і $e = e + k$.

Наведений варіант алгоритму Брезенхема є дещо неефективним в обчислювальному плані, а також містить обмеження на співвідношення координат кінцевих точок відрізка. А саме, використовуються дії над дійсними числами, і неявно прийнято, що $x_e > x_b$, $y_e > y_b$ і $k < 1$. Цих моментів досить легко уникнути. Введемо позначення $\Delta x = x_e - x_b$ $\Delta y = y_e - y_b$ і замінимо ве-

личину оцінки $e = k - \frac{1}{2} = \frac{\Delta y}{\Delta x} - \frac{1}{2}$ на $\bar{e} = 2\Delta x e = 2\Delta y - \Delta x$. Тоді в обчисленнях будуть тільки арифметичні дії над цілими числами.

Введемо позначення

$h_x = 1$ при $\Delta_x > 0$ і $h_x = -1$, при $\Delta_x < 0$;

$h_y = 1$ при $\Delta_y > 0$ і $h_y = -1$, при $\Delta_y < 0$.

Тоді у випадку $\Delta_y < \Delta_x$, $k < 1$ генерація відрізка зводиться до наступної послідовності дій

$x_{i+1} = x_i + h_x$

$\bar{e} < 0 \Rightarrow\ y_{i+1} = y_i$, $\qquad \bar{e} = \bar{e} + 2\Delta y$

$\bar{e} > 0 \Rightarrow\ y_{i+1} = y_i + h_y$, $\qquad \bar{e} = \bar{e} - 2\Delta x$

Щоб зняти обмеження $k > 1$ у випадку $\Delta_y > \Delta_x$ координати x, y треба поміняти місцям. Тоді $k = \frac{\Delta x}{\Delta y} < 1$ і вище наведена процедура набуває форму

$y_{i+1} = y_i + h_y$

$\bar{e} < 0 \Rightarrow\ x_{i+1} = x_i$, $\qquad \bar{e} = \bar{e} + 2\Delta x$.

$\bar{e} > 0 \Rightarrow\ x_{i+1} = x_i + h_x$, $\bar{e} = \bar{e} - 2\Delta y$.

2.2 Згладжування сходового дефекту

Растрова генерація відрізків має наступні недоліки:

- неточне розташування початку і кінця;
- отримане зображення має вигляд сходинки;
- яскравість залежать від нахилу і довжини.

Ясно, що перший недолік не може бути усунуто програмним шляхом. Нерівномірність яскравості не дуже помітне, і може бути компенсоване очевидним шляхом, що потребує взагалі дійсної арифметики, але в зв'язку з реально невеликим числом рівнів яскравості, достатньо обійтись досить грубим цілочисловим наближенням. Найбільш помітно погіршує якість зображення сходовий дефект. Є наступні способи боротьби з цією проблемою:

- збільшення просторового роз рішення за рахунок удосконалення апаратури;
- трактова пікселя не як точки, а як площадки скінченого розміру, яскравість, якої залежить від розміру пікселя, охопленого зображенням відрізка:
- розмиттям" різкої границі, за рахунок часткового освітлення сусідніх пікселів.
- Розглянемо модифікований алгоритм Брезенхема , у якому піксель розглядається як об'єкт, що має скінченні розміри. Площа багатокутника обмеженого відрізком , що проходить через піксель (x_i, y_i) може бути обчислена як площа трапеції і дорівнює

$$S = y_i + \frac{k}{2}$$

Якщо тепер ввести нові змінні у алгоритмі Брезенхема

$$w = 1 - k, \qquad \bar{e} = e + w$$

то модифікована похибка $\bar{e}$ буде змінюватися в межах $0 \le \bar{e} \le 1$ і її можна взяти в якості міри яскравості. Тоді модифікований алгоритм набуде вигляду

Задається початкове значення похибки. $\bar{e} = \frac{1}{2}$ і починаючи з лівої кінцевої точки, до тих пір поки не буде досягнута права кінцева точка по абсцисі

$x_{i+1} = x_i + 1$

якщо $\bar{e} \le w \Rightarrow \quad \bar{e} = \bar{e} + k$

якщо $\bar{e} > w \quad \Rightarrow y_{i+1} = y_i + 1 \quad \bar{e} = \bar{e} - 1$

На рис.3 відповідно зверху вниз показана побудова відрізка за допомогою безпосередньо алгоритму Брезенхема, зафарбування пікселя з яскравістю пропорційно фактичній площі, що відсікає відрізок від пікселя і по модифікованому алгоритму Брезенхема.

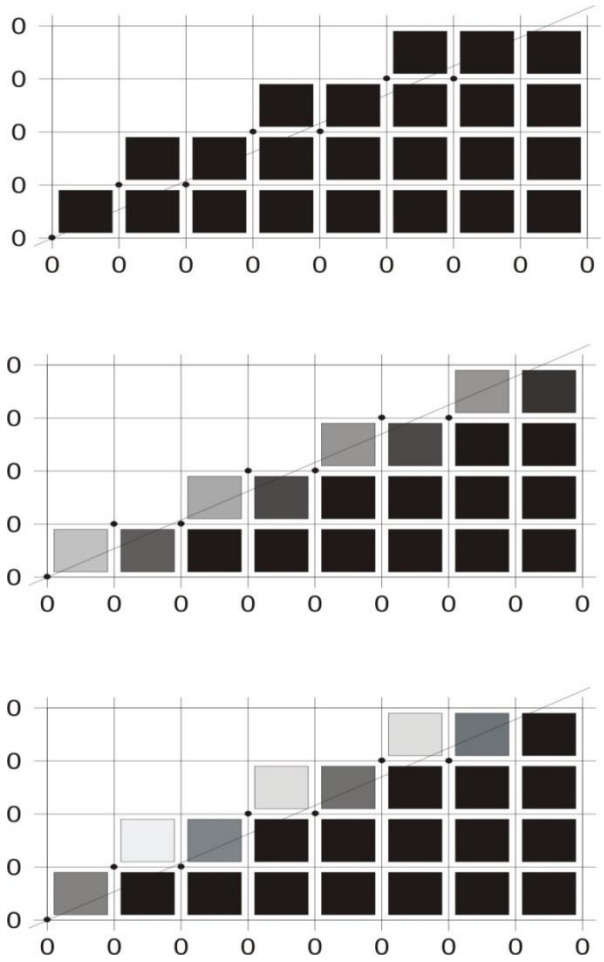


Рис.3

2.3 Растрове розгорнення кола

В багатьох додатках, таких як системи автоматизованого проектування в машинобудуванні, природними графічними примітивами, окрім відрізків прямих і рядків текстів, є конічні перетини, тобто кола, еліпси, параболи і гіперболи. Одним з найпростіших і ефективних алгоритмів рішення цієї задачі є підхід Брезенхема, розширений на випадок растрового розгорнення кривих другого порядку, Нижче така побудова проводиться для випадку кола, як найбільш поширеного примітива. При побудові зображення кола по його параметричному завданні або безпосередньо по формулі $y = \sqrt{R^2 - x^2}$, крім недоліків візуалізації існують труднощі чисто обчислювального плану, обумовлені застосуванням вбудованих функцій. Ідея методу полягає у визначенні одного з трьох пікселів для візуалізації: горизонтального (H), діагонального (D) або вертикального (V), для якого відстань до дуги кола буде найменшою. Більш зручно обчислювати не саму цю відстань, а різницю між її квадратом і квадратом радіуса кола

$$m_H = (x_i + 1)^2 + y_i^2 - R^2$$

$$m_D = (x_i + 1)^2 + (y_i - 1)^2 - R^2$$

$$m_V = x_i^2 + \left(y_i^2 + 1\right)^2 - R^2$$

Далі алгоритм викладається в припущенні, що побудова зображення відбувається при обході кола по напрямку годинної стрілки, і для рішення даної задачі досить побудувати коло тільки в одному першому октанті (рис .4). Після чого повне зображення будується шляхом послідовних відображень:

$$\text{I} \Rightarrow \text{II}, \qquad (\text{I, II}) \Rightarrow (\text{IV, III}), \qquad (\text{I, II, III, IV}) \Rightarrow (\text{VIII,VII,VI,V})$$

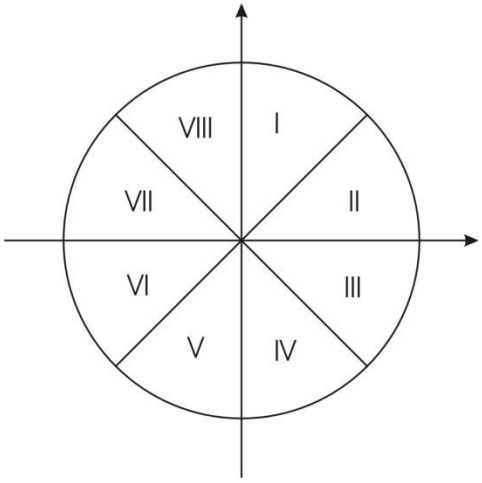


Рис .4

Як відомо, при виконанні нерівності $x_i^2 + y_i^2 - R^2 < 0$ точка (x_i, y_i) буде розташована всередині кола. Тоді з аналізу знаку величини

$$\Delta_i = (x_i + 1)^2 + (y_i - 1)^2 - R^2 = m_D, . \qquad (1.2.)$$

то як випливає з рис.5 , при $\Delta_i < 0$ ближче до кола можуть бути розташовані

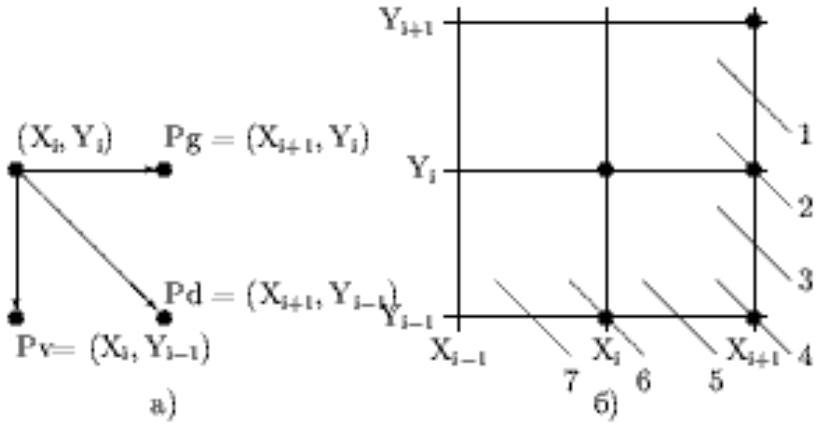


Рис 5.

або горизонтальний, або діагональний пікселі (положення дуги кола 1-3) і діагональний або вертикальний в противному випадку (положення дуги кола 5-7).

Введемо допоміжну величину

$$\delta = \left|(x_i + 1)^2 + y_i^2 - R^2\right| - \left|(x_i + 1)^2 + (y_i - 1)^2 - R^2\right| .$$

Якщо врахувати, що для від'ємного числа a, $|a| = -a$, маємо

$$\delta == 2(\Delta_i + y_i) - 1,$$

і легко бачити , що при $\delta < 0$ горизонтальний піксель знаходиться ближче до кола, ніж вертикальний і підлягає візуалізації. І навпаки, при $\delta > 0$ вибираємо діагональний піксель. Відмітимо, що при вибраному порядку побудови для першого октанту перевірка ситуації, коли $\Delta > 0$ буде зайвою і наступним пік селем може бути тільки горизонтальний або діагональний.

З формули (1.2) випливають прості рекурентні формули для обчислення величини Δ на сусідніх кроках алгоритму. Так при виборі горизонтального пікселя $x_{i+1} = x_i + 1, \quad y_{i+1} = y_i$ отримаємо

$$\Delta_{i+1} = \Delta_i + 2x_{i+1} + 1,$$

а для діагонального $x_{i+1} = x_i + 1, \quad y_{i+1} = y_i - 1$

$$\Delta_{i+1} = \Delta_i + 2(x_{i+1} - y_{i+1} + 1).$$

З цієї ж формули, при $x_1 = 0, \quad y_1 = R$ одержуємо стартове значення для обчислення оцінки

$$\Delta_1 = 2(1 - R).$$

Цей алгоритм є оптимальним як по обсягу обчислень, так як потребує тільки арифметичних дій над цілими числами, так і по якості одержаного зображення.

Контрольні питання

Які вимоги існують до зображення відрізків

Критерій вибору пікселя в алгоритмах Брезенхема

Обмеження класичного формулювання алгоритму Брезенхема для відрізка

Невиконання якої з вимог до зображення відрізку більш всього впливає на якість зображення

Яку частину кола достатньо побудувати за допомогою алгоритму Брезенхема, що отримати зображення всього кола

Геометричний зміст знаків оцінок Δ і δ при растровій розгортці кола

ЛАБОРАТОРНА РОБОТА № 1

Алгоритму Брезенхема розкладу в растр відрізка та кола.

Мета роботи. Оволодіти технікою растрової генерації відрізків і ліній.

Завдання до лабораторної роботи. Написати і відлагодити програму розв'язку однієї з наступних задач(на вибір):

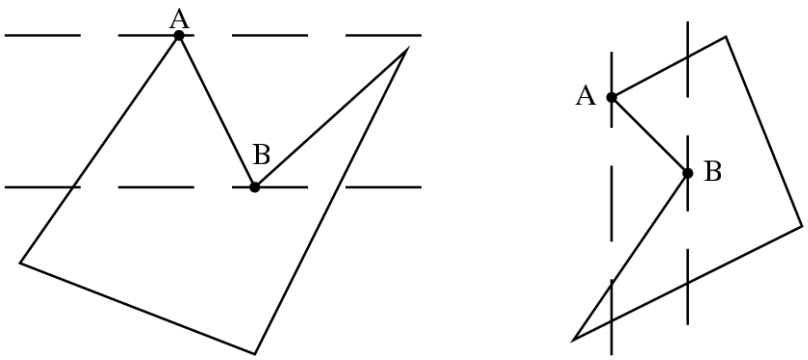
- для відрізка, який визначений парою кінцевих точок, довільно розташованих на площині, реалізувати алгоритм Брезенхема побудови відрізка з довільним кутовим коефіцієнтом, розташованого в будь-якому з чотирьох квадрантів;
- алгоритм Брезенхема побудови кола;

2.4 Заповнення двомірних областей.

Без втрати спільності будемо вважати, що будь-яка двовимірна область може бути представлена у виді багатокутника, і тоді його ребра можна задати в аналітичній формі. Методи рішення цієї задачі поділяється на дві групи: алгоритми сканування і алгоритми запалу. В обох підходах істотним моментом є задача визначення належності пробної точки до даної області.

Найпростішим методом для відповіді на це питання є променевий критерій, суть якого складається в підрахунку кількості точок перетину променя, що сканує з границею області. З пробної точки у довільному напрямку проводиться промінь. Якщо кількість перетинів променя з границею багатокутника з границею є непарне, то точка є внутрішньої, інакше - зовнішньою.

Для реалізації цього методу необхідна наступна класифікація точок перетину прямій з границею багатокутника. Точка називається простою, якщо вона не є вершиною, або ця точки не є точкою локального екстремуму по координаті суміжній з координатою сканування. Якщо точка границі є особою, то перетинання з нею вважається як два, а якщо простою - як одне.



a) b)

Рис.6

Так при скануванні в напрямку осі x точки A і B є особливими, а точка C - простою(рис.6 а). Аналогічна класифікації має місце при скануванні в напрямку y - координати (рис.6,b)

Найпростіший алгоритм заповнення може бути побудований у такий спосіб. Якщо поточний піксель є внутрішньою точкою цієї області, і він не зафарбований, то його необхідно зафарбувати. Природною модифікацією даного підходу є ***алгоритм сканування з прапором.*** Довільно вибирається стартова точка і у залежності від того чи є вона внутрішньою чи зовнішньою, атрибутам пікселя (колір фону або колір області) і значенню прапора привласнюються відповідні значення. Відбувається перехід до наступної точки в напрямку сканування. Якщо ця точка є простою (гранична або вершина), атрибути пікселя і значення прапора змінюються на протилежні.

Зауваження . На перший погляд цей алгоритм в плані програмної реалізації здається майже ідеальним. Але його слабким місцем є вершини, в яких ребра перетинаються під дуже гострим кутом. У них, як правило, алгоритм не встигає" поміняти значень атрибутів і прапора.

Алгоритм ***заповнення по інтервалам*** побудований на основі властивості просторової когерентності геометричних фігур: якщо є множина точок, що належать до одного околу, то з великим ступенем імовірності можна стверджувати, що властивості цих точок будуть однакові. Це дає можливість не перевіряти щораз наступну точку, а розбити промінь, що сканує на інтер-

вали, утворені точками його перетину з ребрами багатокутника. Рівняння прямої, що несе відрізок P_bP_e, заданий початковою P_b і кінцевою точкою P_e має вигляд

$$P = P_b + (P_e - P_b)t \qquad (1.3)$$

При граничних значеннях параметру $t = 0, \mathrm{t} = 1$ з (1.3) маємо кінцеві точки відрізка, а значенням $0 \le \mathrm{t} \le 1$ відповідають точки, що належать відрізку. Якщо рівняння рядка, що сканує $y = y_s$, то з умови перетину його з ребром маємо

$$y_s = y_b + (y_e - y_b)t^*,$$

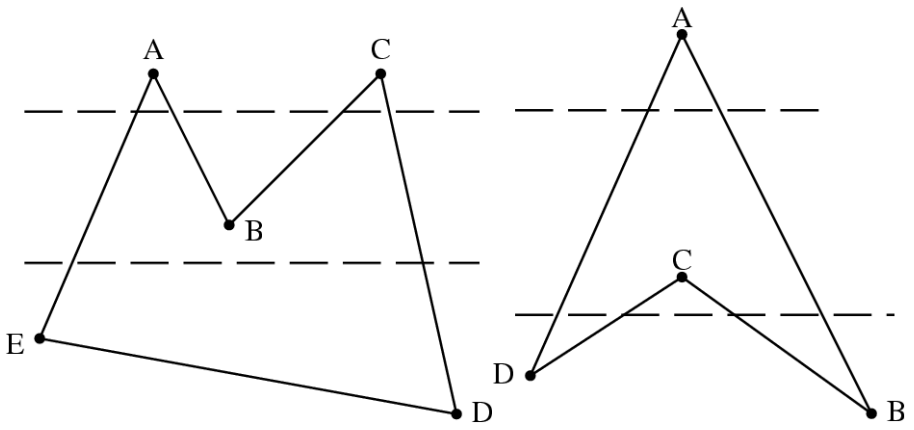
і значення параметра, що відповідає точці перетинання визначиться як

$$t^* = \frac{y_s - y_b}{y_e - y_b}$$

Зауваження. Треба мати на увазі, що співвідношення (1.3) не є рівнянням відрізка(ребра багатокутника), а рівнянням прямої до якої він належить. У загальному випадку рядок, що сканує перетинає всі ребра(окрім паралельних йому), але коректними будуть тільки ті, для яких перетин знаходиться у межах ребра, тобто для яких $t^* \in [0,1]$.

Інтервальний алгоритм зі списком активних ребер дозволяє істотно скоротити обсяг обчислень обумовлений визначення точок перетину рядка, що сканує з ребрами багатокутника і перевіркою коректності перетинань.

Кожний рядок, що сканує може перетнути коректно тільки кілька ребер, що називаються активними. Упорядкуємо ребра багатокутника в порядку зростання y - координат кінців ребер. Якщо рядок, що сканує не проходить через вершину, то порядок проходження видимих і невидимих інтервалів не змінюється.



a) b)

Рис .7

Якщо рядок, що сканує проходить через вершину, і вона є точкою локального мінімуму по y-координаті, то ребра, що сходиться в ній виключаються зі списку. На рис.7.a точка B-локальний мінімум, тому ребра AB і BC повинні бути виключені зі списку. У ситуації показаної на рис. 7.b ребра DC і CB повинні бути включені в список активних ребер, тому що точка C є локальним максимумом.

Алгоритм заповнення за допомогою списку активних ребер не потребує обробки ніяких особливих ситуацій і є оптимальним для задач розглянутого класу.

Іншим підходом до рішення задач заповнення є ***алгоритми типу запалу***. Заповнення запалом застосовується в основному для роботи в інтерактивному режимі і вважається, що робоча область екрану має вже якийсь визначений колір.

В задачі, що розглядається суттєвим є спосіб завдання області. Якщо область завдана своєю границею вона називається гранично визначеною і саме для таких областей призначені алгоритми сканування. Але область може бути задана як двомірний масив, якась частина сусідніх елементів якого має певні атрибути кольору, і колір цих елементів необхідно замінити на інший,

Такі області називаються внутрішньо визначеним і очевидно алгоритми їх перефарбування будуть мати в своїй основі перевірку сусідніх пікселів.

Методи цього напрямку можна розділити на орієнтовані для 4-х зв'язних областей і для 8-ми зв'язних. Область називається 4-х зв’язною, якщо в будь-яку точку цієї області з даного пікселя ми можемо потрапити шляхом руху в наступних 4-х напрямках: праворуч, ліворуч, вверх і вниз. Якщо для цієї мети потрібні ще рухи по діагоналі, то маємо 8-ми зв'язну область.

Таке ділення методів по зв’язності породжує алгоритми, розрізняються тільки в деталях реалізації, тому то для простоти будемо розглядати гранично визначений алгоритм запалу для 4-х зв'язних областей. При цьому умовно будити називати границю, пікселі, що не належать до робочої області.

Найпростіший варіант такого алгоритму зводиться до наступної послідовності дій:

- знайти внутрішній піксель області і помістити його в стек.

- витягти піксель зі стека і зафарбувати його в колір області.

- кожний із сусідніх пікселів, орієнтованих зазначеним образом, якщо вони ще не зафарбовані і не належать до границі, помістити в стек. Алгоритм закінчує роботі, коли стек виявиться порожнім.

Процедура заповнення областей, описана вище, має два недоліки: велика кількість обчислень дублюється і потрібен стек великого обсягу. Однак ці проблеми легко усунути, якщо поміняти місцями операції поміщення пікселів в стек і їх зафарбування.

Контрольні питання

Променевий тест на належність точки області

Звичайні і особливі вершини

Визначення коректного перетину рядка , що сканує з ребром

У якому випадку перетин рядка , що сканує з вершин рахується за два.

Корегування списку активних ребер.

Області застосування методів сканування запалу.

В якому з методів заповнення областей робота обов’язково починається з внутрішньої точки.

ЛАБОРАТОРНА РОБОТА № 2

Заповнення довільних багатокутників

Мета роботи.. Опрацювання методами растрової розгортки двовимірних областей.

Завдання до лабораторної роботи. Провести зафарбування внутрішньої області довільного багатокутника , що заданий списком вершин, за допомогою одного з методів сканування

РОЗДІЛ 3.
ВІДТИНАННЯ.

Ця задача виникає в разі потреби візуалізації частини об’єкта (відрізка або багатокутника) відносно деякого багатокутника, який називається вікном.

Якщо будується зображення об’єкта, що знаходиться всередині вікна, то відсікання називається внутрішнім, у противному випадку - зовнішнім. Звичайно безпосереднє рішення цієї задачі можна одержати як частковий випадок заповнення області. Для цього досить розглянути відрізок, як рядок, що сканує. Але коли необхідно знайти рішення для багатьох відрізків (масовий запит), такий підхід буде далекий від оптимального. Тому для цієї цілі розроблений окремий клас методів.

3.1 Розбиття довільного багатокутника на опуклі складові

Всі існуючі алгоритми рішення цієї задачі, за винятком одного, розраховані тільки для випадок опуклих вікон. Постільки працювати з опуклими фігурами набагато легше ніж з тими, що мають довільну форму, задача розбиття довільного багатокутника на опуклі підобласті має важливе значення.

По визначенню багатокутник є опуклим, коли всі його вершини розташовані по одну сторону від будь якого з його ребер - справа при обході за

напрямком годинникової стрілки або зліва, при обході у протилежному напрямку. Тоді, якщо для якогось ребра (обхід за годинниковою стрілкою) наступна вершина буде розташована зліва від нього, то багатокутник не є опуклим і він розбивається на складові прямою, що містить дане ребро.

Якщо ребро E_i задається вершинами V_i і V_{i+1}, то вершина , V_{i+2} буде розташована справа від E_i, коли векторний добуток

$$V_iV_{i+2} \otimes V_iV_{i+1} = \Pi_i\bar{k}$$

є додатнім. Тобто коли виконується умова $\Pi_i > 0$.У координатній формі для проекції векторного добутку на вісь z маємо

$$\Pi_i = (x_{i+2} - x_i)(y_{i+1} - y_i) - (y_{i+2} - y_i)(\mathrm{x}_{i+1} - x_i)$$

Коли на якомусь кроці обходу маємо ситуацію $\Pi_i < 0$, то багатокутник прямою, що несе це ребро розбивається на дві підобласті. Перший - (опуклий) багатокутник буде утворений вершинами з номерами $i+1, i+2, ...k, q$. Другий - вершинам $q, k+1, ..., 1, 2, ..i$. Тут точка q є точкою перетину продовження поточного ребра з ребром V_kV_{k+1}. Якщо таких точок виявиться більше, ніж одна, то береться точка, з мінімальним значенням параметра t. Після виконання розбиття алгоритм рекурсивно застосовується до отриманих багатокутників.

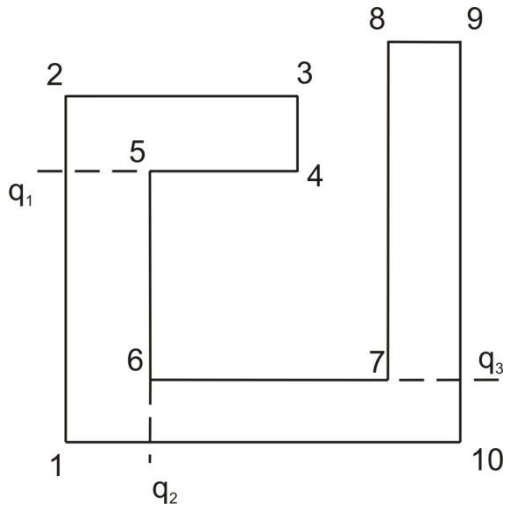


Рис. 8

Слід відмітити, що V_i , V_{i+1} і точка перетину q, знаходяться на одній прямій, і точка V_{i+1} у новому багатокутнику вже не буде вершиною. Так в

ситуації, приведеній на рис. 8, вершини за номерами 5, 6 і 7 після розбиття з розгляду виключаються.

Контрольні питання

Критерій розташування точки відносно направленого відрізка.

Яка перевірка на опуклість багатокутника локальна чи глобальна проводиться в алгоритмі Ключікова – Сороки?

Що є умовою коректності перетину робочої сторони з іншими сторонами багатокутника?

Основна ідея розбиття довільного багатокутника методом обертань.

ЛАБОРАТОРНА РОБОТА № 3

Розбиття довільного багатокутника на опуклі складові.

Мета роботи Розбити багатокутник на опуклі складові.

Завдання до лабораторної роботи

Довільний багатокутник задати у вигляді упорядкованого згідно вибраного правила обходу списку(масиву) вершин;

для кожної сторони багатокутника провести локальну перевірку на опуклість за допомогою правила векторного добутку;

у випадку порушення умови опуклості провести розбиття багатокутника робочою стороною, з контролем коректності перетинів;

застосувати алгоритм рекурсивно до кожної з утворених фігур.

3.2 Одновимірне відтинання

Задача відтинання відрізка прямокутним координатно-орієнтованим вікном є не тільки найбільш розповсюдженою в практиці, але і становить особливий інтерес у плані підходів і ідей, розроблених для її рішення. Корисною виявляється наступна класифікація видимості відрізків, а саме на частково видимі, цілком видимі і цілком невидимі.

Ця класифікація дуже просто проводиться за допомогою 4-х бітового коду видимості, що будується за наступною схемою: значення параметра ві-

дповідно в четвертій, третій, другій і першій позиції коду буде дорівнює одиниці, коли точка знаходиться відповідно ліворуч, праворуч, нижче і вище вікна. В інших випадках величина параметра дорівнює нулеві (рис. 9).

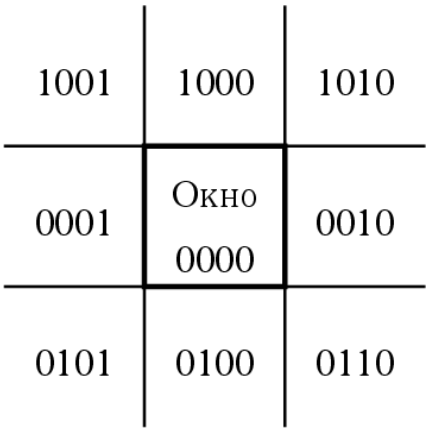


Рис. 9

Цей код дозволяє одержати відповідь на питання чи є відрізок ,заданий своїми кінцевими точками, цілком видимим (у цьому випадку побітова сума кодів його кінцевих точок дорівнює 0) або цілком невидимим (якщо побітовий добуток дорівнює 1).

Найпростішим застосуванням цього коду є метод, заснований на послідовному аналізі коректності перетинання відрізка по черзі з усіма сторонами вікна. При цьому для кожної поточної сторони вікна проводиться аналіз перетину. Так наприклад, якщо ліва, права, нижня, і верхня сторони вікна мають координати x_L, x_R, y_u, y_D, то при поточній лівій стороні вікна аналіз точки перетину $(x_L, \tilde{y})$ аналіз зводиться до перевірки умови

$$y_U \le \tilde{y} \le y_D .$$

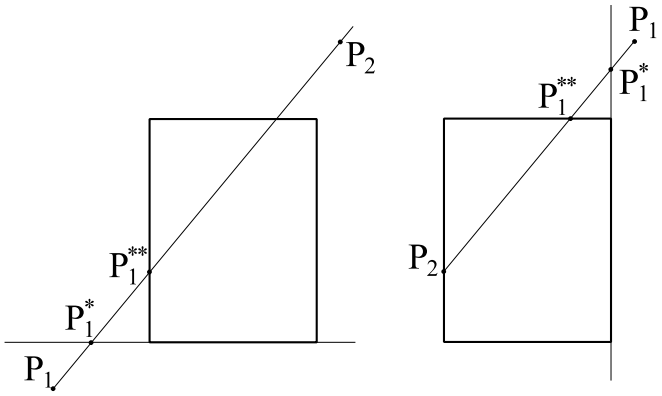
Розглянемо ***алгоритм Сазерленда- Коена,*** який де-факто є стандартом для алгоритмів відсікання і має одну з кращих швидкодій при компактній реалізації. Особливістю цього алгоритму є той факт, що в ньому не проводиться аналіз коректності перетинання відрізка зі сторонами вікна. Формально алгоритм можна записати у виді виконання наступних кроків:

1. Якщо точка P_1 є зовнішньою, то застосовується пункт 3 алгоритму. У противному випадку точки P_1 і P_2 поміняти місцями.

2. Якщо точка P_1 і P_2 виявляться не зовнішніми, то алгоритм закінчує роботу.

3. Знаходиться точка перетину P^* і без аналізу її видимості покладається $P_1 = P^*$ і відбувається перехід до першого кроку алгоритму.

Розглянемо роботу алгоритму на прикладі відрізка з кінцевими точками $P_1(0110)$ і $P_2(1001)$. Значення одиниці в коді виді мості,відповідає стороні з якою відрізок перетинається. Точка $P_1(0110)$ є зовнішньою і рухаючись по коду зліва направо знаходимо перетин відрізка з нижньою стороною вікна $P_1(0110) \Rightarrow P_1^*(0010)$ (рис.10.а). Нова точка залишається зовнішньою і наступний перетин з лівою стороною вікна дає точку $P_1^*(0010) \Rightarrow P_1^{**}(0000)$.



а) б)

Рис 10

Оскільки знайдена точка перетину є видимою, змінимо нумерацію і кінцевими точками будуть $P_1(1001), P_2(0000)$. Повторивши дії попереднього кроку алгоритму маємо (рис.10.б)

$$P_1(1001) \Rightarrow P_1^*(1000) \quad P_1^*(1000) \Rightarrow P_1^{**}(0000),$$

і отримуємо повністю видимий відрізок.

Метод розбиття відрізка середньою точкою орієнтований в основному на апаратне виконання, оскільки в цьому випадку ділення на 2 апаратно еквівалентно зсуву кожного біту вправо. На кожному кроці алгоритму відрі-

зок розбивається середньою точкою m на дві половини. Після першого з ділень, в результаті якого m стає видимою точкою, утворюються дві множини даних ліва і права(або верхня і нижня). Вони містять робочий відрізок і робочу точу. Обробка кожної з множин проводиться окремо. Після кожного розбиття проводиться аналіз видимості одержаних половинок. Повністю невидимий або повністю видимий відрізки відкидаються, а друга половина(частково невидима) розміщається у відповідній множині. Якщо одна з половинок при цьому стає буде повністю видимою, то m стає новою робочою точкою. Так у ситуації показаній на рис. 11 після перших трьох поділів відрізка права множина буде містити тільки одну робочу точку, що одержана при першому поділі (точка 0), а у лівій множині робоча точка послідовно буде приймати положення 0, 1, 2.

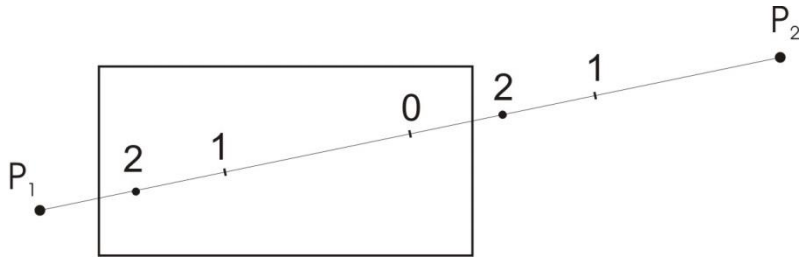


Рис. 11

Алгоритм закінчує свою роботу коли довжина відрізка у кожній з множин буде меша розміру пікселя.

Метод Кіруса- Бека відтинання відрізка довільним опуклим вікном побудований на наступній властивості опуклих багатокутників. Нехай n_i - внутрішня нормаль до i-ої границі багатокутника, а f_i, точка на цій границі, то вектор, утворений як

$$U = P_1 + (P_2 - P_1)t - f_i$$

буде спрямований усередину області при $Un_i > 0$, зовні області при $Un_i < 0$ і збігається з границею і при $Un_i = 0$, рис. 12.

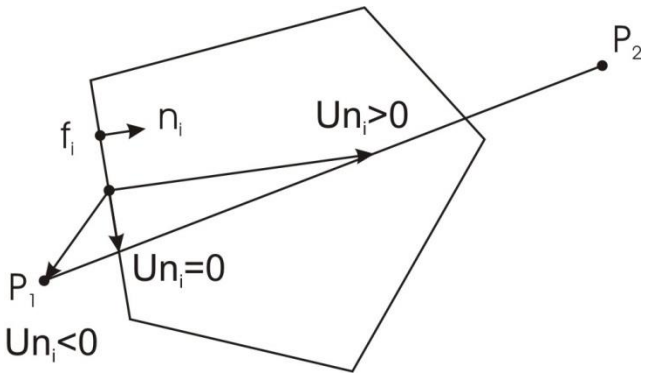


Рис. 12

В останньому випадку кінець вектора належить як границі так і відрізку і точка перетину відрізка з ребром вікна визначиться з умови

$$n_i[P_1 + (P_2 - P_1)t - f_i] = 0$$

Якщо позначити $D = P_2 - P_1$, $W = P_1 - f_i$, то для параметра t маємо

$$t = -\frac{Wn_i}{Dn_i} \qquad (3.1)$$

Задача відтинання формально полягає в визначенні точок коректного перетину відрізка з стороною вікна, Тобто точка перетину повинна одночасно належати і відрізку і стороні вікна. У цьому методі безпосередньо перевіряється тільки одна умова – точка перетину повинна належати стороні вікна. На цьому кроці більшість перетинів відкидаються, а ті що лишились аналізуються спеціальним чином.

Коли відрізок вироджується в точку маємо ситуацію $Dn_i = 0$. У цьому випадку положення точки відносно вікна визначається величиною Wn_i. А саме: точка буде поза, усередині або на границі вікна якщо величина Wn_i буде відповідно менша, більша або рівна нулеві.

Нормалі до границі визначаються з умови рівності нулю скалярного добутку

$$n_x E_x + n_y E_y = 0,$$

яка містить дві невідомі компоненти вектора нормалі. Постільки необхідно знати тільки напрямок вектора, поклавши наприклад $n_y = 1$, маємо

$$n_x = -\frac{E_y}{E_x}.$$

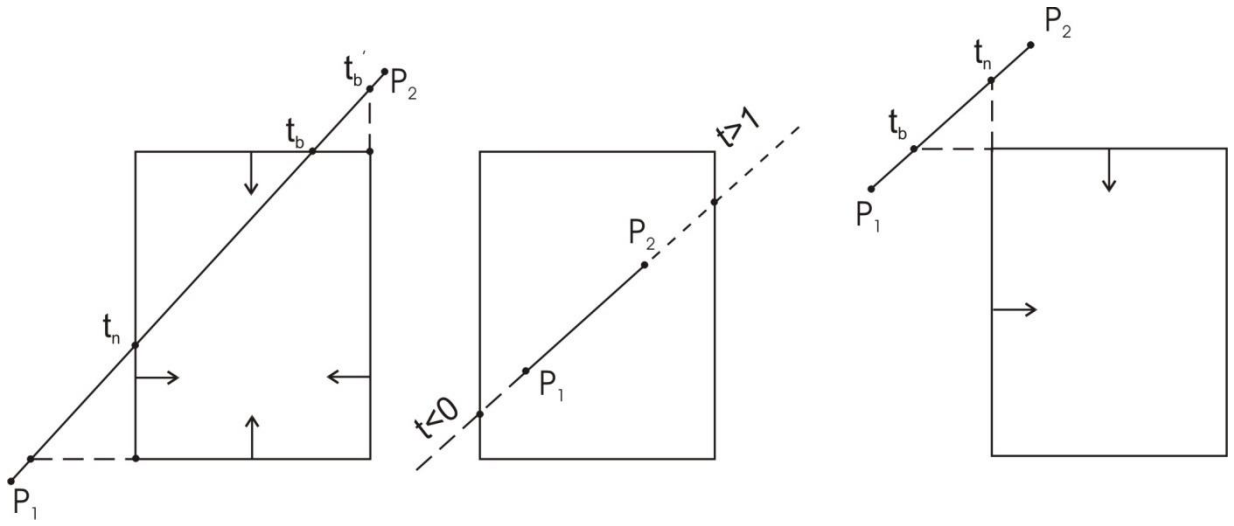
Ця формула непридатна в особих випадках, коли напрямок ребра співпадає з однією з координатних осей. Тоді коли границя вікна паралельна осі $0x$ і $0y$ відповідно маємо наступні вектора нормалі (0,1) і (1,0). Перевірка, чи буде знайдена нормаль внутрішньою, проводиться за допомогою аналізу знаку скалярного добутку $n\,\mathrm{V_i}V_{i+2}$, де як і раніше поточна границя задається вершинами V_i , V_{i+1}. Якщо скалярний добуток буде додатний, то нормаль є внутрішньою. У противному випадку знаки компонент нормалі слід змінити на протилежні.

Тепер сформулюємо послідовність дій реалізації цього метода.

1. Покладається $t_H = 0$ і t_B=1, і якщо ці параметри після закінчення роботи алгоритму не зміняться, те відрізок є цілком видимим ; рис. .13. c

2. Для всіх ребер вікна обчислити параметр t за формулою (3.1). Значення параметра, що не задовольняють умові $0 \le t \le 1$ з розгляду виключаються, рис.13. b

При $Dn_i > 0$ занести t до нижнього списку, при $Dn_i < 0$ – до верхнього, рис. .13, a



a) b) c)

Рис. 13

При закінченні роботи алгоритму після проходу усі сторін багатокутника з нижнього списку вибирається максимальне значення - t_H, а з верхнього мінімальне t_B, які визначають кінцеві точки видимого відрізка.

Випадок $t_H > t_B$ відповідає цілком невидимому відрізкові, рис. 13,с .

3.3 Відтинання багатокутників

Результатом роботи алгоритму відтинання повинен бути один або декілька замкнених багатокутників. При цьому можуть бути додані нові ребра, а ті, що є або збережені, або вилучені. Суттєвим є те , що границі вікна які не обмежують видиму частину, багатокутника, який відсікається, не повинні міститися у результаті. Якщо це не виконується, то можлива некоректне зафарбування границь.

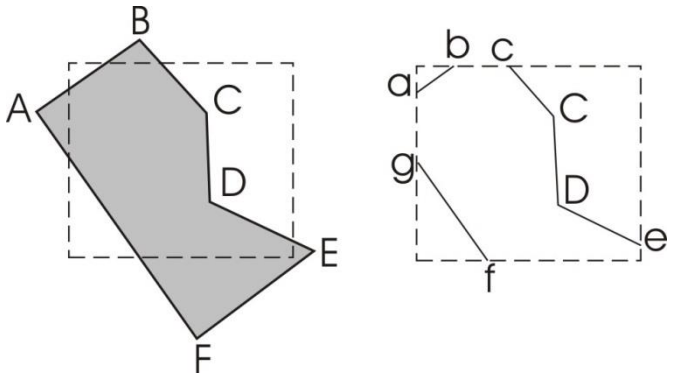


Рис. 14

Безпосереднє застосування для цієї цілі алгоритмів відтинання відрізків може привести до некоректних результатів, коли результатом буде набір не зв’язаних між собою відрізків, рис. 13.

Алгоритм Сазерленда-Ходжмена відтинання довільного багатокутника опуклим вікном полягає в тому , що багатокутник відтинається(далі робочий багатокутник) всіма сторонами вікна по черзі, рис. 14.

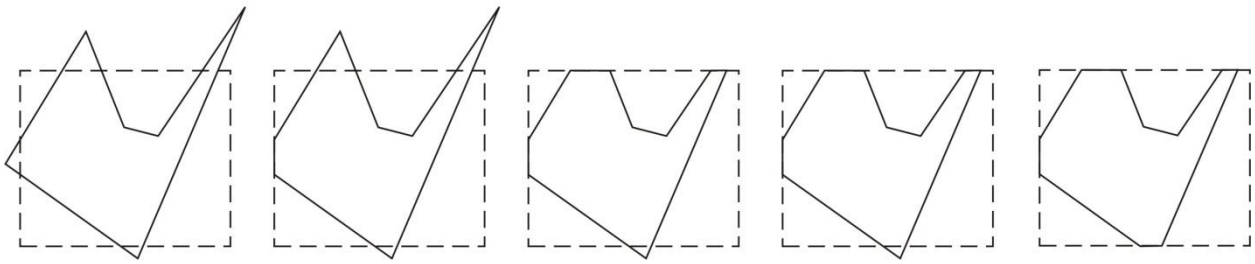


Рис.15

Нехай Ω півплощина, розташована праворуч від сторони вікна P_iP_{i+1} при його обході по годинній стрілці, а V_kV_{k+1} - ребро робочого багатокутника. Очевидно, що можливо всього чотири ситуації орієнтації ребра V_kV_{k+1} відносно області, рис 16:

I вхідне ребро: $V_k \notin \Omega$, а $V_{k+1} \in \Omega$;

II внутрішнє ребро: і V_k і $V_{k+1} \in \Omega$;

III вихідне ребро: $V_k \in \Omega$, а $V_{k+1} \notin \Omega$.

IV зовнішнє ребро: і V_k і V_{k+1} не належать Ω;

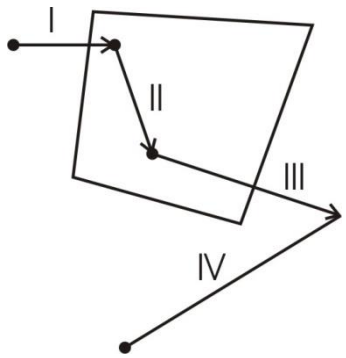


Рис. 16

Випадок, коли ребро збігається зі стороною вікна, аналізується окремо. Для побудови результату формується список робочих вершин багатокутника, що будується в такий спосіб. У випадку, коли вершина стартового ребра $V_k \in \Omega$, то вона заноситься в список, у противному випадку початковий список є порожнім.

У ситуації I список поповнюється точкою перетину ребра V_kV_{k+1} зі стороною вікна P_iP_{i+1} і вершиною V_{k+1}. У випадках II і III список поповнюється тільки одною точкою – вершиною V_{k+1} і точкою перетину, відповідно. У ситуації IV список не змінюється.

Для визначення положення точки V щодо області використовується знак проекції векторного добутку $P_iP_{i+1} \times P_iV = \Pi\bar{k}$ на вісь z.А саме $\Pi > 0$- точка знаходиться ліворуч Ω,. $\Pi < 0$- праворуч, $\Pi = 0$- ребро і сторона вікна співпадають.

Алгоритм Вейлера – Азертона є найбільш загальним із всіх алгоритмів цього класу і дозволяє одержати відсікання довільного багатокутника довільним (не опуклим) вікном.

Нехай кожний з багатокутників заданий списком вершин, причому таким чином, що при русі за списком вершин у порядку їхнього завдання внутрішня область багатокутника знаходиться праворуч від границі. У випадку перетинання границь і багатокутника, що відтинається, і вікна виникають точки двох типів:

вхідні точки, коли орієнтоване ребро робочого багатокутника входить у вікно;

вихідні точки, коли ребро робочого багатокутника виходить з області вікна.

Попередньо необхідно заформувати списки вершин вікна і робочого багатокутника і визначаються всі точки перетину. Ці списки доповнюються новими вершинами - координатами точок перетину. При цьому, якщо точка перетину W_k знаходиться на ребрі, що з'єднує вершини V_i і V_j, то послідовність точок V_i, V_j перетворюється в послідовність V_i, W_k, V_j. Встановлюються двосторонні зв'язки між однойменними точками перетину в списках вершин робочого багатокутника і вікна. Точки торкання і випадок не і випадок , коли ребро одного багатокутника співпадає з частиною другого перетинами не вважаються.

Визначення частини оброблюваного багатокутника, що потрапила у вікно виконується в такий спосіб:

Знаходиться вхідна точка і заноситься у список результатів. Далі рухаємося по вершинах робочого багатокутника поки не зустрінеться наступна точка перетину. Всі пройдені точки заносяться в результат; використовуючи двосторонній зв'язок точок перетину і переходимо на перегляд списку вершин вікна. Рух по вершинах вікна продовжується до наступної точки перетину.

Процедура закінчується , коли зустрінеться точка перетину, з якої стартував алгоритм.

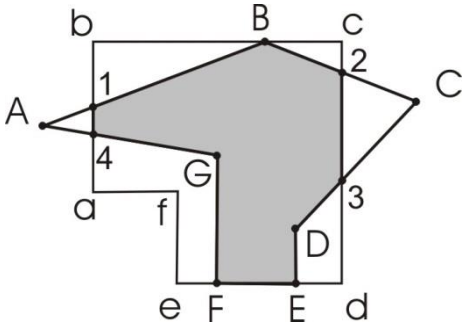


Рис. 17

На рис. 17 вершини робочого багатокутника і вікна позначені відповідно великими і маленькими буквами, а точки перетину їх сторін цифрами. З рисунку видно, що вершина B- точка торкання , і вершини E,F, які утворюють спільне ребро, у список перетинів не входять. Для цієї задачі відповідні списки мають вигляд:

Список робочого багатокутника	A 1 B 2 C 3 D E F G 4 A
Список вікна	a 4 1 b c 2 3 d e f a
Список результат	1 B 2 3 D E F G 4 1.

Контрольні питання

Класифікація взаємного положення відрізка і вікна.

Для яких вікон можна застосовувати код видимості?

В якому з алгоритмів відтинання формується два робочих списка вершин?

В якому з алгоритмів відтинання відсутнє обмеження на опуклість вікна?

Яким чином в алгоритмі Кіруса-Бека забезпечується коректність перетину відрізка зі стороною вікна?

Яким чином в алгоритмі Кіруса-Бека реалізується зображення повністю видимого відрізка?

ЛАБОРАТОРНА РОБОТА № 4

Відтинання відрізка довільним вікном.

Мета роботи. Застосувати алгоритм Кіруса-Бека, який розрахований на опуклe вікно, на випадок відтинання довільним вікном.

Завдання до лабораторної роботи

Використовуючи результати лабораторної роботи № 3 представити довільне вікно у вигляді об’єднання опуклих складових і для кожної з них застосувати алгоритм відтинання;

визначити нормалі сторін робочих багатокутників і перевірити чи є вони внутрішніми;

заповнити внутрішній і зовнішній списки початковими значеннями;

після проходу всіх сторін вікна визначити параметри які відповідають видимій частині відрізка;

об'єднати результати.

4. ЕЛЕМЕНТИ КООРДИНАТНОЇ ГЕОМЕТРІЇ.

4.1 Однорідні координати

Згідно своїм властивостям перетворення координат об’єктів поділяються на наступні дві групи

1. Афінні перетворення, до яких належать паралельний перенiс, повороти, перетворення розтягання - стиску і відображення. При цих перетвореннях величина кутів є інваріантною.

2. Проективні перетворення, у яких: величини кути уже не є інваріантами.

Усі перетворення координат у комп'ютерній графіці проводяться у формі матричного добутку, оскільки в цьому випадку досягається максимально високий ступінь формалізації , так як результат обчислення матричного добутку $C = A \times B$ зводиться до виконання простої операції

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} .$$

При цьому застосовується спеціальний математичний механізм, пов'язаний з визначенням положення точки у просторі - однорідні координати. Звичайного апарату декартових координат , не достатньо для вирішення деяких важливих завдань. Введення однорідних координат дозволяє:

- описувати нескінченно віддалені точки, які неможливо описати, використовуючи евклідові координати;
- уніфікувати матричну запис афінних перетворень;
- будувати за допомогою апарата однорідних координат проективні перетворення.

Однорідні координати точки на площині і у просторі задаються відповідно як вектори (wx, wy, w) і (wx, wy, wz, w). Для того щоб визначити геометричний зміст параметра w розглянемо наступну задачу. Знайдемо проекцію точки $P(x, y, z)$ на площину π у напрямку вектора $\bar{a}(a_x, a_y, a_z)$. Слід зазначити, що подібні задачі виникають при вилучення невидимих площин, в моделях освітленнях,а також при побудові спеціальних ефектів (прозорість,тіні, тощо).

Площина задається рівнянням

$$\bar{n}\bar{r} - d = 0 \qquad (4.1)$$

Тут $\bar{n}(n_x, n_{y,} n_z)$- нормований вектор нормалі $n_x^2 + n_y^2 + n_z^2 = 1$; $\bar{r} = \bar{r}(x, y, z)$- радіус-вектор точки; d - довжина перпендикуляру, опущеного з початку координат на площину.

Проекція знаходиться як точка перетину площини π з прямою $\bar{r}_P = P + \bar{a}t$

Значення параметру t, що відповідає точці перетину можна знайти з рівняння площини ( ), тоді маємо

$$t = \frac{f}{\bar{n}\bar{a}},\ f = d - \bar{n}\bar{r}_p \qquad (4.2)$$

а координати проекції P_π визначаються як

$$x_\pi = x + a_x \frac{f}{\bar{n}\bar{a}},\ y_\pi = y + a_y \frac{f}{\bar{n}\bar{a}},\ z_\pi = z + a_z \frac{f}{\bar{n}\bar{a}} \quad (4.3)$$

З отриманих співвідношень випливають відповідні формули як для паралельних так і перспективних проекцій.

У випадку проекції у напрямку радіус-вектора точки $\bar{a}(-x,-y,-z)$ на площину $z = d, \bar{n}(0,0,1)$ знаходимо

$$t = -\frac{d - z}{z} \qquad (4.4)$$

а сама точка перетину є $P_\pi(x\frac{d}{z}, y\frac{d}{z}, d)$

Знайдемо проекції точки (x, y, w) в евклідовому просторі XYW на площину $w = 1$. Якщо в (4.4) покласти $d = 1$ і $z = w$ то для параметра t маємо $t = \frac{w-1}{w}$, а шукана точка має координати $(x/w, y/w, 1)$. Таким чином, перетворення з однорідних координат в евклідові еквівалентно проекції точки на площину $w = 1$ уздовж лінії, що з'єднує точку з початком координат. Тобто перетворення з однорідних координат в евклідові є однозначним. Обернена ж задача має безліч рішень, тому що всі точки на лінії, що з'єднує точку (x, y, w) і початок координат будуть проектуватися в точку $(x/w, y/w)$.

Виразимо w через параметр t, який є відстанню між точкою і її проекцією у вибраному напрямку - $w = \frac{1}{1-t}$. Тоді з останньої формули випливає, що при

прямуванні $t \to \infty$ параметр $w \to 0$ і точка $(x, y, 0)$ є зображенням нескінченно віддаленої точки в однорідних координатах.

Походження терміну однорідні координати можна пояснити на наступному прикладі. Розглянемо неоднорідне рівняння $Ax + By + C = 0$. Заміна x і y на x/w і y/w дає $A(x/w) + B(y/w) + C = 0$. Помноживши на w, отримуємо однорідне рівняння $Ax + By + Cw = 0$

4.2 Матриці основних перетворень.

Повороти відносно координатних осей z, x, y на кут α, β, γ задаються за допомогою матриць поворотів R_z, R_x, R_y відповідно

$$R_z = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4.5)$$

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta & 0 \\ 0 & \sin\beta & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4.6)$$

$$R_y = \begin{bmatrix} \cos\gamma & 0 & \sin\gamma & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\gamma & 0 & \cos\gamma & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Матриця паралельного переносу початку координат у точку (a, b, c) і матриця відображення щодо осі ox відповідно мають вигляд

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -a & -b & -c & 1 \end{bmatrix} \qquad (4.7)$$

$$M_x = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4.8)$$

4.2 Визначення видових координат об'єкта.

Природним способом завдання положення і геометричних характеристик тіл, що складають сцену, є їхнє представлення в системі координат, що зв'язані із самими тілами. Ці координати називаються світовими, і звичайно їхній початок міститься в центрі ваги тіла. Таке представлення не є достатнім, тому що зображення тіла залежить і від положення спостерігача і виникає необхідність у системі координат, зв'язаної з точкою спостереження, видових координатах.

Нехай x_w і x_v - відповідно світові і видові координати. Тоді координати об'єкта у видових координатах можна одержати шляхом множення вектора на матрицю видових перетворень

$$x_v = x_w V \,,\; x_{vi} = \sum_{k=1}^{4} x_{vk} V_{ki} \,.$$

а перспективні перетворення виконуються окремо.

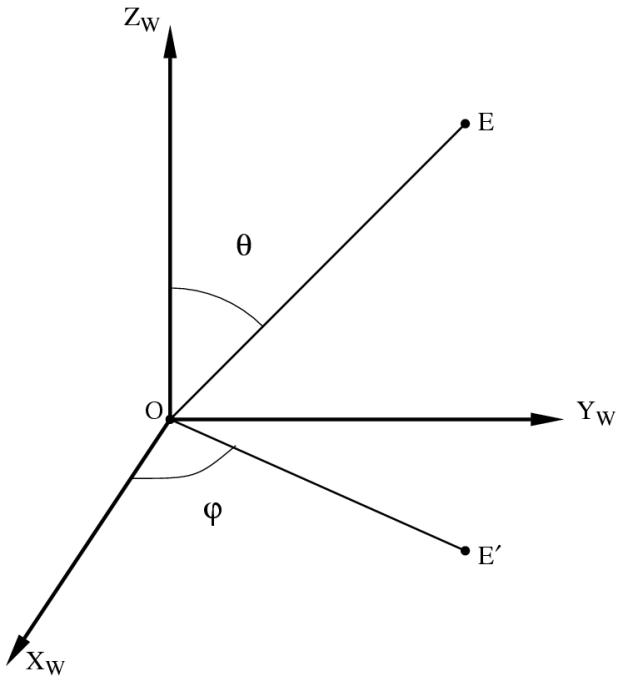


Рис. 18

Точку спостереження E прийнято задавати в сферичних координатах, рис 17.

$$\begin{cases} x = \rho\cos\varphi\sin\theta \\ y = \rho\sin\varphi\sin\theta \\ z = \rho\cos\theta \end{cases}$$

Для того щоб задати ти систему координат необхідно визначити її початок і напрям осей. При напрямку погляду з в додатня напіввісь $0x_v$ направлена вправо, а додатня напіввісь $0y_v$-вверх. Напрям осі $0z_v$ вибирається таким чином, що значення координати збільшується по мірі віддалення від точки спостереження. Таке визначення осей є логічним і зручним, але отримана система координат є лівосторонньою.

Перехід до видових координат складається з декількох етапів. Початок видової системи координат розташовується в точці спостереження $a = x_E, b = y_E, c = -z_E$, що здійснюється за допомогою матриці переносу (4.3). Потім поворотом по годинниковій стрілці системи координат відносно осі z на кут $\alpha = \pi/2 - \varphi$ суміщаємо вісь y з горизонтальною проекцією відрізка OE', що робиться шляхом множення вектора координат на матрицю повороту (4.5). Після цього систему координат треба повернути відносно осі ox

на кут $\beta = \pi - \theta$ за допомогою перетворення (4.6), щоб вісь oz_v збіглася з віссю спостереження. Останній етап складається у відображення відносно осі ox за допомогою матриці (4.8).

Таким чином, усі приведені перетворення складаються в послідовному обчисленні матричного добутку $V = T \times R_z \times R_x \times M_x$.

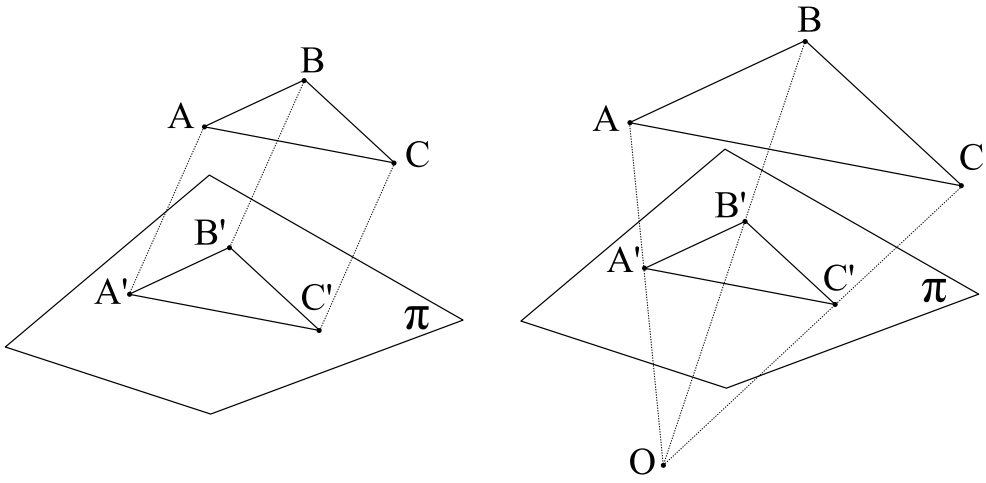
Підсумкова матриця має вигляд

$$V = \begin{bmatrix} -\sin\varphi & -\cos\theta\cos\varphi & -\sin\theta\cos\varphi & 0 \\ \cos\varphi & -\cos\theta\sin\varphi & -\sin\theta\sin\varphi & 0 \\ 0 & \sin\theta & -\cos\theta & 0 \\ 0 & 0 & \rho & 1 \end{bmatrix}$$

4.3 Проективні перетворення.

Після того як визначені видові координати об'єктів сцени, необхідно зробити перехід від тривимірного завдання тіла до двовимірних екранних координат. Цей перехід здійснюється за допомогою проективних перетворень і при цьому величини кутів уже не будуть інваріантами. Але деякі властивості геометричних об’єктів при проективних перетвореннях є інваріантами. Так прямі залишаються прямими, а конічні перетини переходять в криві цього ж класу. Наприклад при проективних перетворення коло може бути перетворено в еліпс або параболу.

В машинній графіці використовуються в основному два види проекцій паралельна (циліндрична)і центральна(перспективна). Для проведення проекційних перетворень необхідно задати положення проективної площини (картинної площини, екрану) і напрямок проекції. В циліндричній проекції проектування проводиться за допомогою пучка паралельних прямих (рис 19.а).



а) б)

Рис. 19

В центральній проекції напрямком проекції є промінь, що з'єднує точку з центром проекції (рис. 19.б). Кожний з цих видів розбивається на декілька підвидів в залежності від взаємного розташування картинної площини і осей координат. Проекцію точки на площину в обох випадках можна безпосередньо отримати з формул (4.2), (4.3)

Звичайно пучок прямих в циліндричних проекціях є ортогональним π. Якщо це не так, то відповідна проекція називається косо кутньою. В залежності від розташування екрана, тобто вектора нормалі $\bar{n}_\pi(n_x, n_y, n_z)$ картинної площини циліндричні проекції в свою чергу діляться на:

ортографічну- площина π є однією з координатних площин;

ізометрія – нормаль до π складає рівні кути з координатними осями;

диметрія – два з цих кутів рівні між собою:

триметрія – всі кути різні.

Осанні три види звичайно об’єднують одним поняттям – аксонометрія.

У випадку ортогональної проекції компоненти вектора нормалі площини і вектора напрямку проекції задовольняють умові $\bar{a} \otimes \bar{n} = 0$, що еквівалентно співвідношенням

$a_x = \lambda n_x,\ a_y = \lambda n_y,\ a_z = \lambda n_z.$

Тоді для нормованого вектора нормалі $\bar{a}\bar{n} = \lambda$ і з (4.2) маємо $t = \frac{f}{\lambda}$. А формула (4.3) набуває вигляду

$$x_\pi = x_p + n_x f\ ,\ \ y_\pi = y_p + n_y f\ ,\ \ z_\pi = z_p + n_z f$$

Для ортографічних проекцій екран співпадає з однією з координатних площин. Так у випадку площини $z = d$, для якої $\bar{n}(0,0,1)$, маємо

$x_\pi = x\ ,\ \ y_\pi = y\ .$

Ортографічна проекція є дуже поширеною в комп'ютерній графіці, тому що просторові координати співпадають з екранними, а саме визначення екранних координат не потребує ніяких обчислень.

Для центру проекції в точці $C(x_c, y_c, z_c)$ напрямок проектування визначається як $\bar{a}(x_c - x, y_c - y, z_c - z)$. Найбільш поширеним випадком є центр проекції розташований на осі z $C(0,0,c)$. Якщо позначити d - відстань між

спостерігачем і площиною екрану, то з формул (4.2) і (4.3) маємо

$$\left(x\frac{c-d}{c-z}, y\frac{c-d}{c-z}, d, 1 \right) \qquad (4.9)$$

У випадку коли проективна площина розташована між об’єктом і центром проекції $d < z$, отримаємо зменшене зображення. Якщо об’єкт розмістити між центром проекції і екраном $d > z$, то зображення буде збільшеним.

Координати проекції ( ) можна отримати за допомогою матричних перетворювань. У цьому випадку відповідна матриця має вигляд

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{d}{c-d} & -\frac{1}{c-d} \\ 0 & 0 & \frac{dc}{c-d} & \frac{c}{c-d} \end{bmatrix}$$

В результаті множення вектора координат $(x, y, z, 1)$ на матрицю Q маємо

$$\left(x, y, \frac{d(c-z)}{c-d}, \frac{c-z}{c-d} \right).$$

Це точка з однорідними координатами, які ще потрібно перетворити в декартові шляхом ділення на четверту компоненту , що приводить до результату (4.9). На практиці досить зручно помістити центр проекції в точку спостереження $C(0,0,0)$ і тоді маємо прості робочі формули

$$x_\pi = x\frac{d}{z},\ y_\pi = y\frac{d}{z}. \qquad (4.5)$$

Контрольні питання.

Що таке однорідні координати

Що таке видові координати

Яке з перетворень координат в матричній формі можна виконати тільки в однорідних координатах

Від чого залежить зображення в системі видових координат

Для якого з перетворень координат величини кутів не є інваріантами

Для якого виду проективних перетворень величини кутів не змінюються

Яким чином відбувається перехід від тривимірних видових координат до системи координат екранної площини

ЛАБОРАТОРНА РОБОТА № 5

Побудова зображення багатогранника в системі видових координат.

Мета роботи. Оволодіння технікою видових і проективних перетворень при побудові зображень просторових об’єктів.

Завдання до лабораторної роботи Задати багатогранник (куб або тетраедр) координатами вершин з врахуванням належності до грані в системі світових координат. Задати вектор спостереження його початковою точкою в сферичних координатах $E(\rho,\varphi,\theta)$.

Вказівки до виконання лабораторної роботи

За допомогою матриці видових перетворень перевести координати вершин у систему координат, зв'язану з точкою спостереження.

За допомогою проективних перетворень перейти від видових координат до системи координат екранної площини.

Зробити допоміжну панель для відображення значень координат точки спостереження, Змінюючи ці значення можна робити висновок про правильність роботи програми.

РОЗДІЛ 5
ВИДАЛЕННЯ НЕВИДИМИХ ЛІНІЙ І ПОВЕРХОНЬ.

5.1 Класифікація методів видалення невидимих поверхонь

Складність задачі видалення невидимих ліній і поверхонь привела до появи великого числа різних методів і алгоритмів її рішення, але загального найкращого рішення поставленої задачі не існує. Головним недоліком всіх алгоритмів є значний обсяг обчислень, необхідних для визначення ліній, що видаляються, і поверхонь.

Алгоритми видалення невидимих ліній і поверхонь класифікуються по способу вибору систем координат або простору, у якому вони працюють. Перший клас - це алгоритми, що працюють в об'єктному просторі, і зв’язані з фізичною системою координат (світові координати), у якій вони описані.
У цьому випадку кожна з N граней об'єкта порівнюється з іншими N-1 гранями і обсяг обчислень зростає як $N^{2.}$

Другий клас алгоритмів працює в просторі зображення і зв’язаний із системою координат того пристрою, на якому ці об'єкти синтезуються.
Тут для кожного пікселя зображення визначається яка з N граней об'єкта видима (при спроможності і екрана M×M обсяг обчислень зростає як $M^2 \times N$).

Крім цього існує велике число змішаних методів, що поєднують обидва підходи .

Алгоритми першого класу використовуються в тих випадках, коли потрібна висока точність зображення об'єктів. Синтезовані в цьому випадку зображення можна вільно збільшувати (зменшувати) у багато разів, переміщати або повертати. Точність обчислень алгоритмів другого класу обмежується дозволяючи спроможністю екрана. Результати, одержані в просторі зображення, а потім збільшені (зменшені) у багато разів, не будуть відповідати вихідній сцені.

Слід відмітити, що разом з наведеними класифікаціями рішення цих задач, існують інші класифікації, в яких розділення на групи проводиться по порядку обробки елементів(сцени: видалення в довільному порядку і у порядку, обумовленому процесом візуалізації) або по об’єктах,, що видаляються (лінії, ребра, поверхні, об’єми).

На практиці, порівняльний аналіз існуючих алгоритмів видалення невидимих ліній украй обмежений. У різних випадках при роботі з різними моделями синтезованого простору ефективні різні алгоритми. Навіть при роботі з однією і тією ж моделлю виявляється, що в залежності від точки спостереження варто використовувати різні алгоритми.

Найбільше часто використовуються алгоритми Робертса, Аппеля, Варнока, Вейлера-Азертона, алгоритм, що використовує список пріоритетів (упорядкувань), метод Z-буфера, метод порядкового сканування.

Для реалізації алгоритмів видалення невидимих ліній часто використовуються алгоритми нижнього рівня - відтинання відрізка (Сазерленда - Коена) і алгоритм належності точки багатокутникові.

Після переходу від світових координат до системи координат спостерігача(видових) необхідно визначити які частини об’єкту є невидимими. Цей етап побудови зображення є ні більш трудомістким.

На початку реалізації будь-якого алгоритму видалення невидимих ліній і поверхонь для підвищення ефективності його роботи звичайно проводиться сортування координат об'єктів синтезованої сцени. Основна ідея сортування полягає в тому, що, чим далі розташований об'єкт від точки спостереження,

тим більше імовірність того, що він буде цілком або частково екрануватися одним з об'єктів, більш близьких до спостерігача.

5.2 Метод Робертса

Коли сцена складається з одного опуклого багатогранника ідеальним для цієї задачі є ***метод Робертса*** (метод не лицьових площин), який працює в просторі об’єктів. У цьому методі виключаються грані тіла, які екрануються від спостерігача іншими гранями цього ж тіла.

Позначимо через $\bar{e}$ - вектор спостереження, а через $\bar{n}_i$ - вектор внутрішньої нормалі до i - ої грані тіла. Тоді, як витікає з рис 20, критерій видимості цієї грані матиме вид

$$\bar{e}\bar{n}_i \geq 0$$

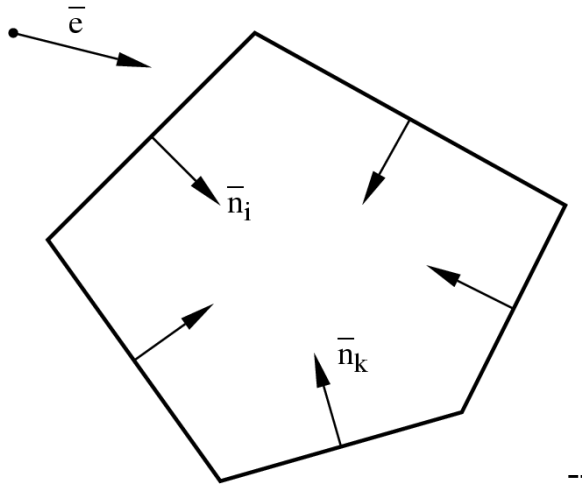


Рис. 20

Метод потребує наступних попередніх дій. Необхідно визначити рівняння площини, що несе грань

$$A_i x + B_i y + C_i z + D_i = 0. \qquad (5.1)$$

Тоді для направляючих косинусів нормалі до неї маємо $n_{xi} = A_i$, $n_{yi} = B_i$, $n_{zi} = C_i$. Для того, щоб визначити чи є дана нормаль внутрішньої для багатокутника досить перевірити умову

$$A\tilde{x} + B\tilde{y} + C\tilde{z} + D > 0,$$

де $(\tilde{x}, \tilde{y}, \tilde{z})$ є внутрішньою точкою тіла. У випадку не виконання цієї умови слід поміняти знаки компонентів нормалі на протилежні. Постільки багатокутник є випуклим, то внутрішня точка може бути визначена, як середнє арифметичне відповідних координат його вершин.

Запишемо рівняння площини, що проходить через три задані точки у вигляді змішаного добутку

$$\begin{vmatrix} x - x_1 & y - y_1 & z - z_1 \\ x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \end{vmatrix} = 0$$

Тоді коефіцієнт в рівнянні площини обчислюються як:

$$A = \begin{vmatrix} y_2 - y_1 & z_2 - z_1 \\ y_3 - y_1 & z_3 - z_1 \end{vmatrix}, \qquad B = -\begin{vmatrix} x_2 - x_1 & z_2 - z_1 \\ x_3 - x_1 & z_3 - z_1 \end{vmatrix},$$

$$C = \begin{vmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{vmatrix}, \qquad D = Ax_1 - By_1 + Cz_1 \qquad (5.2)$$

У тому випадку, коли використовується нормальна форма рівняння площини ( ), проводиться стандартна операція нормування

$$n_x = \frac{A}{S},\ n_y = \frac{B}{S},\ n_z = \frac{C}{S},\ d = -\frac{D}{S}, S = \sqrt{A^2 + B^2 + C^2} \quad (5.3)$$

Хоча метод Робертса розрахований для застосування до сцен, що складаються з тільки з одного опуклого багатокутника, його застосування в якості попередньої обробки для більш загальних сцен дає досить суттєвий ефект економії обчислень.

5.3 Метод Z-буфера і його варіанти

Метод Z-буфера є самим потужнім для рішення задач цього класу і застосовується як до полігональних, так і до криволінійних поверхонь і працює в просторі зображень. Як правило застосовується для зображення складних сцен з застосуванням ортографічних проекцій на екран. Часові характеристики метода лінійно залежать від складності сцени по глибині, тобто середньої кількості граней, які взаємно перекриваються . Він досить просто реалізуєть-

ся, але потребує значних ресурсів як пам’яті, так і обчислень. Це обумовлено тим, що на відміну від методу Робертса метод Z-буфера відпрацьовує кожний піксель окремо і стільки разів, скільки у цій точці перекривається багатокутників сцени. Методи такого типу називаються методами грубої сили.

Алгоритм методу має наступний запис:

1. Всі багатокутники сцени перетворити в растрову форму.

2. Створити два списки (буфера): перший містить глибину піксель, другий - атрибути піксель. Буфер атрибутів заповнити атрибутами тіла, а в буфер глибини помістити максимально можливе її значення.

3. Для кожного багатокутника сцени у довільному порядку виконати наступну операцію:

Обчислити глибину багатокутника Z,

якщо $Z < Z_b$, то $Z_b = Z$, а у відповідному буфері атрибутів замінити атрибути поточного пікселя на атрибути багатокутника;

інакше-перейти до наступного пікселю.

Класичне формулювання алгоритму можна доповнити попереднім виконанням методу нелицьових площин і розглядати прямокутну оболонку, що охоплює тіла сцени, що набагато скорочує як обсяг необхідної пам'яті так і обсяг обчислень.

Основним обчислювальним моментом метода є визначення глибини пікселя для даної грані багатокутника. У випадку центральної проекції , зробивши зворотній перехід від екранних координат до видових за допомогою формул (4.5), з рівняння площини (5.1) маємо

$$Z_{ij} = -\frac{dD}{Ax_j^e + By_i^e + dC},$$

де i - номер рядка по y - координаті, j - номер абсциси у цьому рядку. Одержана формула містить операцію ділення, і як наслідок потребує суттєвих затрат часу. Тому метод Z-буфера, як правило застосовується для ортогональ-

ної проекції. У цьому випадку процедура обчислення глибини суттєво спрощується , тому що екранні координати співпадають з видовими. Постільки y y - координата в межах рядка є постійною з (5.1) маємо

$$z_j = -\frac{Ax_j + By + D}{C}$$

У межах рядка, що сканує $y = const$, $x_{j+1} = x_j + 1$ і остаточно приходимо до простої рекурентної формули

$$z_{j+1} = z_j + \Delta z\ ,\ \Delta z = -\frac{A}{C}.$$

Метод сканування по рядкам звичайно розглядають, як одну з модифікацій Z-буфера(теж для випадку ортогональної проекції) , розроблену з метою економії пам'яті, але його ефективність в плані швидкої набагато вище. ніж оригіналу. У цьому методі буфер має розмір тільки одного рядка, що сканує і в межах рядка грань тіла стає відрізком

$$ax + bz + c = 0.$$

Будь який рядок можна розбити на інтервали таким чином , що буде мати місце одна з ситуацій, показаних на рис. 21

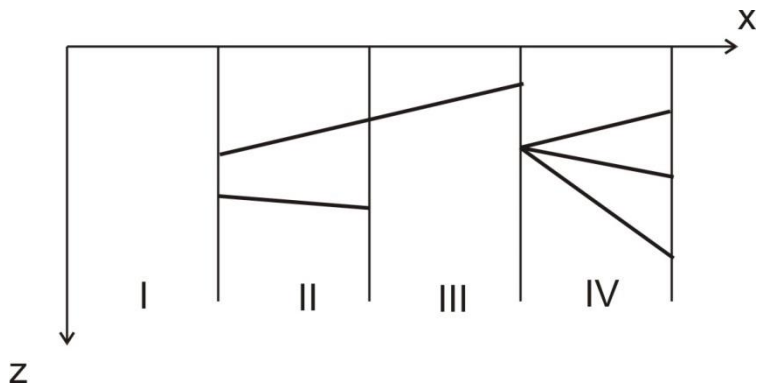


Рис. 21

I. Інтервал є пустим.

II. В інтервалі знаходяться два або більше відрізка, Зображується той, що має меншу глибину.

III. Інтервал містить один відрізок, що підлягає візуалізацію.

IV. Інтервал містить декілька відрізків, які мають спільну точку. Також зображується відрізок, що має меншу глибину.

В останньому випадку оцінка глибини проводиться у середній точці інтервалу. Коли у рядку є два відрізки з номерами k і m, які перетинаються, то після розбиття кожного з них на два точкою перетину, приходимо до ситуації IV. Класифікацію інтервалів досить зручно проводити за допомогою знаку пробної функції

$$(z_k^l - z_m^l)(z_k^r - z_m^r).$$

Де нижні індекси – номери відрізків, а верхні індекси вказують є точка лівою чи правою для відрізка. Коли значення функції більше нуля, рівне нулю і менше нуля маємо відповідну ситуацію: відрізки не перетинаються, є спільна точка і відрізки перетинаються.

Таким чином проводиться оцінка видимості всього відрізка в цілому і для цього достатньо тільки встановити пріоритети відрізків по глибині. Постільки оцінка глибин є найбільш трудомісткою, то дуже корисним для роботи алгоритму є наступне правило Ромні. Якщо відрізки не перетинаються і проекції кінців відрізків на рядок, що сканує, слідують в тому же порядку, що і для попереднього рядка, то пріоритет по глибині не змінюється.

3.3 Алгоритм Варнака

Цей алгоритм працює в просторі зображення й аналізує область на екрані дисплея (вікно) на наявність у них видимих елементів. Якщо у вікні немає зображення, то воно просто зафарбовується кольором фону. Якщо ж у вікні є елемент, то перевіряється чи досить він простий для візуалізації. Якщо об'єкт складний, то вікно розбивається на більш дрібні, для кожного з яких виконується тест на відсутність і або простоту зображення. Рекурсивний процес розбивки може продовжуватися доти поки не буде досягнута межа дозволу екрана.

Можна виділити 4 випадки взаємного розташування вікна і багатокутника: багатокутник цілком поза вікном, багатокутник цілком усередині вікна,

багатокутник перетинає вікно, багатокутник охоплює вікно (рис.22випадки a,b,c,d відповідно).

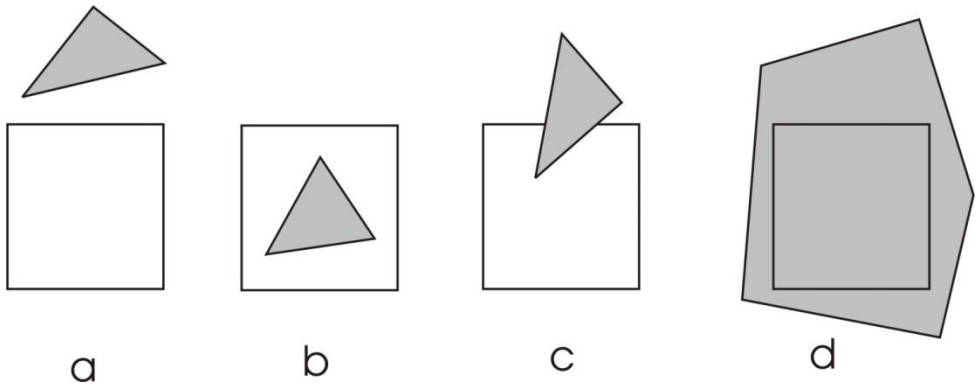


Рис.22

У чотирьох випадках можна відразу прийняти рішення про правила зафарбування області екрана:

всі багатокутники сцени - зовнішні відносно вікна. У цьому випадку вікно зафарбовується фоном;

є тільки один багатокутник, який перетинає вікно або є внутрішнім. У цьому випадку маємо задачу відсікання багатокутника вікном, в результаті рішення якої відповідні частини вікна матимуть атрибути фону і багатокутника

є тільки один багатокутник, що охоплює вікно . У цьому випадку вікно зафарбовується його кольором.

є кілька декілька багатокутників і хоча б один з них охоплює вікно. Якщо при цьому багатокутник, що охоплює, розташований ближче інших до спостерігача, то вікно зафарбовується його кольором.

У будь-яких інших випадках процес розбивки вікна продовжується. Легко бачити, що при растрі 1024×1024 і діленню сторони вікна навпіл потрібно не більш 10 розбивок. Якщо досягнуто максимальну розбивку, але не виявлено жодного з наведених вище чотирьох випадків, то для точки з центром в отриманому мінімальному вікні (розміром з піксель) обчислюють-

ся глибини багатокутників, що залишилися, і зафарбування визначає багатокутник, найближчий до спостерігача.

Перші три випадки ідентифікуються легко. Останній же випадок фактично зводиться до пошуку багатокутника, який охоплює і перекриває всі інші багатокутники, пов’язані з вікном. Перевірка на такий багатокутник може бути виконана в такий спосіб: у кутових точках вікна обчислюються Z-координати для всіх багатокутників, зв'язаних з вікном. Якщо всі чотири такі Z-координати багатокутника, що охоплює, ближче до спостерігача, ніж всі інші, то вікно зафарбовується кольором відповідного багатокутника, що охоплює. Якщо ж ні, то ми маємо складний випадок і розбивку варто продовжити.

Очевидно, що важливою частиною алгоритму є визначення розташування багатокутника відносно вікна.

Перевірка на те що багатокутник зовнішнього або внутрішній відносно вікна для випадку прямокутних вікон легко реалізується використанням прямокутної оболонки багатокутника і порівнянням координат(габаритні тести). Для внутрішнього багатокутника повинні одночасно виконуватися умови:

$$x_{\min} \geq w_l, \quad x_{\max} \leq w_r, \quad y_{\min} \geq w_d, \quad y_{\max} \leq w_u$$

Тут $x_{\min}, x_{\max}, y_{\min}, y_{\max}$ - габарити багатокутника, а w_l, w_r, w_d, w_u - відповідні координати лівої, правої, нижньої та верхньої сторін вікна. Для зовнішнього багатокутника достатнє виконання кожної з наступних умов:

$$x_{\min} > w_r \qquad x_{\max} < w_l \qquad y_{\min} > w_u \qquad y_{\max} < w_d$$

На жаль тут існують особливі випадки і визначений таким чином багатокутник, що охоплює кут вікна не буде ідентифікований як зовнішній, рис. 23.

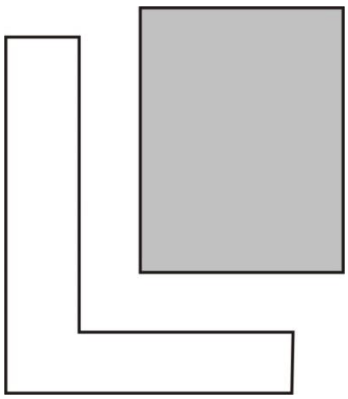


Рис. 23

Перевірка на перетин вікна багатокутником може бути виконана тим же способом який застосовується в алгоритмі Сазерленда – Ходжена. Тобто для всіх ребер багатокутника визначається по яку сторону розташовані вершини вікна. Якщо векторні добутки матимуть різні знаки , то вікно і багатокутник перетинаються. Якщо ж всі знаки однакові, то вікно лежить по одну сторону від ребра, тобто багатокутник може бути або зовнішнім, або що охоплює.

Для того щоб визначити тип багатокутника у цьому випадку існують що найменше два підходи. Перший з них це уже знайомий тест про належність точки до області. Якщо всі вершини вікна будуть зовнішніми відносно багатокутника, то він природно є зовнішнім, а в противному випадку буде охоплювати вікно.

У другому підході обчислюється сумарний кут, на який повернеться промінь, що виходить з деякої точки вікна (звичайно центра), при послідовному обході вершин багатокутника. Якщо сумарний кут дорівнює 0, то багатокутник - зовнішній. Якщо ж кут дорівнює $N \times 360^{\circ}$, те багатокутник охоплює вікно N раз.

Разом з безумовною перевагою (рекурсивністю) алгоритм має не менш яскравий недолік – він дуже погано формалізований. Крім цього у випадку, коли багатокутник є внутрішнім або перетинає вікно, виникає задача відсікання. Якщо помітити, що після розбивки вікна що охоплюють і зовнішні багатокутники успадковуються від вихідного вікна, то залишається тільки пе-

ревірка на перетині і на те, що багатокутник є внутрішнім. Це лає можливість з формулювати наступний варіант алгоритму.

Якщо перевірка на перетин дає позитивний результат, то провадиться наступне розбиття вікна. У противному випадку за допомогою променевого тесту визначається чи охоплює найближчий багатокутник вікно, і тільки у цьому випадку вікну надаються атрибути багатокутника.

Раніше вже відмічалось, що існують алгоритми для яких є суттєвим порядок обробки елементів сцени.

Ефективність двох останніх алгоритмів може бути суттєво підвищена, якщо багатокутники сцени перед переходом до простору зображень упорядкувати по зростанню Z - координати. В разі перетину їх можна розбиті на частини по відповідним лініям. Тоді оцінка глибини буде проводитися тільки у випадку відрізків з спільною точкою.

І на кінець, при такому підході у методі Варнака, досить провести перевірку на охоплення тільки найближчого до спостерігача багатокутника і при негативному результаті продовжити розбиття .

5.4 Метод трасування променів

Метод трасування променів є більш загальним ніж метод Z-буфера, постільки за його допомогою можна реалізувати такі ефекти як прозорість, переломлення і відбиття променів, тіні та інші. В його основі лежить той факт, що промінь світла, що рухається від джерела, відбивається від тіла і попадає до спостерігача. Якщо цілком простежити шлях, пройдений променем за такою схемою, то велика кількість обчислень, необхідна для виконання цієї задачі, виявиться марною внаслідок екранування променя іншим тілами.

Аппель запропонував метод, заснований на трасуванні в зворотному напрямку, тобто від спостерігача до тіла, суть якого полягає в наступному. Через точку спостереження і кожний піксель екрану проводиться промінь і визначаються точки його перетину з тілами сцени (рис.24). Видимою буде та точка тіла, відстань від якої до спостерігача є найменшою.

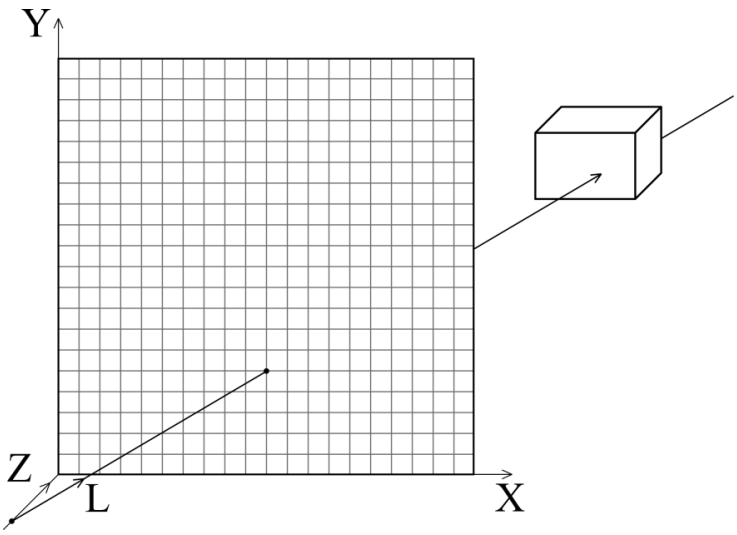


Рис. 24

Сам пошук точки перетину трасуючого променя, рівняння якого має вигляд

$$x = x_s t, \quad y = y_s t, \quad z = dt \quad (5.3)$$

з площиною, що несе грань тіла (5.1) є елементарною операцією, але після цього ще треба рішити задачу про належність точки до довільно орієнтованого у просторі багатокутника.

Тому щоб уникнути марних обчислень, доцільно помістити об'єкт у оболонку і , якщо не буде перетинів променя з оболонкою, то їх не слід шукати і для об'єкта. Найбільш просто цей прийом реалізується для випадку сферичної оболонки. Нехай сферична оболонка має радіус R, а її центр знаходиться в точці (x_0, y_0, z_0). Тоді мінімальна відстань між цією точкою і прямою заданою у параметричній формі , що проходить через точки P_1 і P_2 , тобто $P = P_1 + (P_2 - P_1)t$, визначається як

$$d^2 = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 .$$

А параметр t, що відповідає найближчій точці дорівнює

$$t = -\frac{\alpha(x_1 - x_0) + \beta(y_1 - y_0) + \gamma(z_1 - z_0)}{\alpha^2 + \beta^2 + \gamma^2} .$$

У нашому випадку з рівняння (5.3) маємо

$$\alpha = x_s,\ \ \beta = y_s,\ \ \gamma = 1,\quad x_1 = y_1 = z_1 = 0.$$

Якщо $d^2 > R^2$, то промінь оболонку не перетинає.

Контрольні питання.

Для якого виду проективних перетворень метод Z-буферу є найбільш ефективним?

В якому з методів вилучення невидимих поверхонь (МВНП) багатокутник необхідно попередньо представити в растровій формі?

В якому з (МВНП) необхідно визначити взаємне положення вікна і багатокутника?

В якому з (МВНП) може бути виключена з розгляду вся грань повністю?

Які з (МВНП) працюють в об'єктному просторі в просторі зображень, а які в об'єктному просторі?

В якому з (МВНП) в об'єктному просторі обробляються всі пікселі екрану?

Який з (МВНП) дозволяє реалізувати ефект переломлення променів і можна імітувати прозорість матеріалу?

ЛАБОРАТОРНА РОБОТА № 6

Вилучення невидимих граней

Мета роботи. Оволодіння технікою вилучення невидимих площин на прикладі роботи алгоритму z – буфера.

Завдання до лабораторної роботи

Для сцени, що складається з двох опуклих багатогранників, які перетинаються

- виконати видові перетворення;
- обчислити нормалі площин, які утворюють сцену;
- перевести площини у растрову форму;
- створити буфери глибини і кольорових атрибутів;
- вилучити невидимі площини методом z – буфера.

РОЗДІЛ 6
ПОБУДОВА РЕАЛІСТИЧНИХ ЗОБРАЖЕНЬ

При побудові зображень варто брати до уваги два ефекти, обумовлені особливостями людського зору :

- області з однаковою інтенсивністю освітлення на більш темному фоні сприймаються як більш яскраві;

поблизу границь областей постійної інтенсивності освітлення з'являються смуги (смуги Маха).

6.1 Моделі освітлення.

Модель освітлення це математичне представлення фізичних властивостей джерел освітлення і поверхонь та їх взаємного розташування . Найпростішої модель є фонове освітлення коли проводиться рівномірне освітлення сцени з усіх сторін інтенсивністю I_a. Сумарна кількість відбитих розсіяний променів довільній поверхнею визначається коефіцієнтом фонового відбиття K_a . Таким чином інтенсивність освітлення деякої поверхні є

$$I = I_a k_a .$$

З цієї формули витікає, що фонова складова освітленості не залежить від просторових координат освітлюваної точки і джерела. Дана модель не враховує і не використовує ніяких фізичних законів взагалі і часто просто задається якесь глобальне фонове освітлення всієї сцени.

Промінь, який попадає на тіло безпосередньо від джерела освітлення називається первинним. Моделі освітлення, які враховують тільки первинні промені,називаються локальними. Глобальні моделі на відміну від локальних враховують також промені, які потрапляють на тіло після відбиття від інших об'єктів або заломлення при проходженні прозорих тіл (вторинні промені).

За принципом побудови моделі освітлення прийняти поділяти на фізично обґрунтовані і емпіричні. Фізично обґрунтовані моделі побудовані на апроксимації властивостей реальних матеріалів. В емпіричних моделях як самі параметри, так і форма функціональної залежності інтенсивності

освітлення від параметрів визначається шляхом підбору з міркувань якості отриманого зображення.

В загальному випадку частина світлової енергії, що потрапляє на тіло, в залежності від стану поверхні і фізичної природи тіла може поглинатися, а частина відбиватись. Коли більша частина світла, що попадає на тіло, відбивається , має місце дзеркальне відбиття, якщо переважним є розсіювання - дифузійне відбиття . Цим двом граничним випадкам відповідають дві моделі освітлення.

Позначимо через $\overline{L}$ - напрямок променя світла, $\overline{R}$ - вектор відбиття, а θ - кут між цими векторами і нормаллю $\overline{n}$ до поверхні у даній точці поверхні, рис.25

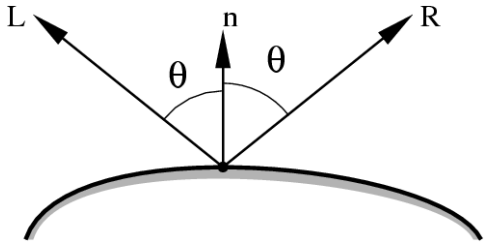


Рис. 25

За законом Френеля вектори L і R рівні між собою і розташовані в одній площині з нормаллю.

Модель Ламберта описує ідеальне дифузне освітлення. Вважається, що світло при попаданні на поверхню розсіюється рівномірно по всім напрямках. При розрахунку такого освітлення враховується тільки орієнтація поверхні (нормаль N) і напрямок на джерело світла (вектор L)

$$I = I_d K_d \cos\theta \ , \qquad (6.1)$$

де I - інтенсивність, що спостерігається, I_d, - інтенсивність точкового джерела освітлення, k_d - характеристика матеріалу. Величини параметрів, що входять у цю модель змінюються в межах $0 \le k_d \le 1, 0 \le \theta \le \pi/2$. Для зручності всі вектори, описані вище, беруться одиничними .В цьому випадку косинус кута між ними співпадає зі скалярним добутком.

Модель Ламберта є однією з найпростіших моделей освітлення і дуже часто використовується в комбінації інших моделей, практично в будь-який інший моделі освітлення можна виділити дифузну складову. Більш-менш рівномірна частину освітлення (без присутності будь-якого сплеску) як правило буде представлятися моделлю Ламберта з певними характеристиками. Дана модель може бути дуже зручна для аналізу властивостей інших моделей (за рахунок того, що її легко виділити з будь-якої моделі і аналізувати залишилися складові).

У моделі ***дзеркального відбиття*** інтенсивність освітлення буде залежати також від положення спостерігача, яке задається вектором *S*, рис. 26.

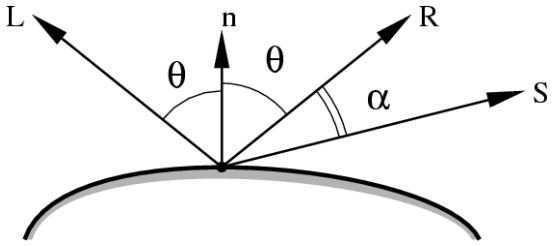


Рис. 26

У цій моделі інтенсивність обчислюються як

$$I = I_s W(\lambda,\theta)\cos^{\beta}\alpha, \qquad (6.2)$$

де I_s - інтенсивність джерела , $\cos^n\alpha$ - функція фокусування променя, .

У загальному випадку функція Фонга $W(\lambda,\theta)$ для кожного тіла, яке дзеркально відбиває світло матиме свій вигляд і крім того має досить складну залежність від довжини світової хвилі λ і кута відбиття. Тому Фонг запропонував замінити її константою $K_s = W(\lambda,\theta)$, величина якої знаходиться підбором для кожного конкретного випадку.

Модифікацією моделі Фонга є Блінна-Фонга , яка не містить відбитого променя і цим спрощує обчислення. Принципової різниці між двома цими моделями немає. Вводиться новий вектор

$$\overline{H} = \frac{\overline{L} + \overline{S}}{\left|\overline{L} + \overline{S}\right|},$$

який є бісектрисою векторів $\overline{L}$ і $\overline{S}$ і модифікація зводиться до заміни скалярних добутків $(\overline{RS})^{\beta} \to (\overline{RH})^{\beta}$

Коли освітлення є результатом дії декількох джерел, значення інтенсивності необхідно просумувати по всіх джерелах. Тоді загальна модель освітлення з врахуванням відстані r між точкою освітленні і його джерелом набуде вигляду

$$I = I_a k_a + \sum_{i=1}^{N} \frac{I_i k_d + I_i k_s \cos^{\beta} \alpha_i}{r_i + k},$$

де k - константа, яка також знаходиться шляхом підбору.

Визначення векторів нормалі і відбиття. Для того , щоб використати формули (6.1),(6.2) для розрахунку інтенсивності необхідно знайти вектор відбиття R і вектор нормалі по заданому вектору L . Хоч вектор нормалі для граней тіла вже є знайденим, нижче при зафарбуванні нам знадобиться його значення у вершинах.

Нехай рівняння площини , що сходяться у даній вершині відомі і мають вид (5.1). Тоді компоненти нормаль у цій вершині можуть бути обчислені як проста сума відповідних коефіцієнтів :

$$A = \sum_{i=1}^{n} A_I \ , \ B = \sum_{i=1}^{n} B_I \ , \ C = \sum_{i=1}^{n} C_I$$

Якщо не нормувати цей вектор, то вклад граней з більшою площею буде виявлятися сильніше.

В окремих випадках рівняння граней невідомо, а заданий набір ребер, що сходяться у вершині. Тоді для випадку, приведеного на рис. 25, нормаль у вершині V_1 буде обчислюватись як

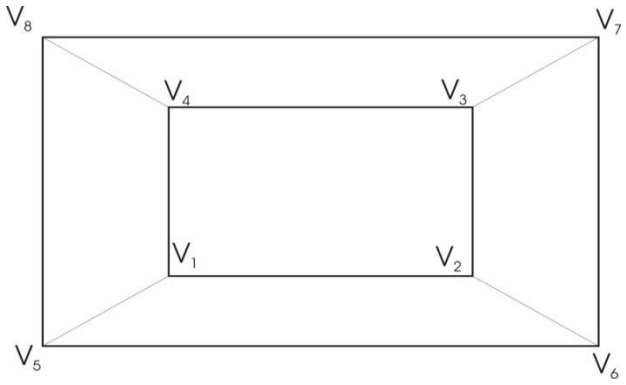


Рис. 27

$$N= V_1V_2 \times V_1V_4 + V_1V_2 \times V_1V_5 + V_1V_4 \times V_1V_5$$

При визначені вектора відбиття необхідного у моделі дзеркального відбиття підходи до рішення цієї задачі залежать від положення вектора освітлення і нормалі відносно системи координат. У випадку ,що часто зустрічається , коли світло падає уздовж осі $0z$, з закону Френеля за допомогою елементарних геометричних перетворень для компонентів вектора відображення R одержуємо, рис.28

$$R_z = 2n_z^2 - 1, \quad R_y = 2n_z n_y, \quad R_x = 2n_z n_x$$

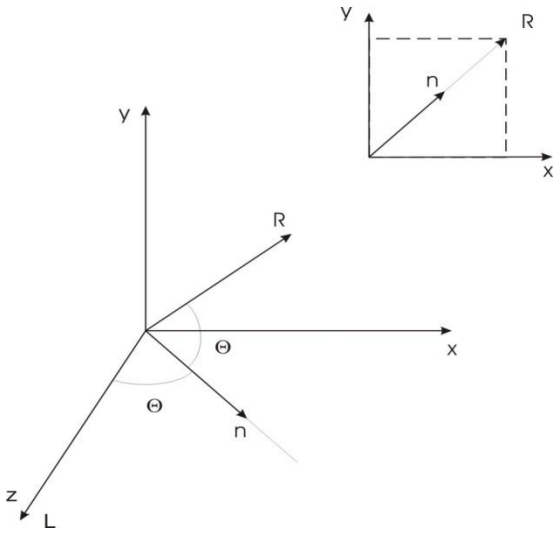


Рис. 28

У випадку коли вісь $0z$ співпадає з нормаллю, маємо

$$R_x = -L_x, \; R_x = -L_x, \; R_z = L_z$$

У довільному випадку освітлення, вектор відбиття можна одержати з закону Френеля, склавши відповідні рівності векторних і скалярних добутків

$$n \otimes L = R \otimes n, \quad nL = nR$$

У цьому випадку одержуємо усереднене рішення

$$R = [N^T N]^{-1} N^T B,$$

де N і B наступні матриці

$$N = \begin{bmatrix} 0 & -n_z & n_y \\ \mathrm{n_z} & 0 & -n_x \\ \mathrm{-n_y} & \mathrm{n_x} & 0 \end{bmatrix} B = \begin{bmatrix} n_z \mathrm{L_y} - n_y \mathrm{L_z} \\ n_x \mathrm{L_z} - n_z \mathrm{L_x} \\ n_y \mathrm{L_x} - n_x \mathrm{L_y}\ 0 \\ n_x \mathrm{Lx_x} + n_y \mathrm{L_y} + n_z \mathrm{L_z} \end{bmatrix}$$

6.2 Моделі зафарбування.

Останнім кроком у побудові зображень є зафарбування. Як і раніше вважаємо, що візуалізації підлягає набір, багатокутників, що мають свої геометричні і кольорові атрибути. Існують кілька моделей зафарбування, що відрізняються як по ступені необхідних ресурсів, необхідних для їхньої реалізації, так і по якості отриманому результату.

Однотонна модель зафарбування, вона ще має назву гранування (faceting), є найкращою з оцінки кількості операцій, але найгіршою по ступеню реалістичності. У цьому випадку зафарбування в межах даного багатокутника проводиться з постійною інтенсивністю, що приводить до відсутності ефекту глибини, тому що нормаль, а значить і інтенсивність, будуть постійними для всієї грані. Крім цього отримане зображення не виглядає плавним. У випадку апроксимації гладкої поверхні багатогранником, при однотонному зафарбовуванні неминуче виявляться ребра, оскільки сусідні грані з різними напрямками нормалей мають різний колір. Ефект смуг Маха додатково посилює цей недолік. Для його усунення при використанні цього способу зафарбовування можна лише збільшити число граней багатогранника, що призводить до збільшення обчислювальної складності алгоритму.

Один із способів усунення дискретності інтенсивності й зафарбовування

був запропонований Гуро. Його метод полягає в тому, що використовуються не нормалі до плоских гранях, а нормалі до апроксимуючої поверхні, побудовані в вершинах багатогранника. Після цього обчислюються інтенсивності у вершинах, а потім у всіх внутрішніх точках багатокутника виконується білінійна інтерполяція інтенсивності.

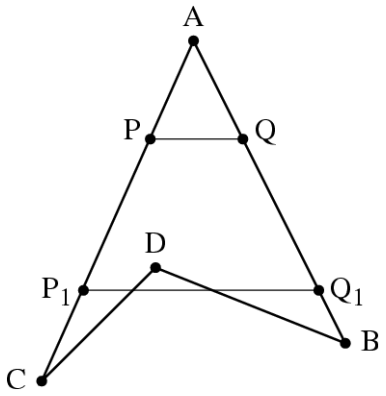


Рис. 29

Нехай інтенсивності у вершинах трикутника ABC відомі і рівні відповідно I_A, I_B, I_C. Тоді інтенсивність освітлення в межах відрізка AB визначається як

$$I_{AB} = I_A + (I_B - I_A)t$$

де параметр $t \in [0,1]$. Аналогічно для відрізків AC і PQ маємо (рис.29)

$$I_{AC} = I_A + (I_C - I_A)t \qquad (6.3)$$

$$I_{PQ} = I_P + (I_Q - I_P)t \qquad (6.4)$$

Тут яскравість і колірна насиченість елементів кожного багатокутника плавно змінюються в інтервалі між значеннями, обчисленими в його вершинах. При цьому поверхні відтворених предметів набувають ідеальну сюрреалістичну гладкість, начебто ми бачимо їх при непрямому освітленні. Основним дефектом в розглянутих способах зафарбування є смуги Маха, які проявляються у розриві неперервності інтенсивності на ребрах суміжних граней.

До недоліків методу Гуро слід віднести ще те, що він добре працює тільки з дифузійної моделлю відбиття. Форма відблисків на поверхні і їх

розташування не можуть бути адекватно відтворені при інтерполяції на багатокутниках. Крім того така побудова сильно залежить від характеру розбивки. .

Фонг запропонував замість інтерполяції інтенсивносте й провести інтерполяцію вектора нормалі до поверхні на рядку сканування. але зате дає кращу апроксимацію кривизни поверхні. Тому дзеркальні властивості поверхні відтворюються набагато краще. Нормалі до поверхні в вершинах багатогранника обчислюються так само, як і в методі Гуро.

Формально ***модель Фонга*** відрізняється від моделі Гуро тільки тим, що в приведених вище робочих формулах, значення інтенсивності необхідно поміняти на компоненти вектора нормалі. При цьому кількість обчислень різко зростає, оскільки формули інтерполяції (6.3), (6.4) застосовуються до трьох компонентів вектора нормалі і для кожного пікселя необхідно обчислювати інтенсивність за формулами моделей освітлення.

Цей метод дозволяє усунути ряд недоліків методу Гуро. Зафарбовування по Фонгу вирішує проблеми смуги Маха, оскільки забезпечує плавну зміна яскравості і насиченості не тільки уздовж ребер кожного багатокутника, але і по самій поверхні. При цьому навіть дзеркальні відблиски на поверхнях виглядають цілком правдоподібно.

Контрольні питання.

В якій з моделей освітлення можна можно регулювати фокусування променя?

В якій з моделей освітлення колір об’єкта визначається кольором матеріла?

В якій з моделей освітлення колір об’єкта визначається кольором джерела освітлення?

Який з (МЗ) потребує найбільшого обсягу обчислювальних ресурсів?

В якому з (МЗ) інтенсивність освітлення поточного пікселя визначається шляхом лінійної інтерполяції інтенсивностей в вершинах граней?

В якому (МЗ)інтенсивність освітлення поточного пікселя визначається за допомогою лінійної інтерполяції нормалей в вершин граней?

В якому з (МЗ) необхідно визначити нормалі всіх вершин грані?

В якому з (МЗ)достатньо мати тільки значення нормалі грані?

ЛАБОРАТОРНА РОБОТА № 7

Моделі освітлення і методи зафарбування.

Мета роботи. Оволодіння технікою побудови реалістичних зображень.

Завдання до лабораторної роботи . Використовуючи результати лабораторної роботи № 6 для видимих площин сцени

- обчислити інтенсивність відповідно з моделями дифузійного і дзеркального освітлення;
- провести зафарбування тіл сцени методом гранування і методом Гуро.

6.3 Спеціальні ефекти при побудові зображень.

Прозорість. В реальному світі існують прозорі об’єкти, які пропускають світло(вода, скло і т.п.). При переході з одного середовища до іншого , наприклад з повітря у воду, світловий промінь переломлюється. Переломлення описується законом Сінеліуса, згідно якого

$$\eta_1 \sin\theta = \eta_2 \sin\theta'$$

де η_1, η_2 - показники переломлення середовищ, θ - кут падіння, θ' - кут переломлення.

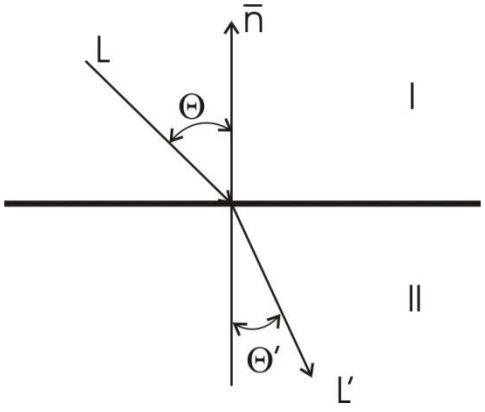


Рис. 30

Жодна речовина не пропускає все світло, що падає, частина його відбивається. Переломлення світла приводить до незвичних ефектів: видимі поверхні можуть стати невидимими, освітлені неосвітленими і навпаки.

Проходячи через прозорий матеріал, промінь потерпає природне ослаблення з коефіцієнтом $k = e^{\beta l}$, де l - довжина шляху всередині матеріалу, β - коефіцієнт прозорості. Але такий підхід є досить трудомістким за визначення величини l. На практиці частіше для обчислення інтенсивності освітлення променя, що проходить через прозоре середовище застосовується наступна формула

$$I = K_s[1 - F(\eta, \theta)]I_0,$$

де функція Френеля $F(\eta, \theta)$ визначається наступним чином

$$F(\eta,\theta) = \frac{1}{2}\left\{\left[\frac{\eta\cos\theta - \cos\theta'}{\eta\cos\theta + \cos\theta'}\right]^2 + \left[\frac{\cos\theta - \eta\cos\theta'}{\cos\theta + \eta\cos\theta'}\right]^2\right\}$$

Одержати рішення задачі побудови зображень з врахуванням прозорості і переломлення променів можна тільки за допомогою метода трасування.

У найпростішій моделі прозорості переломлення не розглядається і не враховується довжина шляху променя у середовищі, тобто зменшення його інтенсивності. При розрахунках по такій моделі можуть використовуватися будь-який алгоритми видалення невидимих поверхонь, що враховують порядок розташування багатокутників.

Прозорі багатокутники помічаються і якщо видима грань є прозорою то буфер кадру заповнюється лінійною комбінацією двох найближчих площин. Тобто інтенсивність визначається як

$$I = kI_1 + (1-k)I_2$$

де I_1- інтенсивність видимої площини, I_2- інтенсивність площини, що розташована безпосередньо за нею, $0 \le k \le 1$ – коефіцієнт прозорість ближнього багатокутника. Якщо k = 1, то він непрозорий . Якщо ж k = 0, то ближній багатокутник є цілком прозорим.

Тіні. Зображення з побудованими тінями мають більш реалістичний вигляд, і, крім цього є досить важливими для моделювання. Наприклад, об’єкт, що нас цікавить може бути невидимим тому, що на нього падає тінь. В технічних застосуваннях, зокрема в будівництві, проектуванні космічних апаратів, тінь повинна враховуватися при визначенні сонячної енергії.

Хоча тінь складається з двох частин: повної, при точковому освітленні і півтіні, при розподіленому освітленні, побудова проводиться звичайно для першого випадку.

Іноді тіні ділять на власну і проекційну. Власна тінь виходить коли сам об'єкт перешкоджає потрапляння світла на деякі його межі. При цьому алгоритм побудови тіні аналогічний алгоритму видалення нелицьових поверхонь. Якщо один об'єкт перешкоджає попаданню світла на інший, то виходить проекційна тінь.

Безпосереднє тіні можна будувати за допомогою формул отриманих для центральних проекцій. Нехай точка $L(x_l, y_l, z_l)$ визначає положення джерела світла, а точка $P(x_p, y_p, z_p)$ є точкою яка відкидає тінь. Тоді напрям променя освітлення визначається як $\bar{a}(x_P - x_L, y_P - y_L, z_P - z_L)$. Після знаходження коефіцієнтів рівняння площини (5.2) і нормування (5.3)

$$x_S = x_L + a_x t \quad y_S = y_L + a_y t \quad z_S = z_L + a_z t$$

На практиці досить часто необхідно будувати так звану тінь «на землю», тобто на площину $z = 0$. В цьому випадку формули суттєво спрощуються і відповідна точка в тіні визначається як

$$x_S = \frac{x_p z_L - x_L z_P}{z_L - z_P}, y_S = \frac{y_p z_L - y_L z_P}{z_L - z_P}$$

По суті визначення об'єктів, що потрапили в тінь є задачею видалення невидимих поверхонь для того випадку , коли точка спостереження не співпадає з точкою, у якій знаходиться джерело освітлення. Тому рішення цієї задачі можна проводити на основі алгоритмів вилучення невидимих повер-

хонь, які застосовуються в два етапи – визначення видимості спочатку з точки спостереження, а потім з точки освітлення.

Фактура. В комп'ютерній графіці фактурою називається деталізація будови поверхні. Звичайно розглядаються два види деталізації. У першому на гладку поверхню наноситься визначений візерунок і в результаті отримаємо також гладку поверхню. Реалізація цього виду фактури по суті складається з знаходження функціонального перетворення, яке відображає криву на поверхні в її зображення на екрані. Звичайно для цього цілком достатньо локально-лінійних функцій.

У другому випадку необхідно зобразити поверхню, яка б здавалась нерівною. Нерівності можуть моделюватися збурення нормалі поверхні. Нехай гладка і поверхня задається функцією своїх координатних ліній $r = r(u,v)$. Вводиться функція збурення $\hat{r} = \hat{r}(u,v)$, яка імітує нерівності. В якості останньої може бути взята будь яка функція, що має часткові похідні. Тоді вектор нормалі, для визначення яскравості поверхні з нерівностями обчислюється як

$$\hat{n} = n + \frac{1}{|n|}[\hat{r}_u \times (n \times r_v) + \hat{r}_v \times (r_u \times n)]$$

Інший спосіб, що використовується для імітації нерівностях оснований на поняттях фрактальної геометрії. Для того щоб одержати полігональну фрактальну поверхню початковий багатокутник рекурсивно розбивається на фрагменти (рис. 31). Для цього можна, наприклад, випадковим чином змістити центр і середини сторін.

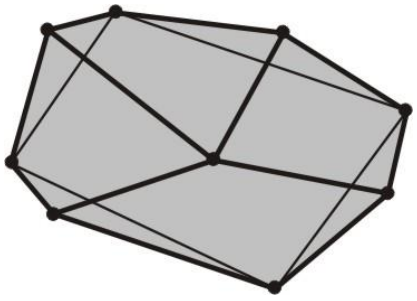


Рис. 31

Одна з переваг фрактальних поверхонь полягає в тому, що їх можна розбивати „нескінченно” і отримати будь-який рівень деталізації. Він може залежати від положення спостерігача: чим ближче точка спостереження, тим з більшим ступенем деталізації зображується об’єкт і навпаки.

6.4 Системи представлення кольорів.

Колірної моделі призначення для того , щоб дати засіб опису кольору в межах деякого колірного охоплення, у тому числі і для виконання інтерполяції кольорів. Наука про кольори називається колориметрією.

Світлова енергія, що падає на поверхню від джерела світла, може бути поглинена, відбита або пропущена. Кількість енергії залежить від довжини світлової хвилі. При цьому колір поверхні об'єкта визначається поглиненими довжинами хвиль.

Білими виглядять об’єкти , які відбивають більше 80 % світла, чорними – менше 3% , проміжні значення відповідають різним відтінкам сірого. Джерело світла називається ахроматичним, якщо світло, що спостерігається містить всі видимі довжини світла в приблизно рівній кількості, Якщо світло, що сприймається містить світло в довільних нерівних пропорціях, то воно називається хроматичним. І на кінець, коли домінує якась визначена довжина хвилі, то маємо монохроматичне світло.

Експериментально встановлено, що колір має тривимірну природу і як всякий тривимірний об’єкт його можна зобразити у вигляді розкладення по деякому базису

$$C = c_1 C_1 + c_2 C_2 + c_3 C_3$$

де C_1, C_2, C_3 - базисні функції, а c_1, c_2, c_3 - відповідні координати. Тому всі існуючі системи представлення кольорів по суті відрізняються тільки вибором базису. Наприклад існує система у якій в якості базису прийняті довжина світової хвилі, яскравість і насиченість, але подібні системи є не зовсім зручними для застосування в комп’ютерній графіці.

Людське око сприймає світлову енергію в діапазоні довжин хвиль від 400(фіолетовий) до 700 нм(червоний). Тому цілком обґрунтованим є вибір базису, який утворений кольорами, що розташовані на краях і в центрі спектру. Ця система відома як RGB (Red, Green, Blue - червоний, зелений, синій) по назвам базових кольорів, а постільки інші необхідні кольори можна одержати за рахунок змішання трьох основних кольорів, вона ще називається адитивною. Колір у цій системі визначається як

$$C = rR + gG + bB,$$

де координати r, g, b змінюються в межах від нуля одиниці.

У комп'ютерній графіці є два типи кольорових об'єктів - самосвітні, випромінюючі об'єкти, такі як екрани електронно - променевих трубок, плазмові панелі, матриці світло діоди і т.п. і не самосвітні об'єкти, що відбивають або переломлюють падаюче на них світло, такі як, наприклад, відбитки на папері, світлофільтри і т.п.

Для самосвітних об'єктів використовується адитивне формування відтінків а для не самосвітніх об'єктів звичайно застосовується субтрактивне формування відтінків. Це представлення основане на відніманні з падаючого світла хвиль визначеної довжини , а відповідна система має назву CMY (Cyan, Magenta, Yellow – блакитний, пурпурний, жовтий). Кольори однієї моделі є додатковими до кольорів іншої моделі. Додатковий колір – це колір, що доповнює даний до білого. Додатковий для червоного - блакитний (зелений + синій), додатковий для зеленого - пурпурний (червоний + синій), додатковий для синього - жовтий (червоний + зелений) і т.д.

Приклад субтрактивного формування відтінків показаний на рис. 32. При освітленні падаючим білим світлом у шарі блакитної фарби зі спектра білого кольору поглинається червона частина, потім зі світла, що залишилось, у шарі пурпурної фарби поглинається зелена частина спектра, відбите від поверхні паперу світло ще раз піддається поглинанню й у результаті ми бачимо синій колір.

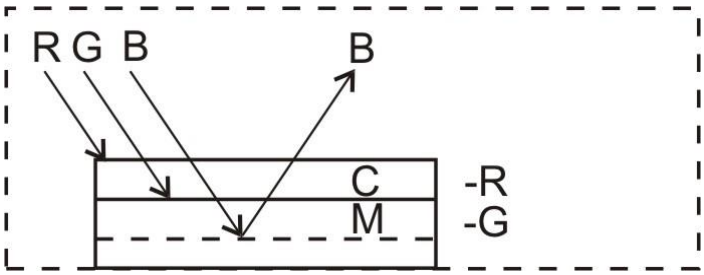


Рис. 32

Ці моделі змішання можна записати у вигляді простої формули. В лівій частині формули фігурують кольори, які змішуються, в правій – колір додатковий до того, який не представлений в лівій частині. І тоді маємо

R+G = Y R+B= M G+B=C **R G B**

C+M= B C+Y=G Y+M+Y=R **C M Y**

Система координат RGB - куб з початком відліку (0,0,0), що відповідає чорному кольору (рис. 33). Максимальне значення RGB - (1,1,1) відповідає білому кольору.

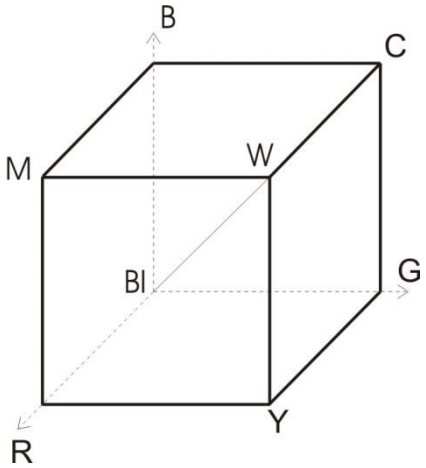


Рис. 33

Постільки кольори моделі CMY є додатковими до кольорів моделі RGB, то система координат CMY - той же куб, що і для RGB, але з початком відліку в точці з RGB координатами (1,1,1), що відповідає білому кольорові. Перехід між просторами RGB і CMY визначаються наступним чином:

$[R\,G\,B] = [111] - [C\,\mathrm{M}\,\mathrm{Y}]$

Причому одиничний вектор-рядок у моделі RGB - представлення білого кольору, а в моделі CMY - чорного.

Різновидом цієї моделі є модель CMYK, в яку доданий чорний колір (від англ. blac**K**)

На відміну від моделей RGB, CMY, модель HSV орієнтована на користувача і забезпечує можливість явного завдання необхідного відтінку кольору.

Якщо зробити проекцію кольорового кубу RGB вздовж чорно - білої діагоналі, одержимо шестикутник з основними і додатковими кольорами в вершинах. При зниженні насиченості або чистоти основних кольорів розмір і можливий кольорове охоплення кубу RGB зменшується , тому відповідна шестикутна проекція теж буде зменшуватися. Якщо проекції куба RGB і його напівкубів зібрати вздовж головної діагоналі, що відповідає є світлоті кольору від чорного(0) до білого (1), то отримаємо об'ємний шестигранний конус моделі HSV(Hue,Saturation,Value-колірний тон, насиченість, кількість світла або світлота) - модель, орієнтована на людину і можливість явного завдання, що забезпечує, необхідного відтінку кольору (рис.34). Підпростір, обумовлений даною моделлю - перевернена шестигранна піраміда.

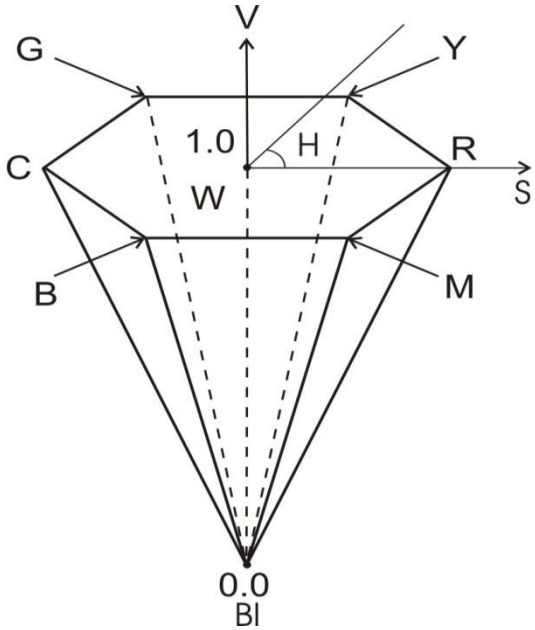


Рис. 34

Інтенсивність вздовж осі піраміди зростає від 0 в вершині до 1 на верхній грані, де вона є максимальною для всіх кольорів. Насиченість визначається віддалю від осі, а тон кутом (0° -360°), що відраховується від червоного. Зокрема, 0° - червоний, 60° - жовтий, 120° - зелений, 180° -блакитний, 240° - синій, 300° - пурпурний, тобто додаткові кольори розташовані один проти другого (відрізняються на 180°). Для того, щоб червоний був у початку відліку колірний куб був повернений на 120° проти годинникової стрілки. Насиченість змінюється від 0 на осі до 1 на грані шестикутника. Слід відмітити, що насиченість залежить від колірного охоплення, тобто від віддалі від осі для кожного V. При S=1 колір або його доповнення є повністю насиченими. Не нульова лінійна комбінація трьох основних тонів не може бути повністю насиченою. Якщо S=0, то тон H є невизначеним, тобто на центральній осі знаходяться ахроматичні сірі кольори. .

Модель HSV відповідає способу , яким складає кольори художник. Чистим пігментом відповідають значення V = 1, S=1, розбіленим - кольори з збільшенню змістом білого, тобто з меншим S. Відтінкам – кольори зі зменшеним V = 1, які утворюються при збільшенні чорного. Тон змінюється як при зменшенні V, так і S.. Перехід між колірним просторам RGB і HSV виконується безпосереднє за допомогою геометричних співвідношень між шестигранним конусом і кубом.

Інша система HLS (Hue, Lightness, Saturation - колірний тон, світлота, насиченість) - модель орієнтована на людину і можливість явного завдання, що забезпечує, необхідного відтінку кольору Ця модель утворює підпростір, що представляє собою подвійну піраміду, у якій чорний колір задається вершиною нижнього конуса і відповідає значенню L = 0, білий колір максимальної інтенсивності задається вершиною верхнього конуса і відповідає значенню L = 1. Загалом, систему HLS можна представити як отриману з HSV "витягуванням" точки V=1, S=0, що задає білий колір, нагору для утворення верхньої піраміди.

У найбільш розповсюджених растрових дисплеях - дисплеях з таблицею кольоровості значення кодів пікселів, що заносяться у відео пам’ять, є індексами елементів таблиці кольоровості. При необхідності відображення деякого піксела, на екран по його значенню вибирається елемент таблиці кольоровості, що містить трійку значень - RGB. Ця трійка і передається на монітор для завдання кольору піксела на екрані.

У повно колірних дисплеях для кожного піксела у відео пам’ять заноситься трійка значень RGB. У цьому випадку при необхідності відображення пік села, з відео пам’яті безпосередньо вибирається трійка значень RGB, що і передається на монітор (але може і передаватися в таблицю кольоровості).

1. У моделі RGB легко задавати яскравості для одного з основних кольорів, але принаймні важко задати відтінок з необхідним колірним тоном і насиченістю. У різного роду графічних редакторах ця задача звичайно вирішується за допомогою інтерактивного вибору з палітри квітів і формуванням кольорів у палітрі шляхом підбору значень RGB до одержання необхідного візуального результату. Більш зручно в цьому випадку використовувати моделі HVS або HLS, що дозволяють безпосередньо задати необхідний відтінок. Звичайно, при занесенні даних у таблицю кольоровості або для повно колірних дисплеїв – у відео пам’ять, необхідно перевести їх значення у систему RGB.

У більш складних моделях враховується освітленість об'єктів не тільки прямими променями з боку джерела світла, але і відбитими променями з боку інших освітлених об'єктів.

РОЗДІЛ 7

ЗАДАЧІ І МЕТОДИ КОГНІТИВНОЇ ГРАФІКИ.

Прийнято виділяти наступні три основні задачі когнітивної графіки [2]:

1) створення моделей подання знань, в яких можна представити як об'єкти, характерні для логічного (символічного, алгебраїчного) мислення, так і об'єкти, характерні для образного мислення;

2) візуалізація тих знань, для яких поки неможливо підібрати текстового опису;

3) пошук шляхів переходу від образу до формулювання гіпотези про механізми і процесах, представлених цими образами.

Когнітивна комп’ютерна графіка виконує дві функції - ілюстративну і когнітивну. Ілюстративна функція дозволяє дати адекватне візуальне оформлення лише того, що вже відомо, тобто. вже існує або в навколишньому нас світі, або як ідея в голові дослідника. Когнітивна функція полягає в тому, щоб за допомогою якогось зображення отримати нове, яке ще не існує навіть в голові фахівця знання або принаймні спосіб відати інтелектуальному процесу отримання цього знання.

Розробники систем інженерного аналізу, автоматизованого проектування і комп'ютерних систем переважно мають справу з другою з наведеної вище класифікації задач когнітивної графіки, коли знання про технічний об'єкт, отримані в ході досліджень на багато вимірних математичних моделях і представлені у звичайній символьно-цифровій формі, стають недоступними для аналізу людиною через великий обсяг інформації. Розглянемо далі деякі способи відображення полів фізичних характеристик технічних об'єктів і алгоритми побудови відповідних зображень, що мають високий когнітивний потенціал.

Вважається, що зображуване полі фізичних характеристик представлено у вигляді дискретних значень у вузлах плоскої сітки елементів (ПСЕ) трикутної або чотирикутної форми. Ця сітка може відображати або все поле, або його фрагмент, наприклад, перетин тривимірного поля площиною.

Така форма представлення параметрів природна для ряду чисельних сіткових методів, наприклад, дуже поширений метод скінченних елементів передбачає сітковий апроксимацію.

На вході прикладних графічних програм, що реалізують розглянуті нижче алгоритми, має бути топологічний і геометричний опис ПСЕ зі значеннями відображаються характеристик в вузлах мережі. Топологію мережі зручно зберігати у вигляді матриці, в кожному рядку якої зазначений номер елемента ПСЕ і номера вузлів, що його оточують . Геометричний опис ПСЕ - це матриця, в рядках якої зазначені координати вузлів сітки.

Залежно від способу візуалізації будемо використовувати два види апроксимації відображаються параметрів в межах елемента ПСЕ: постійну і білінійну. Для постійної апроксимації в межах чотирикутного елемента ПСЕ величина зображуваного параметра $T = \frac{1}{4}\sum_{i=1}^{4} T_i$, де T_i - величини параметрів в вузлах мережі, що оточують елемент ПСЕ. Для білінійної апроксимації введемо безрозмірні координати ζ і η та допоміжний квадрат (рис. 35).

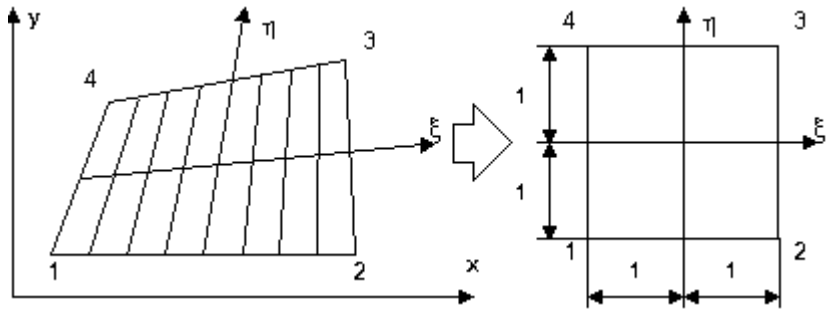


Рис.35

Перетворення координат і зображуваного параметра довільного чотирикутника у допоміжний квадрат здійснюється за формулою, аналогічною функції форми в методі скінченних елементів

$$[X\ \mathrm{Y}\ \mathrm{T}]=\frac{1}{4}[(1-\zeta)(1-\eta),(1-\zeta)(1+\eta),(1+\zeta)(1+\eta),(1+\zeta)(1-\eta)]\times$$
$$\times\begin{bmatrix} X_1\ \mathrm{Y}_1\ \mathrm{T}_1 \\ X_2\ \mathrm{Y}_2\ \mathrm{T}_2 \\ X_3\ \mathrm{Y}_3\ \mathrm{T}_3 \\ X_4\ \mathrm{Y}_4\ \mathrm{T}_4 \end{bmatrix} \qquad (7.1)$$

Для регуляризації алгоритмів елемент трикутної форми вважатимемо окремим випадком чотирикутника, у якого суміщені два сусідніх кута.

Суцільні кольорографічне зображення. Суть цього способу візуалізації полягає в тому, що внутрішня область ПСЕ зафарбовується в різні кольори, що відповідають певним інтервалам величини зображуваного параметра. Зазвично використовується колірна гамма, в якій у напрямку зниження величини параметра кольору змінюються від теплих (червоного і жовтого) до холодних (синього і фіолетового). Зображення будується за елементами ПСЕ Найпростішим алгоритмом зафарбовування елемента є метод сканування по ребрам (розділ 2.4) допоміжного квадрату, при цьому необхідно з (7.1) отримати відповідні кроки сканування $\Delta\zeta$, $\Delta\eta$. Якщо перехід від фізичних координат x_p, y_p до екранних x_s, y_s має вигляд

$$q_s = q_0 + m_q q_p,\ \ q = x, y,$$

де m_q – масштабні коефіцієнти, то кроки по допоміжній області визначаються як

$$\Delta\zeta = \frac{W_x}{x_3 + x_4 - x_1 - x_2}, \Delta\eta = \frac{W_y}{y_2 - y_1 + y_3 - y_4}$$

Тут $W_q = \frac{4}{m_q} - (q_1 + q_2 + q_3 + q_4)$.

Інший алгоритм також використовує ідею растрового сканування вздовж осі ζ і побудови кольорових відрізків уздовж осі η. У цьому алгоритмі колір відрізка визначається інтервалом T, а координати η кінців відрізка знаходяться з (7.1) для фіксованих значень ζ і меж заданих інтервалів T.

Якщо T^- і T^+ значення T на кінцях поточного інтервалу , то відповідні їм значення η^- і η^+ визначаються з (7.1) як

$$\eta = \frac{4T - (1-\zeta)(T_1 + T_2) - (1+\zeta)(T_3 + T_4)}{(1-\zeta)(T_2 - T_1) + (1+\zeta)(T_3 + T_4)}$$

Перехід кольорової палітри через границі елементів ПСЕ відбувається плавно, оскільки апроксимуюча функція (7.1) лінійна уздовж сторін чотирикутників ПСЕ, що забезпечує безперервність поверхні зображуваного параметра. На рис 36 приведено отримане цим методом тонове зображення оптимального розподілу матеріалу в пластині під навантаженням.

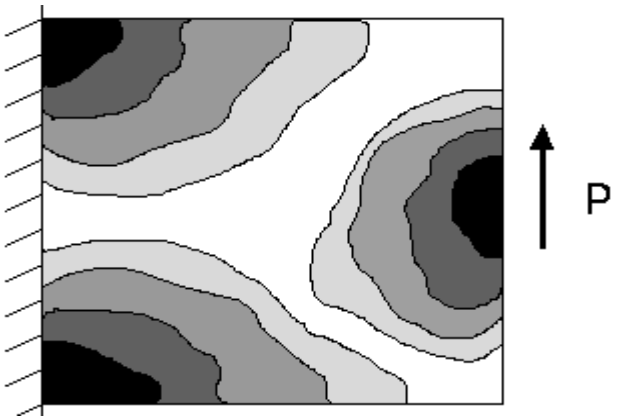


Рис.36

Точкові зображення. Поле кожного елемента ПСЕ на екрані дисплея заповнюється зафарбованими точками. Щільність розташування точок відповідає величині відображуваного параметра. Заповнення ділянок ПСЕ з постійною щільністю (це може бути поле всього чотирикутника або його частини) здійснюється за допомогою датчика випадкових чисел (ДВЧ). Таке заповнення згладжує розриви відображається поверхні навіть при постійній апроксимації параметра в межах одного елемента ПСЕ. Перед побудовою точкового зображення відшукується максимальне значення T, якому ставиться у відповідність щільність заповнення точок, рівна 80-90% від щільності суцільний зафарбовування. За цієї межі нормується в подальшому щільність заповнення точок на кожному чотирикутнику ПСЕ. При побудові зображення на елементі ПСЕ допоміжний квадрат попередньо розбивається

осями і на чверті, оскільки стандартні ДВЧ оперують числами в інтервалі [0,1]. У межах кожної чверті щільність точок вважається постійною. Координати точок і визначаються за допомогою ДВЧ, перетворюються за формулою (7.1) в координати X і Y, а далі переводяться в екранну систему координат. Колір точок визначається за заданими колірним інтервалах з використанням виразу (7.1).

Зображення у вигляді орієнтованих відрізків змінної довжини.

Цей спосіб застосовується для відображення векторних характеристик, наприклад, силових потоків. Для нього використовується постійний закон апроксимації параметрів в межах елемента ПСЕ. Орієнтовані відрізки зображуються в центрах елементів, їх довжини в обраному масштабі відповідають величинам параметрів На рис.37 приведено отриманий цим методом розподіл зусиль в пластині.. Перед побудовою зображення з міркувань наочності обчислюється максимальна довжина відрізка, відносно якої нормуються надалі відрізки на всіх елементах. Зображення будується за елементами ПСЕ. У центрі чотирикутника поміщається місцева прямокутна система координат, одна з осей якої орієнтується в напрямку зображуваного параметра. Далі в координатах місцевої системи визначаються кінцеві точки відрізка так, щоб його середина збіглася з центром елемента, проводиться перетворення отриманих координат в загальну систему і будується пряма лінія, що з'єднує кінцеві точки відрізка.

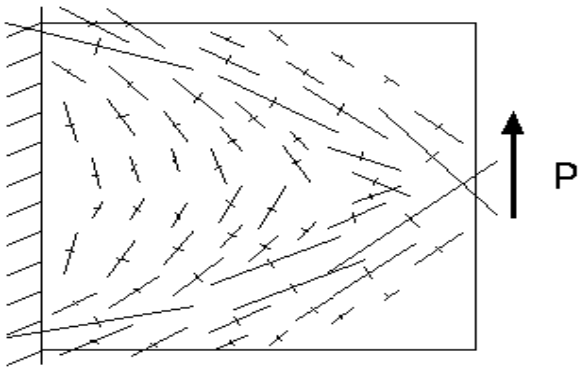


Рис. 37

Зображення у вигляді коротких орієнтованих відрізків постійної довжини. Цей спосіб візуалізації також призначений для відображення векторних характеристик. Після кожного елемента ПСЕ заповнюється за допомогою ДВЧ короткими орієнтованими відрізками постійної довжини. Щільність розташування відрізків відповідає величині зображуваного параметра . На рис.38 приведено отриманий цим методом розподіл зусиль в пластині. Перед побудовою зображення обчислюється з міркувань наочності максимальна щільність відрізків, щодо якої нормується щільність відрізків на всіх елементах ПСЕ. У центрі чотирикутного елемента ПСЕ поміщається прямокутна місцева система координат, одна з осей якої орієнтована в напрямку зображуваного параметра. Координати середніх точок відрізків визначаються за допомогою ДВЧ так, як це робиться при побудові точкових зображень. Далі побудова кожного відрізка проводиться, як і в попередньому алгоритмі.

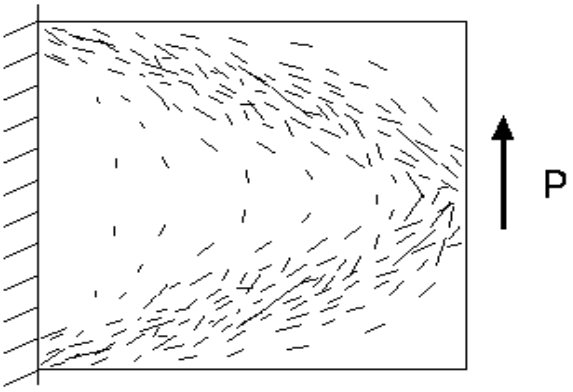


Рис.38

Контрольні питання.

Призначення допоміжного квадрату.

Методи відображення векторних величин.

Як забезпечується неперервність зображення на границях ПСЕ?

Яким чином проводиться перехід від координат допоміжного квадрату до екранних?

ЛІТЕРАТУРА

1. Аммерал Л. Принципы программирования в машинной графике. Пер. с англ. – М.: Сол Систем 1982
2. Гилой В. Интерактивная машинная графика. Пер. с англ. М.: Мир, 1981.
3. 1. Зенкин А.А. Когнитивная компьютерная графика. – М.: Наука, 1991.– 192 с.
4. Новаковский С.В. Цвет в цветном телевидении. М.: Радио и связь, 1988. 288 с.
5. Ньюмен У., Спрулл Р. Основы интерактивной машинной графики. Пер. с англ. М.: Мир, 1976.
6. Павлидис Т. Алгоритмы машинной графики и обработки изображений. Пер. с англ. М.: Радио и связь, 1986.
7. Роджерс Д. Алгоритмические основы машинной графики. Пер. с англ. М.: Мир, 1989. 512 с.
8. Роджерс Д., Адамс Дж. Математические основы машинной графики. Пер. с англ. М.: Машиностроение, 1980.
9 .Справочник по машинной графике в проектировании/ .Е.Михайленко, В.А Анпилогова, Л.А.Кириевский и др.: Под ред. В.Е.Михайленко. А.А.Лященко. Киев: Будівельник, 1984 . 184 с.
10. Фоли Дж., вэн Дэм А. Основы интерактивной машинной графики: В 2-х книгах. Пер. с англ. М.: Мир, 1985.
11.. Шикин Е.В., Боресков А.В., Зайцев А.А. Начала компьютерной графики. - М .: Диалог-МИФИ, 1993 . 138 с.