О. Н. Романюк, О. В. Романюк, Р. Ю. Чехместрук

Міністерство освіти і науки України
Вінницький національний технічний університет

О. Н. Романюк, О. В. Романюк, Р. Ю. Чехместрук

Комп’ютерна графіка

Електронний навчальний посібник

Вінниця
ВНТУ
2023

УДК 004.92
Р69

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (Протокол № 5 від 29.12.2022 р.)

Рецензенти:

С. **В**. **Павлов**, доктор технічних наук, професор ВНТУ
А. В. Пукас, доктор технічних наук доцент, ЗНУ
Л. В. Крупельницький, кандидат технічних наук доцент, ВНТУ

Романюк, О. Н.

Р69 «Комп'ютерна графіка» : електронний навч. посіб. [Електронний ресурс] / О. Н. Романюк, О. В. Романюк, Р. Ю. Чехместрук. – Вінниця : ВНТУ, 2023. – 147 с.

Викладено алгоритмічні основи дво- та тривимірної комп'ютерної графіки, матеріали з базового програмного забезпечення, методів і засобів побудови інтерактивних графічних систем. Наведено приклади розв'язання типових задач.

Посібник розраховано на студентів бакалаврського напрямку спеціальності 121 – «Інженерія програмного забезпечення».

УДК 004.92

ЗМІСТ

ВСТУП

Бурхливий розвиток засобів обробки інформації, підвищення рівня автоматизації процесів виробництва і керування приводить до зростання ролі людського чинника. Для ефективного розв'язання задач оператору потрібно подати великий обсяг інформації в зручному для нього вигляді з заданою точністю в реальному масштабі часу.

У цих умовах однією з основних проблем стає організація ефективної інформаційної взаємодії людини з ЕОМ. Найбільш ємне та наочне подання великих обсягів даних забезпечує графічна форма, яка дозволяє провести візуальне оцінення результатів обчислення, внести необхідні корективи, відібрати з поданого матеріалу дані для наступної машинної обробки. Комп'ютерна графіка дозволяє суттєво збільшити пропускну спроможність інформаційного каналу, через який здійснюється двосторонній зв'язок користувача і комп'ютера. Роль і значення графічного подання результатів обчислень в промисловості та науково-дослідній практиці неперервно зростає.

Комп'ютерна графіка характеризує новий етап застосування комп'ютерів для обробки інформації та забезпечує не тільки підвищення наочності отриманих результатів, але й можливості вирішення принципово нових задач, як, наприклад, геометричне моделювання, дизайн, мультиплікація, автоматизація проєктувальних робіт.

Роль машинної графіки (МГ) як однієї з основних підсистем САПР значна, тому що тільки вона дозволяє в умовах сучасного рівня розвитку обчислювальної техніки реалізувати найбільш прийнятну для проєктувальника технологію автоматизованого проєктування. Це досягається завдяки забезпеченню звичної для нього графічної форми спілкування з системою як при введенні завдань в ЕОМ, так і при оцінюванні результатів автоматизованого проєктування. Саме внесення засобів МГ у САПР перетворює ЕОМ дійсно в інструмент проєктувальника, значно спрощує йому доступ до знань, закладених в ЕОМ у вигляді автоматичних проєктних процедур і довідкових даних, дозволяє автоматизувати виконання трудомістких креслярських і розрахунково-графічних робіт.

Широке застосування методів комп'ютерної графіки для вирішення інженерних, економічних, дослідних, конструкторських та дизайнерських задач зумовлює необхідність вивчення сучасними фахівцями основ комп'ютерної графіки.

В навчальному посібнику наведені основні відомості про дво- та тривимірну графіку, їх основні алгоритми та процедури, приклади розв'язання типових задач.

1 МЕТОДИ ТА АЛГОРИТМИ ФОРМУВАННЯ КОНТУРНИХ ЗОБРАЖЕНЬ

1.1 Елементи графічних зображень

Графічні зображення формуються з використанням примітивів.

Примітивами в машинній графіці прийнято вважати найменші, неподільні, з точки зору прикладних програм, графічні елементи, що використовуються як базові для побудови більш складних зображень. В апаратних засобах машинної графіки генерація графічних примітивів здійснюється спеціальними блоками, які попікселно формують зображення графічного елемента.

Графічними примітивами можна вважають такі елементи, для генерації яких в апаратних чи програмних засобах вводять команди, що їх ідентифікують.

Залежно від сфери застосування визначають набір примітивів, які найбільш доцільно застосувати для формування графічних зображень. Так, наприклад, в машинобудівних кресленнях найбільш поширеними є відрізки прямих, дуги кіл та алфавітно-цифрові символи.

В деяких застосуваннях як графічні примітиви застосовують цілу множину зв’язаних між собою графічних елементів, які визначають об’єкт для ідентифікації.

Примітиви поділяються на геометричн*і* (точки, відрізки прямих, ламані, дуги кривих, частини поверхонь); текстові і символьні (маркери).

Розрізняють апаратний, програмний та програмно-апаратний рівні формування примітивів.

Фізичним примітивом називають графічний елемент, для генерації якого в графічному пристрої є відповідний апаратний блок. В більшості систем машинної графіки апаратно реалізують такі примітиви, як точка, відрізок прямої, ламана лінія, рядок тексту, дуга та ін.

Логічним примітивом називають графічний елемент, що є елементарним об’єктом конкретної програми. Прикладами логічних примітивів можуть служити найпростіші геометричні фігури: трикутник, квадрат, багатокутник, паралелепіпед, конус, куля, циліндр, частина довільної поверхні та ін.

При програмно-апаратній реалізації примітивів на програмному рівні виконуються підготовчі розрахунки, після яких керування передається відповідному апаратному вузлу, що завершує генерацію потрібного графічного елемента. Так, наприклад, при заповненні ділянки, обмеженої

полігоном, програмним шляхом визначають межі контуру, а апаратним – заповнення між ними.

Примітив задається:

- параметрами, які визначають його форму, розміри і місце розташування;
- візуальними властивостями, які визначають його видимість, колір, яскравість, динамічні властивості, тип і товщину ліній;
- статусом, який визначає відношення до таких різних операцій, як вилучення, перетворення, вказання світловим пером.
- режимом занесення у відеопам'ять (режими заміщення, накладання, зворотне читання та ін.).

Графічна система містить засоби для об'єднання примітивів. Наявність структурованих даних дозволяє отримати з обмеженого набору базових елементів достатньо велику кількість видів та проєкцій зображень. Спрощується заміна, поворот та переміщення як окремих фрагментів, так і всього зображення. Структурні зв'язки між графічними даними значно полегшують процес пошуку та ідентифікації інформації, а також формування динамічних зображень. Останнє пояснюється тим, що при формуванні зображень, що відтворюють рух, на екрані з'являється тільки декілька нових елементів, а повна зміна відбудеться через деякий інтервал часу, який визначається швидкістю руху зображення та розмірами поля виведення. Тому імітацію руху здійснюють шляхом формування нових фрагментів при зберіганні статичних фрагментів без змін.

Розрізняють такі структурні одиниці, як примітив, графічний елемент, графічний об'єкт, графічний сегмент.

Графічним елементом називається впорядкована сукупність примітивів одного й того же типу. Зрозуміло, що графічний елемент може складатись всього з одного примітива.

Графічним об'єктом називається сукупність примітивів, які мають однакові візуальні властивості та статус і ідентифікуються одним іменем. При формуванні графічного об'єкта відпадає потреба у зміні режимів роботи графічного контролера (апаратних блоків), які відповідають за візуальні властивості.

Оскільки зображення може складатись з примітивів різного типу, то вводять поняття сегмента, який визначається як сукупність графічних елементів. Рівню сегмента в мовах програмування відповідає така конструкція, як процедура.

Виконуючи групування примітивів в пойменовані сегменти, програміст може селективно змінювати окремі частини повного зображення шляхом вилучення чи модифікації сегмента.

Графічні елементи є синтаксичними об'єктами (опис графічних даних), в той час, як сегменти – семантичними (зв'язок графічних даних), наприклад, будівля, кресленики.

Контрольні питання

1. Які структурні елементи зображень можна вважати графічними примітивами?
2. Чим відрізняються логічні примітиви від фізичних?
3. Якими атрибутами задаються графічні примітиви?
4. Чим зумовлено введення структурних одиниць типу графічний об’єкт, графічний сегмент?

1.2 Методи та алгоритми формування векторів

Відрізки прямих у сукупності графічних примітивів мають найбільшу питому вагу. В зв’язку з цим при розробці графічних пристроїв алгоритмам лінійної інтерполяції приділяють особливу увагу.

При виборі алгоритму основними критеріями є швидкодія формування крокової траєкторії, похибка інтерполювання, обчислювальна складність, тип формування крокової траєкторії (програмний, апаратний, програмно-апаратний).

Серед способів задання відрізків прямих в машинній графіці найбільшого поширення набули такі:

а) координатами початкової та кінцевої точок (рис. 1.1, а);

б) приростами координат (рис. 1.1, б).

В другому способі вектор задається початковою точкою та приростами ΔX та $\Delta У$ вектора, що визначають кінцеву точку.

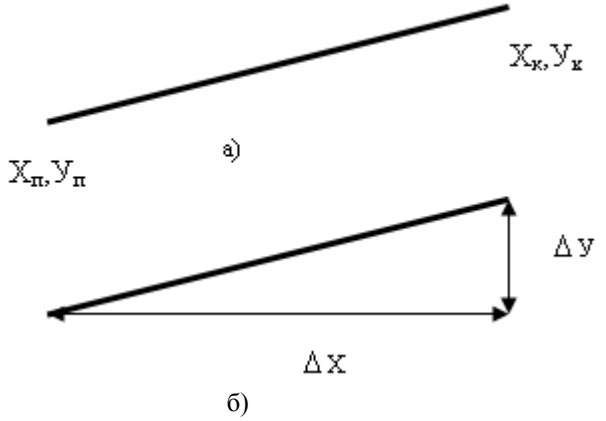


Рисунок 1.1 – Способи задання відрізків прямих

Залежно від типу формованих крокових переміщень на дискретній прямокутній сітці розрізняють алгоритми з чотири- та восьмивекторною направленістю крокових приростів. В перших алгоритмах використовують спільні кроки по веденій та ведучій координатах, а в других – кроки.

Алгоритми з чотиривекторною направленістю крокових приростів забезпечують похибку інтерполювання, яка не перевершує кроку дискретизації, а з восьмивекторною направленістю – вдвічі меншу.

Більшість методів лінійного інтерполювання орієнтовано на формування траєкторії в одному з квадрантів прямокутної системи координат. При формуванні траєкторії в будь-якому квадранті використовують спеціальні прийоми перемикання знаків змінних.

Лінійне інтерполювання містить два етапи: цикл підготовки і цикл інтерполювання. У циклі підготовки визначають мажоритарність відрізка прямої, а також його орієнтацію відносно координатних осей, що дозволяє встановити, в якому з октантів розміщено заданий відрізок прямої. У циклі інтерполювання за алгоритмічними залежностями знаходять значення крокових приростів, здійснюють їх видачу і аналізують закінчення формування траєкторії.

Серед методів формування відрізків прямих найбільшого поширення знайшли: метод симетричного інтегратора, «прямий» метод, метод цифрового диференціального аналізатора, метод оцінювальної функції.

Згідно з «прямим» методом координати точок траєкторії знаходять з відомого рівняння прямої

$$f(x) = f(x_1) + \frac{f(x_2) - f(x_1)}{x_2 - x_1}(x - x_1),$$

де $f(x_1)$, $f(x_2)$ – значення функції в базових точках x_1, x_2.

Підставляючи в наведене рівняння х знаходять значення функції.

Необхідність виконання «довгих» операцій, що визначає швидкодію формування векторів, суттєво обмежує застосування методу.

Оскільки реалізація операції ділення вимагає значного часу, то при використанні програмних засобів операцію ділення заміняють другими іншими діями.

При використанні методу цифрового диференціального аналізатора (ЦДА) відношення $P = \Delta У/\Delta X$ знаходять в циклі підготовки, а операцію множення на Х заміняють накопичувальним підсумовуванням. Очевидно, що при $\Delta X \geq \Delta У$ $P \leq 0$. Якщо в результаті накопичувального підсумовування операнда $\Delta У/\Delta X$ має місце сигнал переповнення для цілої частини числа, то до поточного значення У додають одиницю. Поточне значення Х збільшують на одиницю в кожному такті.

При $\Delta X < \Delta У$ знаходять відношення $\Delta X/\Delta У$ і виконують раніше перераховані дії, але вже зі зміною координат.

Похибка інтерполяції при реалізації методу може досягати кроку дискретизації та завжди одного знака. Похибку можна зменшити, якщо змістити на 0,5 рівень відліку.

В методі цифрового диференціального аналізатора (ЦДА) використовується також параметричне задання відрізка прямої, тому такі лінійні інтерполятори називають параметричними.

Розглянемо реалізацію методу ЦДА з використанням за базову мікрооперацію мікрооперацію лічби. За основний реалізувальний вузол в даному випадку застосовують двійковий помножувач, назва якого визначається виконанням останнім функції вигляду

$$f_{вих} = f_{вх(КК/2^n)},$$

де КК – значення керувального коду,

n – розрядність помножувача,

$f_{вх.}$, $f_{вих}$ – значення, відповідно, вхідної та вихідної частот опорної імпульсної послідовності.

Спрощена функціональна схема двійкового помножувача наведена на рис. 1.2. Двійковий помножувач ДП забезпечує перетворення керувального коду КК в кількість імпульсів, які формуються на його виході за $T = 2^n$ тактів лічби.

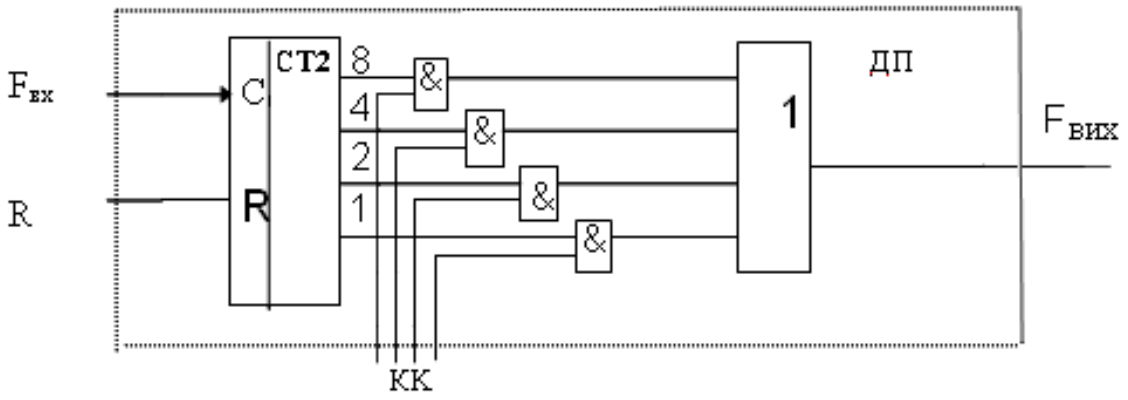


Рисунок 1.2 – Функціональна схема двійкового помножувача

Двійковий помножувач містить двійковий лічильник СТ2 та логічну схему, до складу якої входять елементи І та елемент АБО. За цикл перерахунку лічильника СТ2 на його виходах будуть сформовані імпульсні послідовності (рис. 1.3), кількість імпульсів в яких дорівнює степеням двійок (утворюють двійкові розряди). Одиничні значення розрядів керувального коду КК дозволяють роботу відповідних елементів І, що зумовлює проходження на їх вихід імпульсних послідовностей, які генеруються на виходах лічильника. Елемент АБО забезпечує об’єднання імпульсних послідовностей у вихідний потік $F_{вих}$. За $T = 2^n$ тактів на виході двійкового помножувача буде сформовано КК імпульсів.

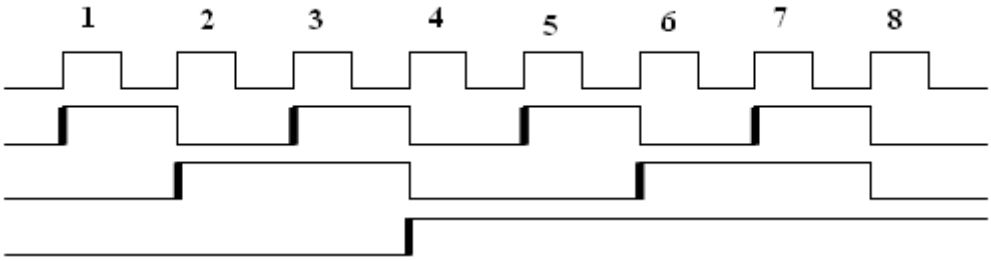


Рисунок 1.3 – Часова діаграма роботи
трирозрядного двійкового лічильника

В даному підході використовується допоміжна змінна – час T, який визначається циклом перерахунку двійкового помножувача.

Структурна схема параметричного лінійного інтерполятора наведена на рис. 1.4. Інтерполятор містить два регістри для зберігання приростів заданого вектора, які є керувальними кодами для ДП, а також два двійкових помножувача. Останні за 2^n тактів формують на своїх виходах кількість імпульсів, які рівні заданим приростам.

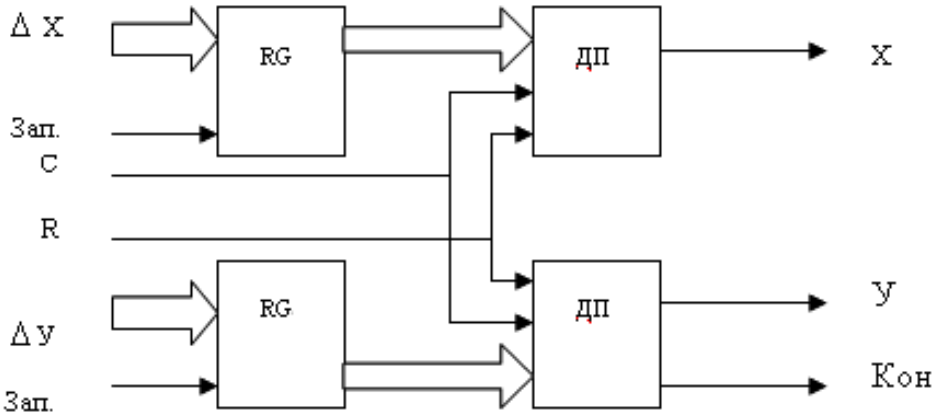


Рисунок 1.4 – Структурна схема параметричного
лінейного інтерполятора

На рис. 1.5 наведений приклад формування параметричним лінійним інтерполятором відрізка прямої з $\Delta X = 5$, $\Delta Y = 3$.

Наведений параметричний лінійний інтерполятор формує відрізки прямої, незалежно від їх довжини, за 2^n тактів, що не є доцільним. Час інтерполювання можна зменшити за рахунок нормалізації – одночасного збільшення вихідних приростів в задане число разів. При цьому цикл інтерполювання зменшують в стільки ж разів.

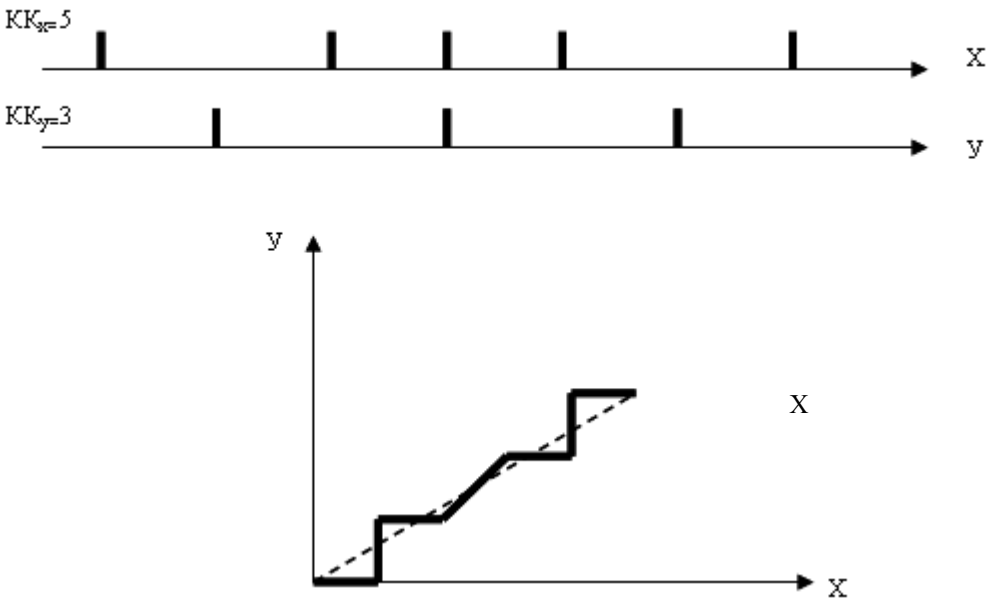


Рисунок 1.5 – Приклад формування відрізка прямої параметричним лінійним інтерполятором

Розглянутий підхід, орієнтований на апаратну реалізацію, має досить високу швидкодію, але сформовані траєкторії мають недостатню згладженість за рахунок можливого формування горизонтальних, вертикальних і діагональних кроків. Похибка інтерполювання визначається нерівномірністю надходження імпульсів на вихід ДУ, дискретним характером формування траєкторії й пропорційна розрядності інтерполятора.

Згідно з алгоритмом симетричного інтегратора обчислення інтеграла виконується приблизно, шляхом додавання (наприклад за методом Ейлера). При цьому обчислюються:

$$\Delta X(i*\Delta t)=\sum_{i}\Delta X/N=(\Delta X/N)i,\ i=0,1,...,N\ ,$$
$$\Delta Y(i*\Delta t)=\sum_{i}\Delta Y/N=(\Delta Y/N)i,\ i=0,1,...,N\ .$$

Якщо потрібно забезпечити режим креслення з постійною швидкістю (швидкість генерації вектора залежить від його довжини), розмір N має дорівнювати довжині L вектора. Проте і при іншому значенні N буде отримана правильна довжина вектора (після N кроків досягаються розміри ΔX та ΔУ незалежно від значення N). Отже, L можна приблизно виразити через N.

З іншого боку, значення N має бути вибрано таким, щоб кроки по ΔX (i*Δt) та ΔY(i*Δt) були менші розміру піксела, оскільки лише в цьому випадку вектор буде виглядати як суцільна лінія. Дана умова задовольняється, якщо розмір кроку менше однієї одиниці растра. Звідси можна одержати умову для визначення N:

$$\Delta X/N < 1 \text{ та } \Delta Y/N < 1,$$

або $$N > \max(|\Delta X|, |\Delta Y|).$$

Похибка наведеного методу полягає в тому, що прирости ΔX, ΔY потрібно ділити на N. Ділення можна замінити зсувом, якщо прийняти величину N рівною одному зі степенів двійки. У цьому випадку одержуємо умову

$$N = 2^{\log \max(|\Delta X|, |\Delta Y|)}.$$

Загальним прийомом для всіх інтерполяторів, що працюють за методом оцінювальної функції, є аналіз знаків оцінювальної функції, на основі якого роблять крок за тією чи іншою координатою з подальшим розрахунком її нового значення і корекцією відповідної координати поточної точки.

Нехай потрібно проінтерполювати відрізок прямої, яка задана приростами Δx і Δy по осях координат (рис. 1.6).

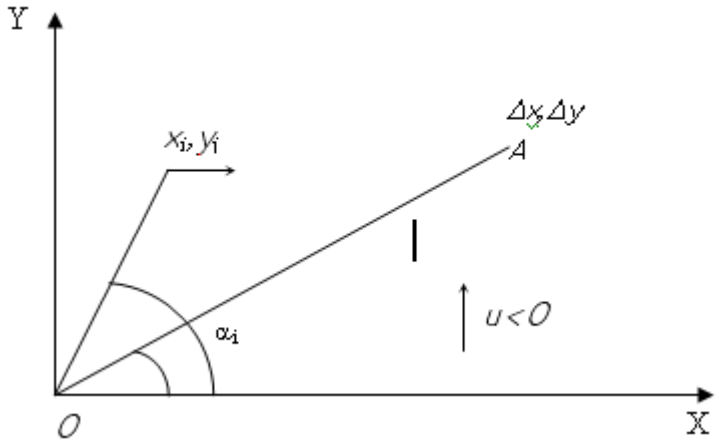


Рисунок 1.6 – Формування оцінювальної функції

Якщо точка траєкторії знаходиться вище від прямої ОА, то оцінювальна функція $U > 0$ і наступний крок потрібно виконувати по осі X; якщо точка знаходиться нижче від цієї прямої, то $U < 0$ і наступний крок потрібно виконувати по осі Y. Таким вимогам відповідає функція

$$U_i = tg\alpha_i - tg\alpha,$$

де $tg\alpha_i = y_i/x_i$;

$tg\alpha = \Delta y/\Delta x$;

x_i, y_i – поточні координати формованої траєкторії.

Тоді $$U_{ij} = y_i/x_i - \Delta y/\Delta x = (y_i \Delta x - x_i \Delta y)/x_i \Delta x.$$

Оскільки добуток $x_i\,\Delta x$ додатний і не впливає на знак U_{ij}, знаменником дробу можна знехтувати. Таким чином,

$$U_i = (\,y_i\,\Delta x - x_i\,\Delta y\,). \qquad (1.1)$$

Визначимо нове значення оцінювальної функції при виконанні кроку по осі *X*. У цьому випадку $x_{i+1} = x_i + 1$. Після підстановки в (1.1) значення для x_{i+1} одержимо

$$U_{i+1,j} = y_i\,\Delta x - \Delta y\,(x_i + 1) = y_i\,\Delta x - \Delta y x_i - \Delta y = U_{ij} + \Delta x.$$

Після кроку по осі *y* при $U < 0$ одержуємо нове значення $y_{i+1} = y_i + 1$. Тоді нове значення оцінювальної функції

$$U_{i\,,j+1} = \Delta x(y_i + 1) - \Delta y x_i = \Delta x y_i + \Delta x - \Delta y x_i\,, \text{ або } \quad U_{i\,,j+1} = U_{ij} + \Delta x.$$

Початкове значення оцінювальної функції беруть таким, що дорівнює нулю.

Наведений алгоритм має чотиривекторну орієнтацію крокових приростів і забезпечує похибку, не більшу від кроку інтерполювання.

Похибку інтерполювання можна зменшити, якщо використовувати діагональні крокові прирости. Такий тип приросту трактують як одночасне горизонтальне та вертикальне переміщення. Враховуючи вказане після такого кроку нове значення оцінювальної функції визначається як

$$U_{i+1\,,j+1} = U_{ij} + \Delta x - \Delta y.$$

Ненульове початкове значення оцінювальної функції дозволяє відсиметрувати похибки інтерполювання, що зменшує її вдвічі. На рис. 1.7, а наведена невідсиметрова крокова траєкторія.

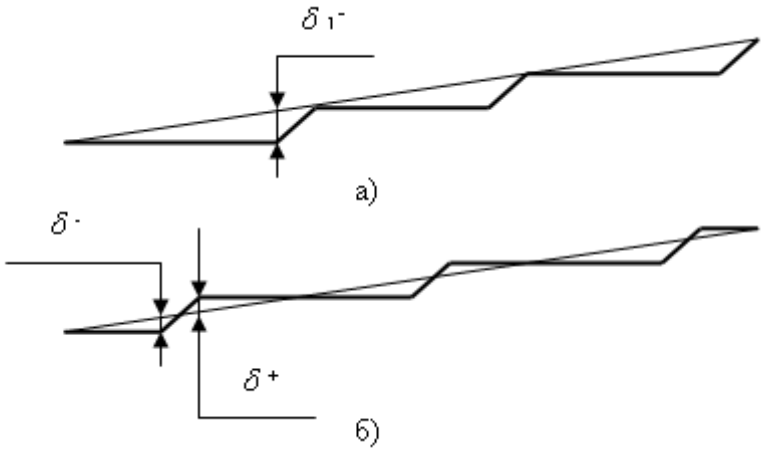


Рисунок 1.7 – Зменшення похибки інтерполювання
за рахунок симетрування

За рахунок розміщення діагонального крокового приросту посередині цифрового сегмента похибка інтерполювання δ_1^- замінюється на дві: δ^+ та δ^- (рис. 1.7, б), які значно менші від вихідної.

З формули (1.1) видно, що значення оцінювальної функції визначає похибку інтерполювання. Вказане використовується в алгоритмі Томпсона, згідно з яким визначається оцінювальна функція для двох можливих крокових переміщень.

Дійсний крок виконується в тому напрямку, де значення оцінювальної функції менше.

Алгоритм Томпсона забезпечує максимальну точність інтерполювання, однак потребує визначення відразу двох оцінювальних функцій, що, в свою чергу, позначається на його обчислювальній складності.

Згідно з відомим алгоритмом Брезенхема значення крокових приростів визначаються за знаком оцінювальної функції, яку розраховують відповідно до виразів:

$$\begin{cases} U_0 = 2N - M \\ U_i + 2(N-M) \text{ , якщо } U_i \geq 0 \\ U_{i+1} = U_i + 2N \text{ , якщо } U_i < 0 \end{cases}$$

де N, M – більше і менше значення приростів відрізка.

Цикл інтерполювання закінчується після видачі M крокових приростів. Якщо $U_i < 0$, то виконується крок по ведучій координаті, а при $U_i \geq 0$ – комбінований крок по обох координатах.

Алгоритм Брезенхема забезпечує мінімальну похибку інтерполювання, що дорівнює половині кроку дискретизації, але вимагає збільшення початкових операндів вдвічі. Це в окремих випадках призводить до необхідності роботи зі словами, які перевищують розрядну сітку процесора. Цей недолік ліквідовано в алгоритмі Пєтуха – Обідника.

Значення оцінювальної функції в цьому випадку розраховується за формулами:

$$\begin{cases} U_0 = \lfloor M/2 \rfloor, \\ U_i + (M - N), \text{ якщо } U_i < 0, \\ U_{i+1} = U_i + N, \text{ якщо } U_i \geq 0. \end{cases}$$

Якщо $U_i >= 0$, то виконується крокове переміщення по ведучій координаті. В іншому випадку виконується діагональне переміщення. Аналіз кінця інтерполювання не відрізняється від вказаної процедури алгоритму Брезенхема.

Інтерполювальні пристрої формують адреси точок траєкторії в дискретному координатному просторі. На рис. 1.8 наведена функціональна схема підключення інтерполятора.

Адреси точок траєкторії формують два координатних лічильника. Початкова точка $X_{поч.}$, $У_{поч.}$ заноситься у відповідні координатні лічильники в циклі підготовки і визначає їх вихідний стан.

Вказана мікрооперація отримала назву позиціювання. При формуванні інтерполятором крокової траєкторії стан координатних лічильників змінюється інкрементно і визначає адресу комірки відеопам'яті.

Для кожної точки траєкторії визначають код кольору, який виставляється на шину даних відеопам'яті.

Для деяких режимів запис в відеопам'ять блокується.

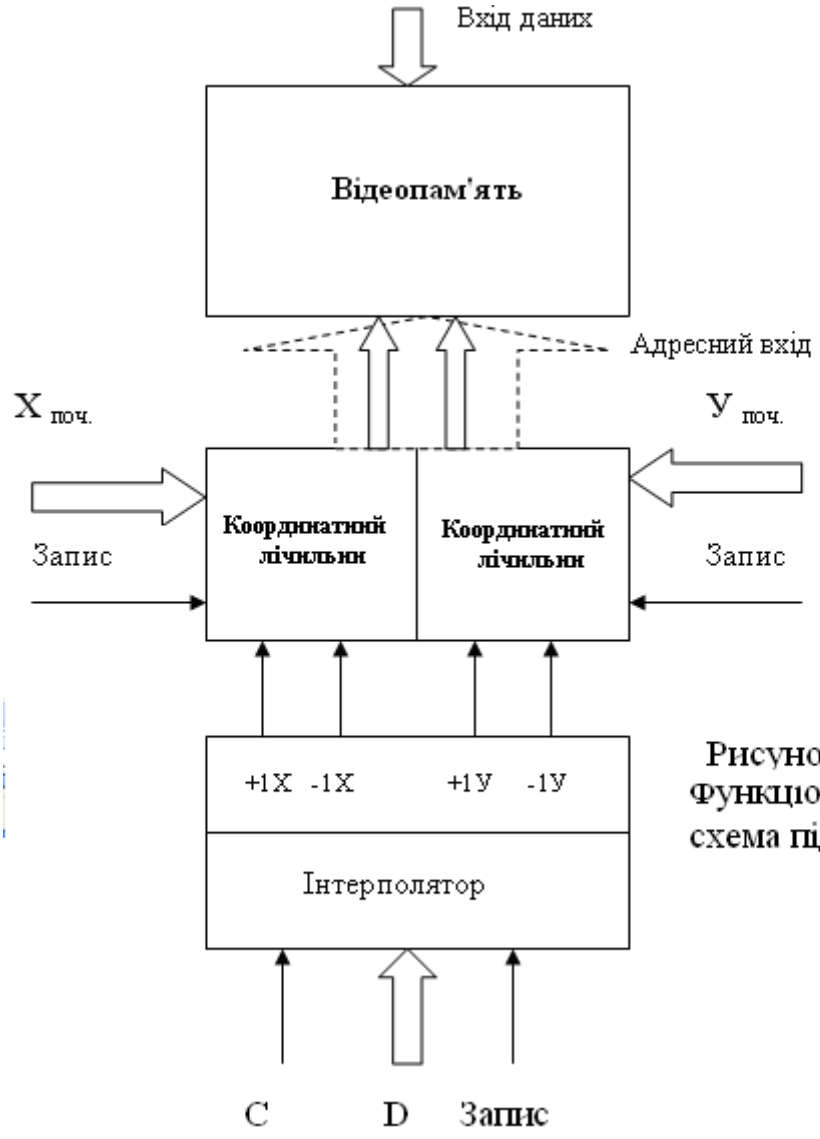


Рисунок 1.8 – Функціональна схема підключення

Контрольні питання

1. В чому полягають суттєві недоліки прямого методу інтерполювання?
2. За який час формується відрізок прямої параметричним лінійним інтерполятором ?
3. Дайте порівняльну характеристику точності методів лінійного інтерполювання.
4. Які базові операції виконуються при реалізації наведених методів лінійного інтерполювання?
5. Виконайте порівняльний аналіз відомих алгоритмів лінійного інтерполювання за методом оцінювальної функції.
6. Яким чином формують різні типи векторів – суцільні, штрихові, штрихпунктирні?

1.3 Методи та алгоритми формування кривих другого порядку

1.3.1 Коло

Дуги кіл, як і відрізки прямих, відносять до найбільш поширених графічних примітивів. Існує декілька методів задання вихідних даних для кругового інтерполювання.

Однією з можливих форм опису кола є задання її в полярній системі координат

$$X = X_c + R \cos Q,$$
$$Y = Y_c + R \sin Q,$$

де X_c, Y_c – координати центра кола,

R – радіус кола, Q – полярний кут.

Найчастіше вказують напрямок руху (за чи проти годинниковою стрілкою), прирости координат початкової та кінцевої точки дуги кола відносно його центру, а також координати початкової точки дуги в екранній системі координат (рис. 1.9).

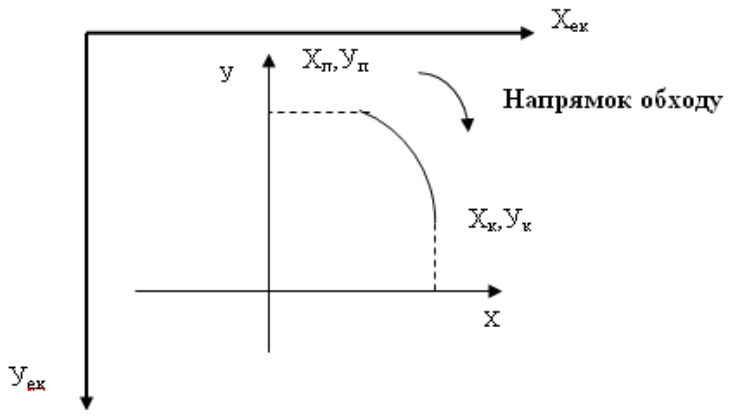


Рисунок 1.9 – Вихідне задання параметрів для колового інтерполювання

В процесі реалізації колового інтерполювання потрібно враховувати таке:

1. Коло є симетричною фігурою, що дозволяє суттєво скоротити час обчислення, оскільки для отримання всіх точок кола досить сформувати їх лише для одного октанта;
2. Потрібно проводити аналіз переходу через межі октантів, оскільки при цьому змінюється напрямок крокових приростів (рис. 1.10);
3. За рахунок похибки інтерполювання, а також округлень при визначені початкової та кінцевої точок дуги існує ймовірність непотрапляння в кінцеву точку дуги, що зумовлює необхідність введення в обчислювальний процес режиму доведення в кінцеву точку дуги.

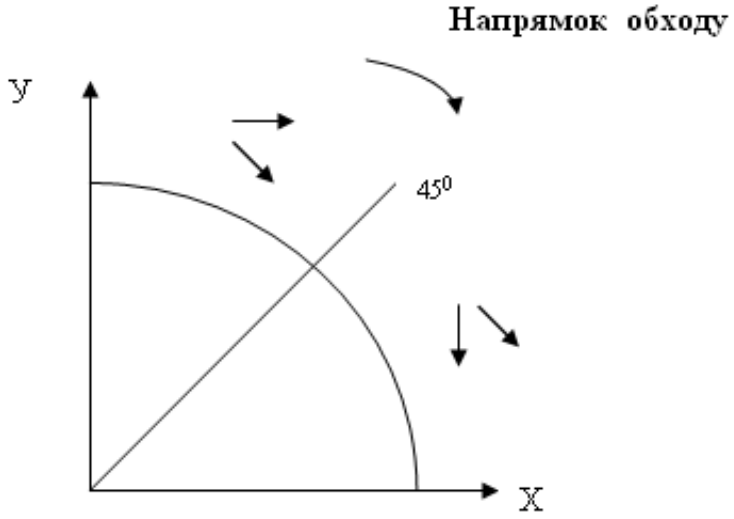


Рисунок 1.10 – Типи крокових приростів

Розглянемо процедуру формування кола в полярній системі координат.

Оскільки для обчислення координат точок кола X, Y потрібно задавати дискретні значення полярного кута, то виникає проблема раціонального обрання кроку приросту цього кута – ΔQ. При малому кроці час побудови кола збільшується, але забезпечується висока точність відтворення кола. При великому значенні ΔQ якість відтворення погіршується, але час побудови зменшується.

Виходячи з дискретної структури растра, можна стверджувати, що мінімальна відстань між сусідніми точками кола не може бути меншою 1. Отже, всі точки кола, без пропусків, будуть відтворені за умови, що крок приросту полярного кута обиратиметься з умови $\Delta Q = 1/R$.

Для скорочення часу формування кола з обчислювальної процедури бажано вилучити громіздке обчислення тригонометричних функцій cos Q та sin Q. Для цього обчислимо чергову $i + 1$ точку кола у вигляді

$$X_{i+1} = R \cos (Q + \Delta Q),$$

$$Y_{i+1} = R \sin (Q + \Delta Q).$$

Після заміни отримаємо:

$$X_{i+1} = X_c + (X_i - X_c) \cos \Delta Q - (Y_i - Y_c) \sin \Delta Q,$$

$$Y_{i+1} = Y_c + (X_i - X_c) \sin \Delta Q - (Y_i - Y_c) \cos \Delta Q.$$

Отримані рекурентні співвідношення для даного підходу забезпечують максимальну швидкість обчислень, оскільки значення $\sin \Delta Q$ та $\cos \Delta Q$ розраховуються лише один раз.

Згідно з прямим методом координати точок траєкторії кола визначаються з використанням виразу вигляду

$$X^2 + Y^2 = R^2,$$

де X, Y – поточні точки кола, а R – його радіус.

Для $X >= Y$ значення абсциси послідовно збільшують на одиницю, а ординату обчислюють за формулою $Y = \sqrt{R^2 - X^2}$, а для $X < Y$ для поточного Y обчислюють значення $X = \sqrt{R^2 - Y^2}$. Похибка інтерполювання визначається кількістю розрядів, відведених для обчислення, а також способом округлення.

Відносна складність обчислень суттєво обмежує застосування наведених методів.

Траєкторію кола можна сформувати шляхом її подання сукупністю хорд. Похибка інтерполювання визначається кількістю хорд, які були взяті для апроксимації.

Для деяких застосувань такий метод є прийнятним. Однак кола, побудовані за допомогою хорд, не є достатньо згладженими.

Для реалізації функції колового інтерполювання найчастіше використовують метод оцінювальної функції. Вигляд оцінювальної функції вибирають таким чином, щоб всередині та за колом вона мала протилежні знаки, а на самому колі – нульове. Таким вимогам відповідає функція вигляду

$$U_{ij} = X^2 + Y^2 - R^2.$$

Для $U \geq 0$ (рис. 1.11) виконують інтерполяційний крок вздовж осі Y, а для $U < 0$ – вздовж осі X. Після кроку по осі X нове значення оцінювальної функції знаходять, підставляючи в формулу для U_{ij} замість X_i величину $X_{i+1} = X_i + 1$. Тоді

$$U_{i+1,j} = (X_i + 1)^2 + Y^2_i - R^2 = X^2 + Y^2 - R^2 + 2X_i + 1 = U_{ij} + 2X_i + 1.$$

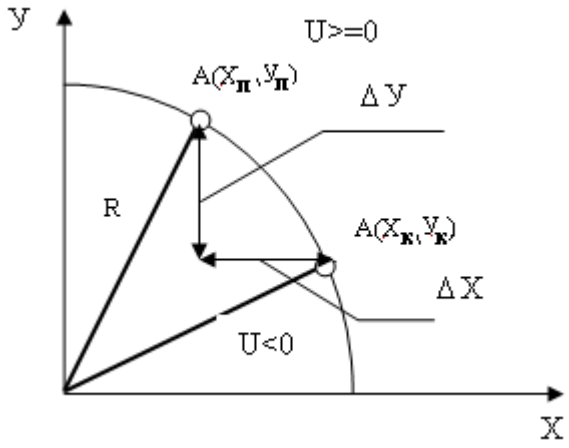


Рисунок 1.11 – Принцип формування оцінювальної функції

Після кроку по осі Y нове значення оцінювальної функції можна визначити, підставивши у формулу для U_{ij} замість Y_i величину $Y_{i+1} = Y_i - 1$. Тоді значення оцінювальної функції після кроку по осі Y буде дорівнювати

$$U_{i+1,j} = X_i + (Y_i - 1) - R^2 = U_{ij} - 2Y_i + 1.$$

Таким чином, після кроку по осі X до значення оцінювальної функції потрібно додати величину $2X_i + 1$, а після кроку по осі Y – величину $2Y_i + 1$. Таким чином, у будь-якому випадку додається одиниця і прямий чи доповняльний код величин $2X_i$, $2Y_i$. Після цього потрібно скорегувати значення X_i, Y_i для одержання $X_{i+1} = X_i + 1$, $Y_{i+1} = Y_i - 1$.

При реалізації алгоритму з восьмивекторною орієнтацією крокових приростів діагональний крок формують шляхом одночасного елементарного переміщення по обох координатах. В цьому випадку нове значення оцінювальної функції обчислюють за формулою

$$U_{i+1,j} = U_{ij} + 2X_i + 1 - 2Y_i + 1 = U_{ij} + 2(X_i - Y_i) + 2.$$

На практиці широкого поширення набув алгоритм колового інтерполювання співробітника фірми IBM Брезенхема. При формуванні кола в другому октанті за годинниковою стрілкою виконуються такі дії:

$$U =: 3 - 2R, \quad X := 0, \; Y := R.$$

Якщо $U < 0$, то $U := U + 4X + 6$, $X =: X + 1$. В протилежному випадку $U := U + 4(X_i - Y_i) + 10$, $X := X + 1$, $Y := Y - 1$.

Вказані дії виконують до формування кінцевої точки октанта.

З обчислювального процесу вилучені «довгі» операції, що обумовлюють високу продуктивність методу, а також можливість апаратної реалізації.

Розглянемо режим «доводки» в кінцеву точку дуги. Для першого квадранта можливі два випадки, які наведені на рис. 1.12.

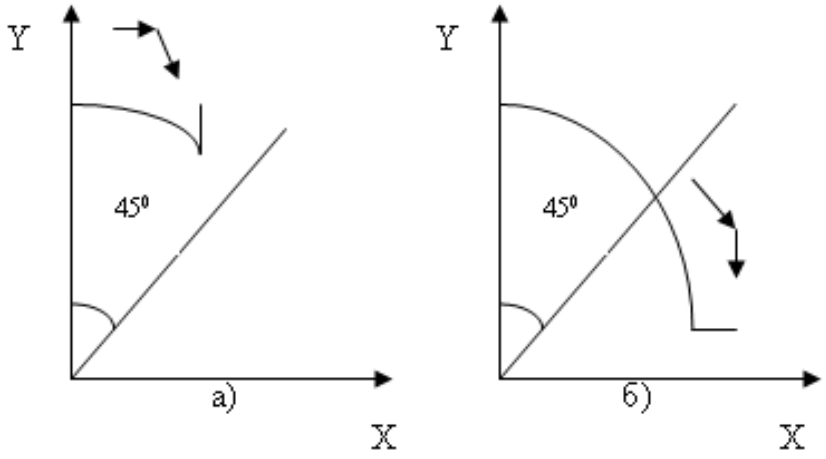


Рисунок 1.12 – Доведення в кінцеву точку дуги

У другому октанті (рис. 1.12, а) крокова траєкторія формується горизонтальними та діагональними (комбінованими) кроковими приростами.

Враховуючи, що діагональний приріст містить переміщення по осі X, то можна констатувати, що всі елементарні кроки до точки $X = Y$ містять переміщення по осі абсцис. Звідки випливає, що при формуванні траєкторії гарантовано буде досягнуто координату X_k, а доведення буде потрібне до координати Y_k.

Аналогічно можна показати, що при формуванні дуги в першому октанті доведенню підлягає координата X_k.

Контрольні питання

1. Вкажіть характерні особливості колового інтерполювання.
2. Які форми задання дуг кіл вам відомі?
3. Наведіть порівняльну характеристику алгоритмів формування кіл.
4. Чим зумовлена необхідність доведення в кінцеву точку дуги?
5. Поясніть, чому формули для розрахунку точок траєкторії кола різні при різних напрямках обходу.
6. Скільки елементарних кроків треба виконати для формування чверті кола?

1.3.2 Парабола

Параболу відносять до поширених графічних примітивів.

Для параболічного інтерполятора $y = x^2$ оцінювальна функція має вигляд $u_{ij} = x_i^2 - y_i$. Вона дозволяє відобразити залежність, зображену на рис. 1.13.

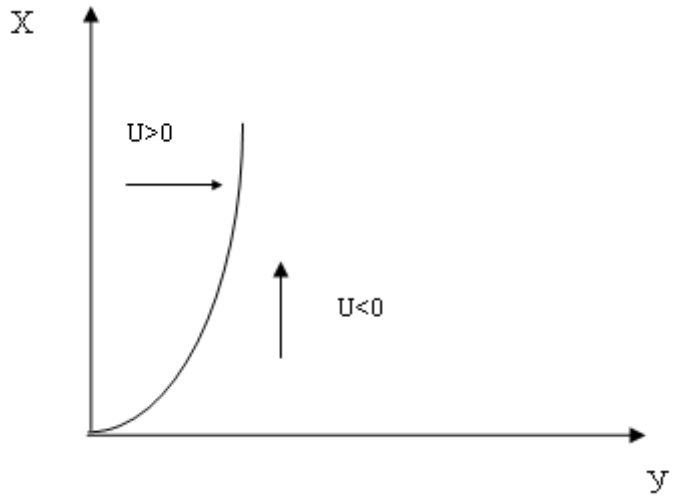


Рисунок 1.13 – Принцип параболічного інтерполювання
за методом оцінювальної функції

Для $U \geq 0$ виконують крок по осі X, а для $U < 0$ – по осі Y. Після кроку по осі X нове значення оцінювальної функції розраховують згідно з виразом

$$u_{i+1,j} = (x_i + 1)^2 - y_i = u_{ij} + 2x_i + 1.$$

При виконанні кроку вздовж осі Y значення оцінювальної функції визначають як

$$u_{i,j+1} = x_i^2 - (y_i + 1) = u_{ij} - 1.$$

Таким чином, параболічний інтерполятор – це коловий інтерполятор з одним заблокованим блоком для розрахунку оцінювальної функції.

Контрольні питання

1. Чим відрізняється параболічне інтерполювання від колового?
2. Як вибрати початкове значення оцінювальної функції для параболічного інтерполювання?

1.3.3 Гіпербола

Гіперболічний інтерполятор реалізує функцію вигляду $A = XY$.
Оцінювальна функція для нього має вигляд

$$U_{ij} = X_i Y_i - A.$$

Для $U >= 0$ виконується крок по осі Y (рис. 1.14), а джля $U < 0$ – крок по осі X, причому $X_{i+1} = X_i + 1$, а $Y_{i+1} = Y_i - 1$.

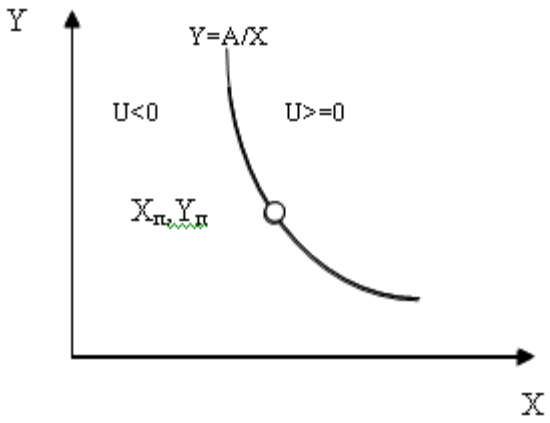


Рисунок 1.14 – Формування оцінювальної функції
при гіперболічній інтерполяції

Після кроку по осі X нове значення оцінювальної функції обчислюється як

$$U_{i+1,j} = U_{i,j} + Y_j,$$

а після кроку по осі Y –

$$U_{i,j+1} = U_{i,j} - X_i.$$

Якщо обхід гіперболи буде протилежним до розглянутого випадку, тобто $X_{i+1} = X_i - 1$, а $Y_{i+1} = Y_i + 1$, то корегування оцінювальної функції зміниться на протилежне.

При реалізації алгоритмів з восьмивекторною орієнтацією крокових приростів при діагональному переміщенні нове значення оцінювальної функції матиме вигляд

$$U_{i+1,j+1} = U_{i,j} + Y_j - X_i.$$

Контрольні питання

1. Назвіть сфери застосування гіперболічної інтерполяції.
2. Знайдіть значення оцінювальної функції для різних обходів гіперболи.

1.3.4 Еліпс

Рівняння еліпса в полярній системі координат має вигляд

$$X = a\cos Q,$$
$$Y = b\sin Q,$$

де a, b – розміри еліпса по осях X та Y відповідно.

Стосовно процедури формування еліпса справедливі усі положення, розглянуті при побудові кола, з урахуванням особливостей:

• опорним фрагментом є точка одного з квадрантів ;

• для еліпса довжина радіус-вектора $R_i = \sqrt{X_i^2 + Y_i^2}$ і крок приросту полярного кута $\Delta Q_i = 1/R_i$ є змінними та мають обчислюватися на кожному кроці.

Розглянемо формування еліпса за методом оцінювальної функції.

Рівняння еліпса має вигляд

$$X^2/a^2 + Y^2/b^2 = 1$$

або

$$X^2b^2 + Y^2a^2 = a^2b^2.$$

Оцінювальна функція $U_{ij} = X^2b^2 + Y^2a^2 - a^2b^2$ за еліпсом (рис. 1.15) має додатне значення, а всередині нього – від'ємне. Для першого квадранта для $U_{ij} >= 0$ виконується крок по осі Y і $Y_{i+1} = Y_i - 1$, а для $U_{ij} < 0$ – крок по осі X, при цьому $X_{i+1} = X_i + 1$.

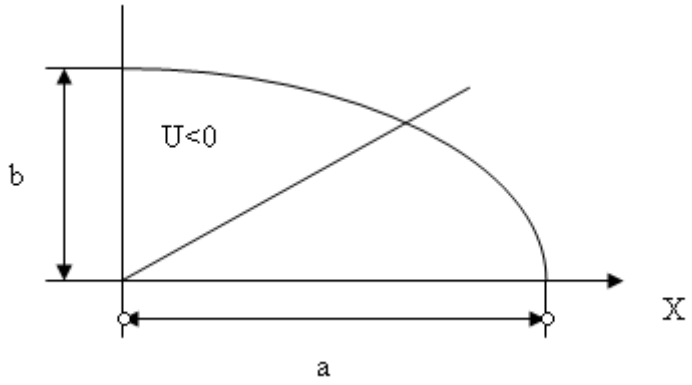


Рисунок 1.15 – Формування оцінювальної функції
при еліптичній інтерполяції

Після кроку по осі Y

$$U_{ij+1} = X^2{}_i\, b^2 + (Y_j - 1)^2\, a^2 - a^2b^2 = b^2X^2{}_i + a^2Y^2{}_j + 2a^2Y_j + a^2 - a^2b^2 = \\ = U_{ij} + 2a^2Y_j + a^2.$$

Аналогічно можна показати, що після виконання кроку по осі X нове значення оцінювальної функції буде мати вигляд

$$U_{i+1,j} = U_{ij} + 2b^2X_i + b^2.$$

Наведені формули показують, що після кожного кроку потрібно виконувати операцію множення, що суттєво обмежує швидкодію формування крокової траєкторії.

Контрольні запитання

1. Назвіть характерні особливості еліптичного інтерполювання.
2. Які методи застосовують для еліптичного інтерполювання?
3. Дайте порівняльну характеристику колового та еліптичного інтерполювання за методом оцінювальної функції.

1.4 Інтерполяція та апроксимація кривих довільного типу

1.4.1 Форми завдання кривих

В задачах машинного проєктування знайшли дві форми задання кривих: аналітична та параметрична.

Аналітична форма передбачає задання кривої у вигляді рівняння Y = F(x) з використанням звичайних однозначних функцій.

Форма більшості об'єктів в техніці не залежить від системи координат. Якщо потрібно відновити криву або поверхню за множиною окремих точок, отриманих в результаті вимірювань на моделі, то важливим фактором, який визначає форму об'єкта, буде співвідношення між самими цими точками, а не між точками і якою-небудь довільно вибраною системою координат. В багатьох застосуваннях необхідно, щоб вибір системи координат не впливав на форму. Крім того, форми інженерних об'єктів можуть мати вертикальні дотичні. Якщо така форма була б подана звичайною функцією вигляду $Y = F(x)$, то наявність вертикальних дотичних зробила б неможливою апроксимацію цієї форми багаточленами. Потрібно відмітити, що криві та поверхні в машинній графіці часто є неплоскими або замкнутими, що взагалі не дає можливість їх подати у вигляді функцій. Тому важливу роль відіграє подання форми в параметричному вигляді, коли крива на площині подана не функцією вигляду Y = F(x), а парою функцій $X = X(t)$, $Y = Y(t)$ від параметра t.

Параметрична форма дозволяє ліквідувати вказані недоліки. Крім того, дві функції $X = X(t)$, $Y = Y(t)$ можуть бути використані як функції управління для системи відхилення променями електронно-променевої трубки або сервосистеми графопобудовника.

Для формування кривих використовують методи інтерполювання та апроксимації. Задача інтерполяції зводиться до знаходження деякої аналітичної функції, яка точно проходить через задані точки. В багатьох випадках сформована за базовими точками крива є недостатньо згладженою, наприклад, має хвилястість. Поняття точності для більшості задач інтерактивного конструювання об'єктів не має сенсу. Для них особливо важливою є форма об'єкта, його згладженість. В цьому випадку застосовують методи апроксимації, під якими розуміють знаходження за сукупністю базових точок такої функції, яка проходить безпосередньо близько від заданих точок. Задача апроксимації виникає при заміні кривої,

заданої рівняннями функцій складної природи (наприклад, з точки зору швидкості розрахунку її значень і похідних, інтегрування, диференціювання) іншою кривою, близькою до заданої, рівняння якої більш прості.

Криву можна побудувати шляхом:

- інтерполюванням або апроксимацією за точками;
- деформацією кривої (переміщення точки, зміна полінома);
- обчисленням еквідистанти до заданої кривої;
- формуванням розімкнутого або замкнутого контуру з відрізків або дуг кіл на площині;
- обчисленням конічних перетинів (еліпс, парабола тощо);
- обчисленням перетину поверхонь;
- сполученням кривих.

Контрольні питання

1. В яких випадках використовують аналітичне та параметричне завдання кривих?
2. Чим інтерполювання відрізняється від апроксимації?
3. Якими методами можна побудувати криву?

1.4.2 Інтерполяційні методи Лагранжа та Ньютона

До найбільш простих інтерполяційних поліномів відносять поліном Лагранжа.

Нехай задано $n + 1$ точок $(x_0,y_0),(x_1,y_1),...,(x_n,y_n)$ на координатній площині, причому $x_i < x_{i+1}, i = 0,n.$ Інтерполяційний поліном Лагранжа n-го степеня для даної множини точок має вигляд

$$y = \sum_{i=0}^{n} \left(\frac{(x - x_0)...(x - x_{i-1})(x - x_{i+1})...(x - x_n)}{(x_i - x_0)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n)} \right)_i y_i.$$

Візьмемо для прикладу $n = 1$. Тоді після підстановки в формулу маємо

$$y = \frac{x - x_1}{x_0 - x_1} y_0 + \frac{x - x_0}{x_1 - x_0} y_1.$$

У результаті отримано рівняння відрізка прямої лінії, яка з`єднує точки (x_0,y_0) і (x_1,y_0).

Поліном Лагранжа – це не єдиний багаточлен, який відповідає задачі інтерполювання. У дійсності існує нескінченно багато відповідних багаточленів, але багаточлен Лагранжа – єдиний багаточлен степеня n, який є розв`язком задачі. Разом з цим інтерполяційний метод Лагранжа має істотний недолік, оскільки степінь багаточлена Лагранжа відповідає кількості вузлів

(мінус 1) і будь-яка спроба покращити точність апроксимації шляхом збільшення кількості вузлів викликає збільшення степеня полінома. Якщо степінь полінома більший 5, то на кривій з'являється «хвилястість», яка одержала назву ефекту Рунге–Мерей. Таким чином, неможливо досягти точкової збіжності кривої $P(x)$ до $f(x)$ шляхом збільшення кількості рівномірно розподілених вузлів.

Хвилястість звичайно не є припустимою. Один із можливих шляхів розв'язання цієї проблеми полягає в розбитті інтервалу $[x_0, x_n]$ на кілька підінтервалів і «склеюванні» кількох багаточленів Лагранжа низьких степенів, кожний з яких апроксимує задану функцію на одному з підінтервалів. При цьому можна досягти будь-якої точності, але ціною можливої недиференційовності об'єднаної функції в деяких вузлах.

При рівномірній дискретизації (Δx = const) зручно користуватися інтерполяційною формулою Ньютона. Щоб записати формулу Ньютона введемо поняття різниці функції.

Нехай задані значення аргументу $a,\ a + h,\ a + 2h,\ a + 3h,\ \ldots$ і відповідні значення функції. Випишемо значення аргументу та функції двома колонками. Потім кожне число другої колонки, починаючи з першого, віднімемо з наступного числа. Одержимо перші різниці функції і першу з них позначимо через $\Delta f(a)$. Знайдені різниці записуємо в третю колонку. Аналогічно першим різницям складаємо другі. Першу з других різниць позначимо символом $\Delta^2 f(a)$. Аналогічно складаються треті різниці і т. д. Інтерполяційна формула Ньютона має вигляд

$$f(x) = f(a) + \frac{x-a}{h}\Delta f(a) + \frac{(x-a)(x-a-h)}{2!\,h^2}\Delta^2 f(a) + \\ + \frac{(x-a)(x-a-h)(x-a-2h)}{3!\,h^3}\Delta^3 f(a) + \ldots$$

Зауважимо, що поліном Ньютона третього порядку, порівняно з поліном Ньютона другого порядку, дозволяє підвищити точність відновлення лише в 1,5 разу.

При введені нових вузлів наведена формула більш зручна для обчислення, ніж запис інтерполяційного багаточлена у формі Лагранжа, тому що додавання нових вузлів інтерполяції спричиняє обчислення тільки нових доданків, що додаються до тих, які були обчислені з меншим числом вузлів. При використанні форми Лагранжа в цій ситуації потрібно виконувати всі обчислення знову.

Контрольні питання

1. Який з інтерполяційних поліномів має найменший степінь?
2. В чому полягають недоліки методу Лагранжа?
3. В яких випадках зручно використовувати метод Ньютона?
4. Як уникнути ефекту Рунге – Мерей?

1.4.3 Апроксимація кривих методом Безьє

На практиці є низка задач, в яких потрібне не точне наближення, а згладжене формування фігури, що апроксимує вхідні дані, тобто коли властивості апроксимації в цілому важливіші точності наближення і коли вимоги, які висуваються до проєктованого об'єкта, не можуть бути достатнь о просто виражені математично.

Досить типова задача має місце в процесі проєктування автомобілів. Вона полягає в знаходженні математичного подання для рисунка чи глиняної моделі, що її запропонував дизайнер. В цьому випадку неможливо визначити «найкраще» наближення. Якість наближення залежить, головним чином, від думки дизайнера. Отже, логічно використовувати інтерактивний метод, що дозволяє користувачу експериментувати з різноманіттям форм, причому від нього не потребується ніяких знань про використані математичні методи. Проєктування таким способом значно полегшується, якщо є можливість керувати формою кривої за допомогою зміни невеликої кількості параметрів, особливо, якщо ці параметри задавати в графічному вигляді.

Розроблені методи для авіаційної, автомобільної та суднобудівельної промисловості, найбільш поширеним серед яких є метод Безьє, що використовує апроксимацію багаточленами Бернштейна.

Замість безпосереднього використання точок для задання багаточлена використовують множину точок-орієнтирів.

Якщо $(X_0,\ Y_0)$, $(X_1,\ Y_1)$, ..., $(X_m,\ Y_n)$ – вказані точки-орієнтири, то відповідний багаточлен Безьє визначається як

$$P_x(t) = \sum_{i=0}^{m} C_m^i t^i (1-t)^{m-i} X_i$$

$$P_y(t) = \sum_{i=0}^{m} C_m^i t^i (1-t)^{m-i} Y_i$$

де $$C_m^i = \frac{m!}{i!(m-i)!}\ .$$

Остання формула достатньо складна, тому на практиці використовують вираз вигляду $C_m^i = C_{m-1}^i + C_{m-1}^{i-1}$.

Багаточлени Безьє мають такі властивості:

$P(0) = P_0, \quad P(1) = P_m$

при $t = \overline{0,1}$.

Вказане визначає те, що крива Безьє проходить через першу та останню точки-орієнтири.

Якщо використовувати диференціальне числення, то можна показати, що крива Безьє розміщена всередині опуклої оболонки множини точок-орієнтирів. Нахил дотичних векторів в крайніх точках кривої збігається з нахилом, відповідно, першої та останньої ланок ламаної Безьє

Багаточлени Безьє задовольняють теорему Вейєрштрасса, тобто вони рівномірно збігаються до апроксимувальної функції зі зростанням m. Не дивлячись на ці позитивні якості, багаточлени Безьє ніколи не використовувались широко для побудови апроксимацій з мінімальною нормою відхилення. Причина у тому, що багаточлени Безьє дуже повільно збігаються у рівномірній нормі. Однак для багаточленів Безьє достатньо просто, порівняно з іншими методами, написати програми.

В процесі інтерактивного конструювання не ставиться задача точності, а потребується засіб керування формою.

Практичне конструювання за методом Безьє таке.

Спочатку конструктор вручну робить рисунок бажаної кривої. Потім він вказує на вершини ламаної кривої, яка є, насправді, першим наближенням. Наступний крок складається в переміщенні вершин таким чином, щоб поступово покращити наближення. Якщо потрібно, деякі вершини видаляються чи додаються нові.

Контрольні питання

1. Чим визначається степінь полінома Безьє?
2. Назвіть характерні особливості апроксимації Безьє.
3. Як практично здійснюється апроксимація кривих за методом Безьє?
4. Порівняйте обчислювальну складність інтерполювання за методом Лагранжа та апроксимацію за методом Безьє.

1.4.4 Сплайнова інтерполяція

Методи апроксимації функцій за допомогою сплайнів, запропоновані вперше в 40-х роках минулого століття, одержали значне поширення лише останнім часом.

Основний недолік інтерполяційних багаточленів як апарата наближення функцій, застосовуваного для відновлення дискретизованих сигналів, полягає в тому, що поводження цих багаточленів в околі якоїсь точки визначає їхнє поводження в цілому. Якщо досліджуваний сигнал на різних ділянках має різний характер, наприклад, на одній ділянці постійний, а потім круто зменшується або зростає і т. д., то використання інтерпо-ляційних багаточленів прийнятних результатів не дає. У таких випадках краще користуватися сплайнами.

Англійське слово spline означає «гнучка рейка». Таку рейку використовують як гнучке лекало при кресленні плоских кривих по опорних точках.

Основна ідея застосування сплайнів полягає в нижчевикладеному. Інтервал, на якому відновлюють функцію, розбивають на підінтервали (рис. 1.16), на кожному з який функцію задають поліномом достатньо низького степеня і забезпечують неперервність кривої в точках «склейки» шляхом прирівняння значень поліномів на межах підінтервалів. При цьому важливою умовою є також неперервність декількох похідних.

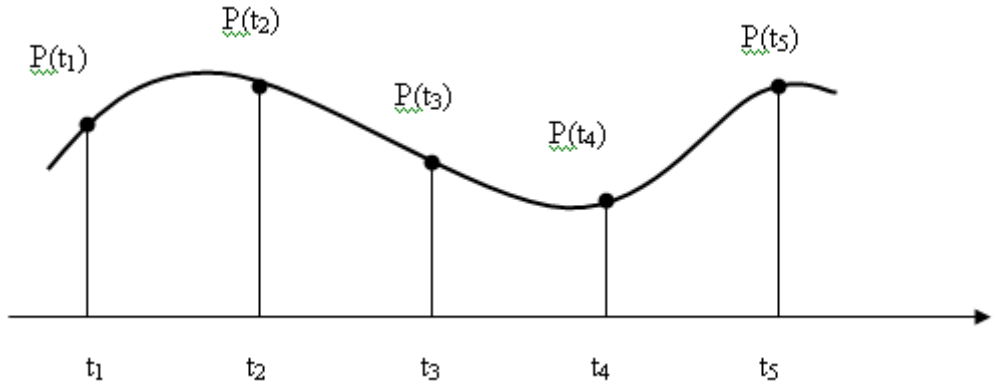


Рисунок 1.16 – Відновлення сигналу за допомогою сплайнів

Отже, сплайном P_n називають сукупність багаточленів P_{ni} степеня n, заданих на i-му кроці дискретизації, які задовольняють умову

$$P_{ni}(t_i) = P_{n(i\text{-}1)}(t_i),$$

тобто степеня n сплайн-функціями, складеними зі «шматочків» багаточленів даного степеня, що сполучені так, щоб функція, що утворилася, була неперервною і мала декілька неперервних похідних.

Таким чином, сплайн є кусково-поліноміальною функцією. Сплайн нульового степеня збігається зі східчасто-інтерпольованою функцією, а сплайн першого степеня – з лінійно-інтерпольованою. Потрібно зазначити, що сплайни другого і більш високих степенів не будуть збігатися з інтерполяційними поліномами відповідних степенів.

Відомо, що коли однорідний брус закріпити в двох довільних точках і надати його осі заданий нахил у цих точках, то форма кривої бруса описується поліномом третього степеня.

Припустимо, що кубічна парабола, задана в параметричній формі

$$P(t) = a_3t^3 + a_2t^2 + a_1t + a_0,$$

проходить через дві точки $P(t_1)$ і $P(t_2)$, у яких відомі значення похідних $P'(t_1)$ і $P'(t_2)$. Це означає, що задані чотири необхідних і достатніх умови для визначення чотирьох невідомих коефіцієнтів у наведеному вище виразі. Цей багаточлен іноді називають кубічним інтерполяційним багаточленом Ерміта.

Параметричне подання кривої має ту перевагу, що можна вибрати довільний діапазон зміни параметра. Для простоти обчислювального процесу будемо вважати, що параметр t у межах кожного сегмента змінюється в діапазоні від 0 до 1 ($0 \le t \le 1$).

Неважко помітити, що

$$P(0) = a_0, \quad P(1) = a_0 + a_1 + a_2 + a_3.$$

Визначимо похідну $P'(t)$.

$$P'(t) = 3a_3t^2 + 2a_2t + a_1.$$

Звідси випливає, що

$$P'(0) = a_1, \qquad P'(1) = 3a_3 + 2a_2 + a_1.$$

З наведених виразів легко визначити невідомі a_0, a_1, a_2, a_3.

$$a_0 = P(0), \quad a_1 = P'(0), \quad a_2 = 3\big[P(1) - P(0\ )\big] - 4P'(0) - P'(1),$$
$$a_3 = 2\big[P(0) - P(1)\big] + P'(0) + P'(1).$$

Розглянута процедура виконується для кожної пари заданих точок кривої. Так, визначивши кубічну параболу між точками $P(t_1)$ і $P(t_2)$ на проміжку t_2, t_3, для знаходження наступної такої дуги кривої на проміжку t_2, t_3 потрібно на межі інтервалу (точка t_2) прирівняти значення як самих функцій, так і їх перших похідних, тобто виконати «склеювання».

Знайдемо вираз для кубічної кривої $X(t)$ у формі Ерміта в матричній формі, якщо відомі кінцеві точки і дотичні вектора до кривої у цих точках.

Визначимо, що P_1, P_4 – точки, R_1, R_4 – дотичні.

$$X(0) = P_{1x} \qquad X(1) = P_{4x}$$
$$X(0) = R_{1x} \qquad X(1) = R_{4x}$$

$$X(t) = \begin{bmatrix} t^3, t^2, t, 1 \end{bmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

або

$$X(t) = \mathrm{T}\ C_x, \qquad (*)$$

де

$$\mathrm{T} = [t^3, t^2, t, 1],$$
$$\mathrm{C} = [a_0, a_1, a_2, a_3],$$

$$X(0) = P_{1x} = [0, 0, 0, 1]\ C_x,$$
$$X(1) = P_{4x} = [1, 1, 1, 1]\ C_x,$$
$$X'(\mathrm{t}) = [3t^2, 2t, 1, 0]\ C_x,$$
$$X'(0) = R_{1x} = [0, 0, 1, 0]\ C_x,$$
$$X'(1) = R_{4x} = [3, 2, 1, 0]\ C_x.$$

$$\begin{pmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{pmatrix} = \begin{pmatrix} 0, & 0, & 0, & 1 \\ 1, & 1, & 1, & 1 \\ 0, & 0, & 1, & 0, \\ 3, & 2, & 1, & 0 \end{pmatrix}$$

«Обертаючи» матрицю розміром 4×4, отримуємо

$$C_x = \begin{pmatrix} 2, & -2, & 1, & 1 \\ -3, & 3, & -2, & -1 \\ 0, & 0, & 1, & 0, \\ 1, & 0, & 0, & 0 \end{pmatrix} \begin{pmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{pmatrix} = \mathrm{M_h}\ \mathrm{G_{hx}}\ .$$

$\mathrm{M_h}$ називається матрицею Ерміта, $\mathrm{G_h}$ – геометричним вектором Ерміта. Підставивши C_x в (*),отримаємо

$$X(t) = \mathrm{TMG}.$$

Знайдемо добуток TM

$$\mathrm{TM} = (2t^3 - 3t^2 + 1)(-2t^3 + 3t^2)(t^3 - 2t^2 + t)(t^3 - t^2).$$

Перемножуючи цей вираз справа на $\mathrm{G_{hx}}$, отримаємо

$$X(t) = \mathrm{TM_hG_{hx}} = P_{1x}(2t^3 - 3t^2 + 1) + P_{4x}(-2t^3 + 3t^2) + R_{1x}(t^3 - 2t^2 + t) + R_{4x}(t^3 - t^2).$$

Чотири отриманих функції іноді називаються функціями спряженості.

За допомогою перших двох функцій визначаються точки P_1 і P_4, а за допомогою решти отримаємо згладжене об’єднання. Причому, чим більший дотичний вектор, тим крутіша сама крива.

До недоліків такого підходу варто віднести необхідність задання похідних у вузлах. Для їхнього обчислення використовують низку підходів.

Найпростіше використовувати замість похідних їхні наближені значення, отримані в результаті інших апроксимацій.

Наприклад, для кожного вузла за значеннями функцій у найближчих $2m$ вузлах будується багаточлен степеня $2m$+1, похідні якого потім використовуються для побудови ермітового сплайна.

Розроблено підхід, відповідно до якого похідні в проміжних точках знаходять шляхом розв’язання матричного рівняння вигляду

$$\begin{vmatrix} 4 & 1 & 0 & 0 & . & . & . & . \\ 1 & 4 & 1 & 0 & 0 & . & . & . \\ 0 & 1 & 4 & 1 & 0 & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & 0 & 0 & 1 & 4 \end{vmatrix} \times \begin{vmatrix} P'(t_2) \\ P'(t_3) \\ P'(t_4) \\ . \\ P'(t_{n-1}) \end{vmatrix} = \begin{vmatrix} 3\left(P(t_3) - P(t_1)\right) - P'(t_1) \\ 3\left(P(t_4) - P(t_2)\right) \\ 3\left(P(t_5) - P(t_3)\right) \\ . \\ 3\left(P(t_n) - P(t_{n-2})\right) - P'(t_1) \end{vmatrix}$$

Підхід базується на рівності на межах інтервалів перших і других похідних. З наведеного матричного виразу знаходять похідні, подані другою матрицею в лівій частині рівняння.

Таким чином, у даному підході як вихідні задаються всі базові точки, через які проходить крива, а також похідні в початковій і кінцевій точках.

Інші методи визначення відсутніх параметрів для побудови сплайнів базуються на накладенні деяких додаткових умов, що призводить до потреби розв’язання системи рівнянь, що виражає ці умови.

При заданні кривої у формі Бєзьє використовуються чотири точки P_1, P_2, P_3, P_4 (рис. 1.17).

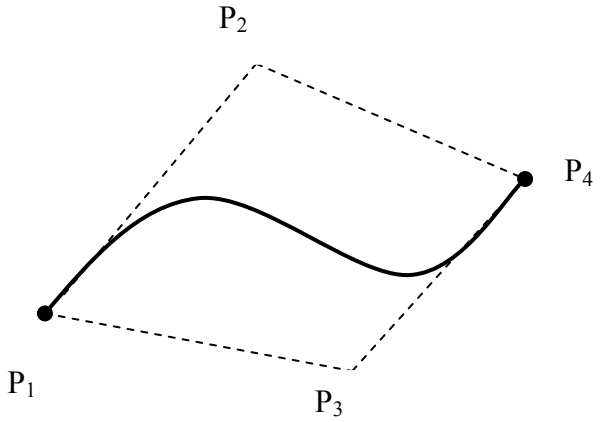


Рисунок 1.17 – Знаходження похідних за методом Безьє

Дотичні вектори у кінцевих точках задаються відрізками P_1P_2, P_3P_4 і розраховуються так:

$$R_1 = 3(P_2 - P_1) = P'(0),$$
$$R_4 = 3(P_4 - P_3) = P'(1).$$

$$G_h = \begin{pmatrix} P_1 \\ P_2 \\ R_1 \\ R_4 \end{pmatrix} = \begin{pmatrix} 1, & 0, & 0, & 0 \\ 0, & 0, & 0, & 1 \\ -3, & 3, & 0, & 0, \\ 0, & 0, & -3, & 3 \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{pmatrix} = M_{hb} G_b$$

Це співвідношення між матрицею Ерміта G_h та геометричною матрицею Бєзьє. Підставляючи отримані значення, знаходимо:\

$$X(t) = TM_hG_h = TM_hM_bG_b.$$

Позначимо добуток M_hM_b як M і отримаємо вираз $X(t) = TMG_b$, котрий зараз має форму Бєзьє.

$$M_b = \begin{pmatrix} -1, & 3, & -3, & 1 \\ 3, & -6, & 3, & 0 \\ -3, & 3, & 0, & 0, \\ 1, & 0, & 0, & 0 \end{pmatrix}$$

Форма Бєзьє використовується у машинній графіці частіше, ніж форма Ерміта Причиною цьому є:

– по-перше, у випадку форми Ерміта дотичні вектори мають задаватися у явному вигляді. У формі Бєзьє дотичні знаходять «локатором»;

– по-друге, у формі Бєзьє чотири точки задають опуклий багатокутник, що можна виконати гумовою ниткою.

Знайдемо $X = TM_bG_b$.

$$X = TM_bG_b = (1-t)^3P_1 + 3t(t-1)^2P_2 + 3t(1-t)P_3 + t^3P_4$$

Можна показати, що сума всіх коефіцієнтів біля t дорівнює 1, що визначає оптимальне середнє значення для чотирьох точок керування.

У загальному випадку, коли в моменти часу $\varepsilon = 0, 1, 2, \ldots$ задано дискретні відліки сигналу $x_0, x_1, x_2, \ldots$ сплайн-функціями, що відновлює дискретизований сигнал n-го степеня та має вигляд

$$S_n(\varepsilon) = \frac{1}{n!}\sum_{i=0}^{\infty} x_i B_n(\varepsilon - i),$$

де $B_n(\varepsilon)$ – деякий сплайн степеня n (B-сплайн Шенберга), який називається ядром сплайна. Функція $B_n(\varepsilon)$ задається так:

$$B_n(\varepsilon) = \sum_{k=0}^{n+1}(-1)^{k_1} C_{n+1}^{k}(\varepsilon - k)_+^n,$$

$$\varepsilon_+^n = \begin{cases} \varepsilon^n, & якщо\ \varepsilon > 0, \\ 0, & якщо\ \varepsilon \le 0, \end{cases}$$

де C_{n+1}^{k} – число сполучень із $(n+1)$ по k.

При переході від інтерполяції багаточленами до інтерполяції сплайнами переслідуються дві мети. Перша – це покращання якості наближення: при однакових обчислювальних витратах абсолютні похибки інтерполяції сплайнами менші, ніж похибки інтерполяції багаточленами, а при однакових похибках зменшується обсяг обчислень. Сплайни дозволяють уникнути осциляцій. Для вирішення задачі збіжності накладаються слабкіші вимоги, ніж у випадку багаточленів. Друга мета – різке зменшення обчислювальних витрат, оскільки як при побудові алгоритмів розв'язання задач, так і при подальшій роботі з апроксимантами використовуються багаточлени невисоких степенів або інші елементарні функції. При роботі зі сплайнами можна використовувати або кусково-багаточленне подання, або подання через базисні функції. У першому випадку досягається найбільша економія арифметичних операцій, але потрібно зберігати великий обсяг інформації про багаточлени. В другому випадку маємо обернене.

Контрольні питання

1. В чому полягають переваги сплайнової інтерполяції відносно інтерполяції за методом Лагранжа?
2. Який мінімальний степінь полінома використовується в сплайновій інтерполяції?
3. В чому відмінність обчислювального процесу при формуванні кривих Єрміта та Безьє?
4. Якими шляхами знаходять похідні на межах інтервалів при сплайновій інтерполяції?
5. Які цілі переслідуються при переході від інтерполяції багаточленами до інтерполяції сплайнами?

1.5 Усунення ефекту аліайзингу векторних меж полігонів

В процесі розгортання в растр графічних образів виникають спотворення в зображенні векторів, ребер багатокутників, які отримали назву ступінчастого ефекту чи ефекту аліайзингу. Основна причина його появи полягає в тому, що кольорові межі об'єктів мають суцільну природу, тоді як растрові пристрої відображення дискретні.

В машинній графіці знайшли використання два основні методи усунення ступінчастого ефекту. Перший із них – метод растеризації – пов'язаний зі збільшенням дискретизації під час формування зображень, що дає можливість враховувати дрібні деталі зображення. В цьому випадку обчислюють растр з вищою роздільною спроможністю, а відображають згідно з роздільною спроможністю пристроїв відображення, використовуючи усереднення.

Недолік цього методу полягає у великій обчислювальній складності, яка залежить від роздільної спроможності координатного простору, оскільки зі збільшенням лінійних розмірів зображення в n разів його площа збільшується у n^2 разів. Практично, для анімації сцени з частотою 10 Гц, що містить близько 200 тис. трикутників, збільшення дискретизації в 16 разів вимагає технічних засобів зі швидкодією 16 GFLOPS. Враховуючи, що пікова продуктивність сучасних процесорів не перевищує 100 MFLOPS, така продуктивність недосяжна навіть для сучасних графічних станцій.

Згідно з другим методом усунення ступінчастості, піксел розглядається не як умовна точка, а як скінченна область. Метод базується на згортанні функції. Для згладжування беруть згортку сигналів для зображен-ня з ядром згортки, а результат використовують для визначення атрибутів піксела.

Як функцію згортання часто використовують прямокутну функцію $h(x)$, $0 < x < 1$. З її допомогою отримують задовільні результати, хоча трикутний та гаусів фільтри дають ще більш якісне згладжування.

Метод згортання, порівняно з методом растеризації, дає краще згладжене зображення, але характеризується значно більшою обчислювальною складністю.

В машинній графіці найбільшого практичного застосування набув окремий випадок методу згортання, який полягає у встановленні інтенсивності кольору піксела пропорційно площі тієї його частини, що відтинається відрізком прямої (рис. 1.18)

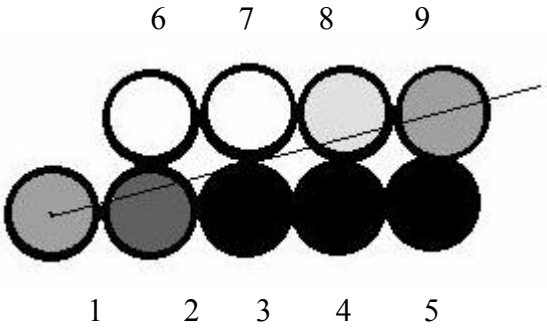


Рисунок 1.18 – Усунення ефекту аліайзингу векторної межі

Алгоритм А. Руа в процесі визначення інтенсивності кольору піксела враховує площі двох суміжних пікселів – основного та допоміжного. При цьому обчислювальний процес визначення координат та інтенсивностей кольору точок розділений. Алгоритм дає придатні результати, але вимагає виконання «довгих» операцій, що суттєво обмежує його використання.

Доцільнішим є метод, що базується на переході від інтерполяції відрізка прямої за значеннями його приростів до інтерполяції за параметрами, що визначаються інтенсивністю кольору кінцевих точок.

Н. Піттуею та Д. Уоткінсону, що запропонували вказаний підхід, не вдалось встановити однозначну відповідність з одним із відомих алгоритмів інтерполяції за методом оцінювальної функції.

Оцінювальна функція визначає похибку між ідеальним відрізком прямої та її кроковою траєкторією. Доведено, що модуль оцінювальної функції в заданій точці дорівнює площі тієї частини дискрети, що відтинається відрізком прямої.

Встановимо взаємозв'язок між значенням оцінювальної функції визначення точок відрізка прямої в дискретному координатному просторі зі значенням їх інтенсивностей кольору.

Нехай відрізок прямої в першому октанті задається своїми більшим M та меншим N приростами на координатній осі. Відомо, що крокові переміщення всіх відрізків прямих з приростами ΔM і ΔN ідентичні і повторюються через M тактів, де n – ціле число.

Якщо I_M – значення інтенсивності кольору, з яким потрібно відтворити відрізок прямої, то для визначення невідомого параметра I_K для інтерполювання відрізка прямої складемо пропорцію

$$N/M=I_K/I_M \ .$$

Звідки

$$I_K = N * I_M/M = I_M*k,$$

де k – кутовий коефіцієнт нахилу прямої.

Таким чином, інтерполювання відрізка прямої з параметрами M та N можна звести до інтерполювання за M тактів відрізка прямої з параметрами I_M та I_K.

Найдоцільніше за алгоритм інтерполювання використати алгоритм оцінювальної функції, розроблений Пєтухом А. М, Обідником Д. Т. Це пояснюється тим, що інтенсивність кольору початкової та кінцевої точок траєкторії має дорівнювати $I_M/2$, оскільки відрізок прямої проходить через центри вказаних точок. Зазначену умову забезпечує вибраний базовий алгоритм.

Контрольні питання

1. Які основні підходи до усунення ефекту аліайзингу Вам відомі?
2. Як здійснюється усунення ефекту аліайзингу з використанням методу оцінювальної функції?

2 ПРОЦЕДУРИ МАШИННОЇ ГРАФІКИ

2.1 Афінні перетворення

Афінні перетворення знаходять широке застосування в задачах машинної графіки. Найбільшого поширення набули і окремі випадки афінних перетворень: зсув, поворот, масштабування.

Нехай у площині задана початкова система координат *0XY* і деяка нова система координат $0_1 X_1 Y_1$. Тоді перетворення, які полягають у тому, щоб у відповідність точці *P* площини *0XY* ставиться точка P_1, яка в новій системі має такі самі координати, що й точка *P* у початковій, називаються афінними.

Основні властивості афінних перетворень

1. Множина точок, яка в початковій системі координат задовольняє деяке рівняння, переходить у множину точок, координати яких у новій системі задовольняють таке саме рівняння. Так, пряма переходить у пряму, площина – у площину.

2. Відношення площ і об’ємів геометричних фігур зберігається.

3. Зберігається просте співвідношення трьох точок.

4. Існує єдине перетворення площини, що переводить трійку точок, які не належать одній прямій, у нову трійку точок, які також не належать прямій.

5. Якщо початкова та нова системи координат є декартовими з однаковими одиничними відрізками по осях, то при перетвореннях зберігаються всі метричні властивості геометричних фігур.

На рис. 2.1 зображені геометричні співвідношення між початковою системою 0XY і системою $0X_1Y_1$, яку одержали при повороті початкової системи на кут α.

За допомогою побудов на рис. 2.1 одержуємо систему рівнянь

$$X_1 = X\cos\alpha + Y\sin\alpha,$$
$$Y_1 = Y\cos\alpha - X\sin\alpha,$$

яку можна зобразити в матричному вигляді

$$[X_1\ Y_1] = [x\ y]\begin{vmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{vmatrix}.$$

При повороті зображення не завжди одержують цілочислові координати, що приводить до необхідності їх округлення.

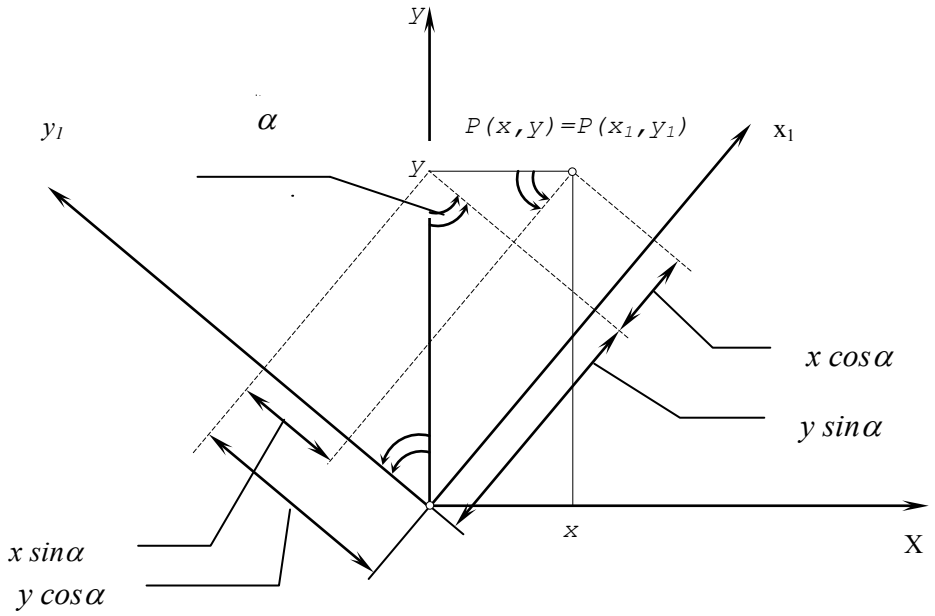


Рисунок 2.1 – Поворот системи координат

Функціональний метод повороту зображення усуває з обчислювального процесу виконання синусно-косинусних перетворень та «довгих» операцій.

Сутність методу полягає в нижчевикладеному.

Декартова площина 1 і відповідна їй система координат ХОУ утворена площиною екрана індикатора зображення, а декартова площина 2 і відповідна їй система координат $X_1OУ_1$ – площиною матриці світлочутливих елементів (рис. 2.2).

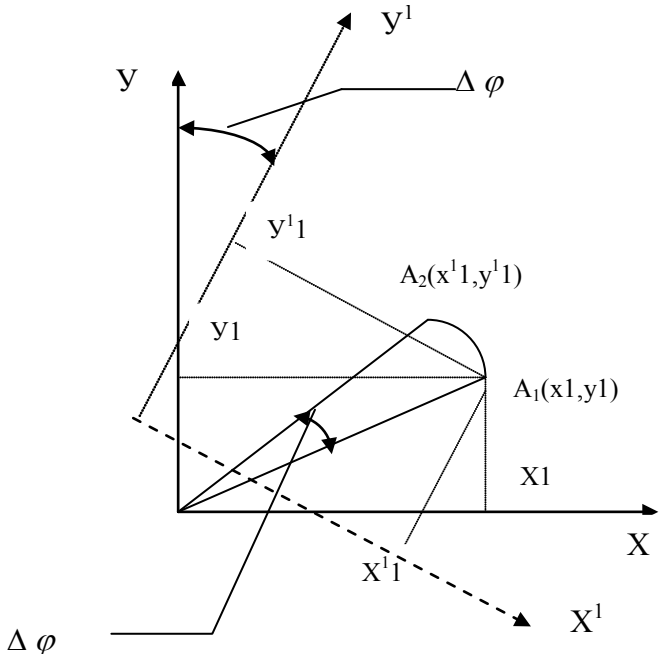


Рис. 2.2. – Функціональний метод повороту

В початковий момент часу точка А1 з координатами Х1, У1 відображається на екрані індикатора. Матриця світлочутливих елементів, яку повернуто відносно екрана на кут $\Delta\varphi$, сприймає точку в системі координат $X^1O^1Y^1$, відносно якої остання має координати $X^1 1, Y^1 1$. Очевидно, що, відображаючи на екрані точку з координатами $X^1 1, Y^1 1$, здійснюється поворот вихідної точки на кут $\Delta\varphi$ в системі координат ХОУ. Виконуючи вказану процедуру для всіх точок зображення, за час одного кадру здійснюється поворот зображення на кут $\Delta\varphi$.

За рахунок повторення процесу n разів, де $n = \varphi/\Delta\varphi$, забезпечується поворот зображення на заданий кут φ.

Оскільки вихідна та нова системи координат є прямокутними декартовими з однаковими одиничними відрізками по осях, то при перетворенні зберігаються всі метричні властивості геометричних фігур.

Недолік функціонального методу повороту полягає у відносно великій похибці перетворення.

При масштабуванні здійснюють збільшення або зменшення розмірів зображення згідно з перетвореннями вигляду

$$[x'\ y'] = [x\ y] \begin{vmatrix} Kx & 0 \\ 0 & Ky \end{vmatrix},$$

де Kx, Ky – масштабні коефіцієнти.

При $Kx = Ky = K$ здійснюється перетворення подібності. Зображення збільшується в K разів при $K > 1$ і зменшується при $K < 1$.

Точка зсувається додаванням до кожної координати точки додатної або від'ємної констант:

$$[x'\ y'\ 1] = [x\ y\ 1] \begin{vmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ m & n & 1 \end{vmatrix},$$

де m, n – значення параметрів зсуву.

Контрольні питання

1. Які перетворення відносять до афінних?
2. Які основні властивості афінних перетворень?
3. Наведіть формули для повороту точки зображення на кут α проти годинникової стрілки.
4. Чи можливе суміщення в одній процедурі різних типів афінних перетворень?
5. Як організовується скролінг вікна зображення?

2.2 Алгоритми відсікання векторів

Область (ділянка) системи координат, в якій формується зображення для виведення на графічні засоби, називається вікном, а область прикладної системи координат, на яку відображається вікно, – областю індикації.

Відсікання – операція, яка відкидає частину зображення, що лежить поза вікном. Його можна виконувати як до перетворення зображення, так і після нього. Відсікання, яке виконують до відображення, забезпечує економію часу за рахунок того, що невидимі лінії не підлягають перетворенням. Якщо сторони вікна похилі відносно координатних осей, то алгоритм відсікання потребує відносно складних обчислень, тому відсікання невидимих частин повернутого зображення виконують після відображення.

Якщо зображення задається як список точок, то кожна точка зображується чи відкидається залежно від результату порівняння її координат з координатами області індикації. Якщо зображення задається як множина відрізків прямих, задача буде складною. Вікно називається регулярним, якщо воно має форму прямокутника зі сторонами, які паралельні осям координат екрана.

Враховуючи, що для формування графічних зображень найчастіше використовуються відрізки прямих, розглянемо відсікання вказаного типу примітивів.

Внаслідок того, що область індикації, як правило, має форму прямокутника, відрізку належить не більше як одна видима частина.

Для реалізації відсікання потрібно знайти координати точки перетину сторін вікна з відрізком і відкинути ту його частину, яка знаходиться за вікном.

В машинній графіці найчастіше використовуються прямокутні вікна, які задаються рівняннями їх сторін (рис. 2.3).

$$X = X_л, \quad X = X_п,$$
$$Y = Y_л, \quad Y = Y_п.$$

Підставляючи в рівняння відрізка прямої $Y = kX + b$ значення $X_л$, $X_п$ знаходять ординати перетину відповідно з лівою та правою межами вікна. Аналогічно знаходять і точки перетину з верхньою та нижньою межами вікна. Для цього в рівняння прямої підставляють відомі значення $У_л$, $У_п$.

Відрізки прямих можуть повністю знаходитись всередині або зовні вікна. Для таких випадків розв’язувати систему рівнянь недоцільно. Взагалі рекомендується обчислювати перетин відрізка прямої з вікном в останню чергу, оскільки для цього потрібний великий обсяг розрахунків.

Провести тестування повної видимості чи невидимості відрізків можна за допомогою алгоритму Д. Коена і А. Сазерленда.

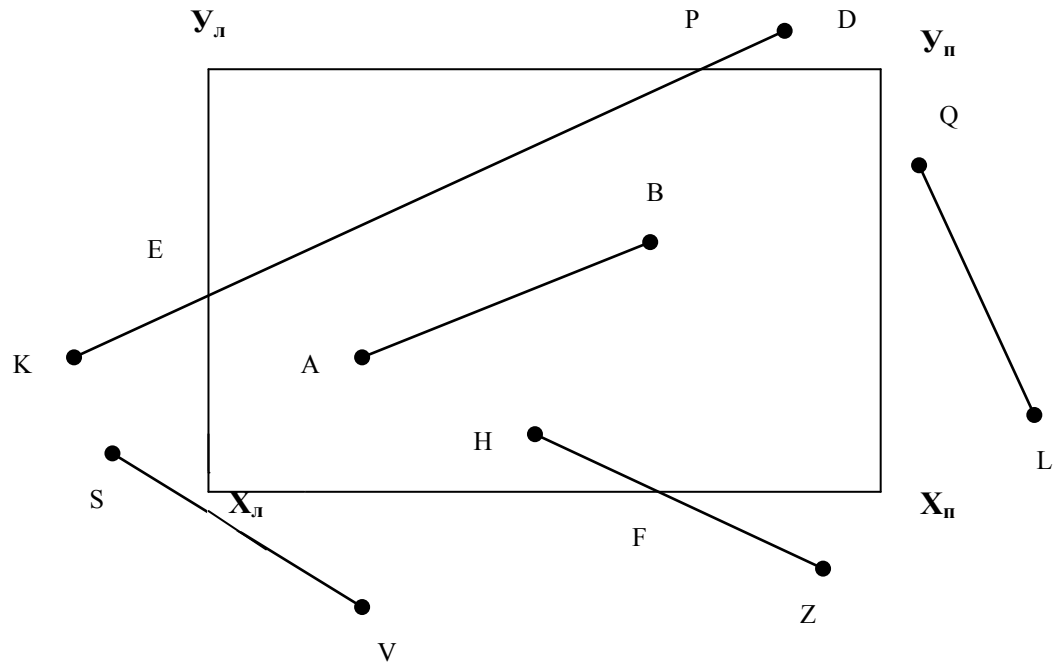


Рисунок 2.3 – Розміщення відрізків прямих відносно вікна

Він ґрунтується на тому, що кожен відрізок або повністю лежить в межах області індикації, або його можна розділити так, щоб одна з його частин була повністю відкинута.

Для перевірки на відсікання межі області індикації проводять так, щоб вони ділили поле, на якому знаходиться зображення, на дев’ять підобластей. Кожній з них присвоюють чотирирозрядний код. Цей код присвоюють кінцевим точкам відрізка, який знаходиться у відповідних підобластях.

1001	1000	1010
0001	0000	0010
0101	0100	0110

Одиниці у відповідних розрядах коду означають: у першому – точка над верхнім краєм області індикації; у другому – точка під нижнім краєм області індикації; у третьому – точка праворуч від області індикації; у четвертому – точка зліва від лівого краю області індикації.

Розглянемо характерні випадки.

1. Чотирирозрядні коди межових точок відрізка прямої AB, який повністю належить вікну (див. рис. 2.3), нульові.

Таким чином, якщо чотирирозрядні коди для обох кінців відрізка дорівнюють нулю, то відрізок повністю лежить в області індикації.

2. Чотирирозрядні коди межових точок відрізка прямої, який розміщений по одну зі сторін вікна (рис. 2.3, відрізок QL), збігаються. Для встановлення вказаного достатньо виконати операцію логічного множення цих двох кодів, яка для вказаного випадку дасть ненульовий результат.

3. Чотирирозрядні коди межових точок відрізка прямої, який частково знаходиться в області індикації (рис. 2.3, відрізки KD, HZ) не збігаються. Тому результат логічного множення цих кодів нульовий.

Для вказаного випадку потрібно виконати відсікання. Координати точки відрізка, яка належить одній з меж вікна, знаходять із рівняння відрізка y = k * x + b підстановкою в нього відповідних координат межових точок вікна і знаходженням невідомої координати.

Для відрізка HZ знаходять точку перетину F, яка ділить відрізок на два зіставних. Надалі для кожного отриманого відрізка виконуємо алгоритм Коена – Сазерленда. В результаті аналізу відрізок FZ відкидається.

Найбільш складним для відсікання є випадок, коли відрізок прямої двічі перетинає межі вікна (рис. 2.3, відрізок KD). В цьому випадку алгоритм Коена – Сазерленда виконується для трьох зіставних відрізків – KE, EP, PD.

4. Якщо відрізок прямої лежить за областю вікна і при цьому його межові точки не належать однаковим підобластям (рис. 2.3, відрізок SV), то результат логічного множення чотирирозрядних кодів дасть нульовий результат. Для визначення повної видимості потрібно перевіряти значення кодів обох кінців окремо.

Один із шляхів визначення точки перетину відрізка прямої з межами вікна зводиться до розв’язання системи рівнянь, яка містить рівняння заданого відрізка прямої та відрізків, які є межовими для вікна. При цьому виконуються операції множення чи ділення, реалізація яких вимагає обчислювальних засобів та великих затрат часу. Вказаного можна уникнути, якщо реалізувати двійковий пошук такого перетину шляхом ділення заданого відрізка навпіл.

Ділення на 2 еквівалентно зсуву операнда на один розряд в сторону розрядів з меншою вагою.

Алгоритм був запропонований Спрулом і Сазерлендом і містить алгоритм Коена – Сазерленда.

Алгоритм використовується тоді, коли алгоритм Коена – Сазерленда не дав позитивних результатів щодо належності відрізка вікну або його розташування за вікном.

В алгоритмі відсікання методом ділення відрізка навпіл використовуються коди кінцевих точок відрізка і перевірки, які виявляють повну видимість відрізків, наприклад відрізок a на рис. 2.4, і тривіальну невидимість відрізків, наприклад відрізок b на рис. 2.4.

Ті відрізки, які за допомогою таких простих перевірок не можна віднести до одної з двох категорій, розбиваються на дві рівні частини. Координати середньої точки розбиття обраховуються за формулами:

$$Xm = (X_2 + X_1) / 2,$$
$$Ym = (Y_2 + Y_1) / 2.$$

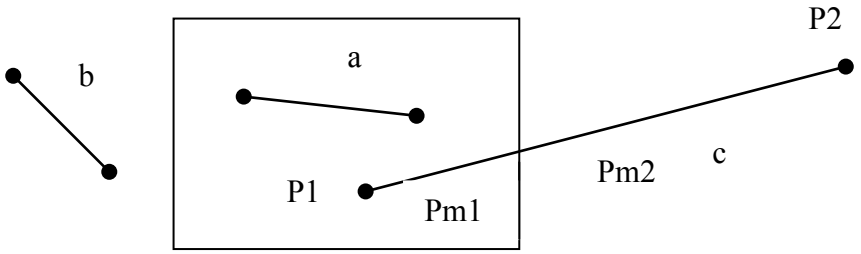


Рисунок 2.4 – Реалізація відсікання згідно алгоритму ділення відрізка навпіл

Аналогічні дії застосовуються до кожних одержаних половин відрізків до тих пір, поки не буде виявлено перетин з однією зі сторін вікна або довжина відрізка, що розглядається, стане занадто малою, тобто, поки вона не перетвориться в точку. Кількість кроків даного алгоритму дорівнює $\log_2 N$, де N – довжина відрізку.

Даний метод проілюстрований на прикладі відрізка с (рис. 2.4). Точка P1 запам'ятовується як поточна видима точка, а відрізок с розбивається навпіл точкою Pm1. Відрізок P2Pm1 відкидається як невидимий, а відрізок P1Pm1 розбивається навпіл точкою Pm2. Відрізок Pm1Pm2 відкидається, а у вікні залишається відрізок P1Pm2 тому, що перевірка по алгоритму Коена-Сазерленда виявляє, що даний відрізок повністю лежить в області індикації.

Контрольні питання

1. В чому полягає процедура відсікання ?
2. Чим зумовлена поява методу ділення відрізка навліл?
3. Які дії виконується згідно з методом Коена – Сазерленда?

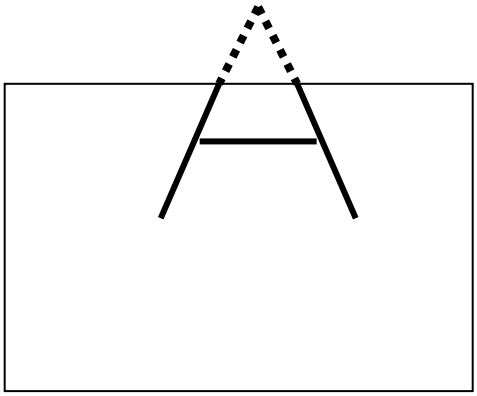
2.3 Алгоритми відсікання тексту

Текст можна відсікати одним з декількох способів.

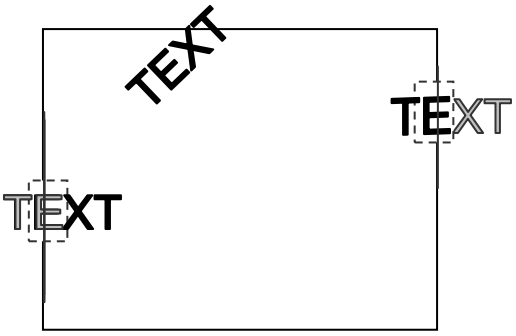
Якщо кожну літеру подавати у вигляді набору коротких відрізків прямих ліній (штрихів), то можна виконати операцію відсікання над кожним відрізком.

Такий підхід забезпечує добрі результати (рис. 2.5, а), але він дуже повільний і несумісний зі звичайними апаратними генераторами літер.

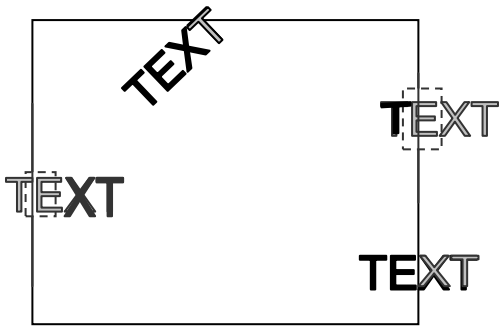
Можна вважати літери об'єктами, які не діляться, та відсікати рядок літер з точністю до літери. Кожна літера уявно поміщається в прямокутник. Деяка точка цієї комірки – центр або один з кутків – узгоджується з вікном: якщо точка всередині, то літера рисується.



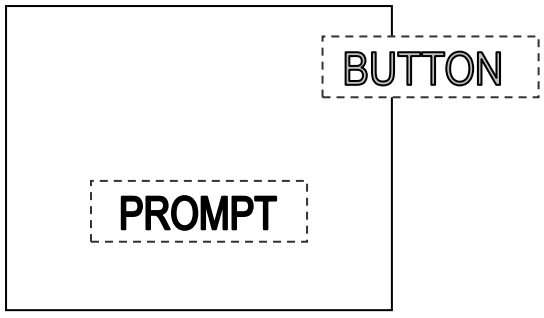
а)



б)



с)



д)

Рисунок 2.5 – Різні підходи до відсікання тексту

Можна узгодити з вікном всю комірку літери або її діагональ. Якщо комірка або її діагональ повністю входять до вікна – літера рисується, навпаки – ні. На рис.2.5, б, с показані результати роботи цих двох підходів: рис. 2.5, б – відсікання літери по нижньому кутку комірки; рис. 2.5, с – відсікання по комірці літери. Відсікання по кутовій точці та комірці/діагоналі дають однаковий результат тільки тоді, коли комірка не перетинається з вікном. Більш того, відсікання по комірці літери та відсікання по її діагоналі еквівалентні тільки тоді, коли сторони комірки паралельні сторонам вікна. При відсіканні потрібно враховувати всю комірку літери (діагоналі недостатньо).

Третій, найбільш простий підхід до відсікання тексту полягає в тому, що весь рядок літер вважається об’єктом, що не ділиться, і весь показується, або не показується (рис. 2.5, д). При цьому підході потрібно узгоджувати з вікном або деяку точку прямокутника, який охоплює рядок, або весь прямокутник.

Контрольні питання

1. Перечисліть основні підходи до відсікання тексту.
2. Коли еквівалентні відсікання по комірці літери та відсікання по її діагоналі?
3. В чому полягають недоліки відсікання тексту при векторному заданні букв?
4. Як виконується відсікання при векторному формуванні букв ?

2.4 Алгоритми заповнення областей, обмежених поліномами

Однією з найбільш поширених задач машинної графіки є задача визначення і заповнення внутрішньої частини контуру.

Зазначена процедура збільшує реалістичність формування графічних зображень, особливо при формуванні векторних сцен.

Процедура заповнення полягає у визначенні адрес всіх внутрішніх точок області, обмеженої контуром, і у встановлені її відповідним пікселам заданого кольору, тобто за визначеними адресами у відеопам’ять записують код кольору.

Заповнення проводять під кутами, кратними 45°, оскільки тільки в цьому випадку будуть відсутні «точки-просікання». Останні мають місце за рахунок зміщення початку вектора, що приводить до пропуску точок області при формуванні діагональних кроків (рис. 2.6).

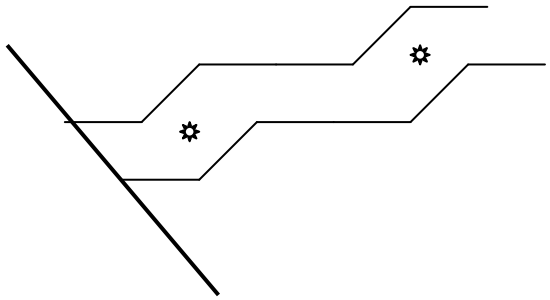


Рисунок 2.6 – Поява точок-просікання

Задачу заповнення розв’язують різними способами, які поділяють на два великих класи Перший з них передбачає наявність точного опису контуру. При цьому визначення частин площини, які лежать всередині області, базується на аналізі рівнянь, які задають відповідні лінії. Методи другого класу передбачають відображення заповнюваного контуру на дискретну площину і визначення внутрішньої частини області на основі значень яскравості пік-селів.

Заповнення області по мірі формування контуру, який її обмежує, полягає в нижчевикладеному (рис. 2.7).

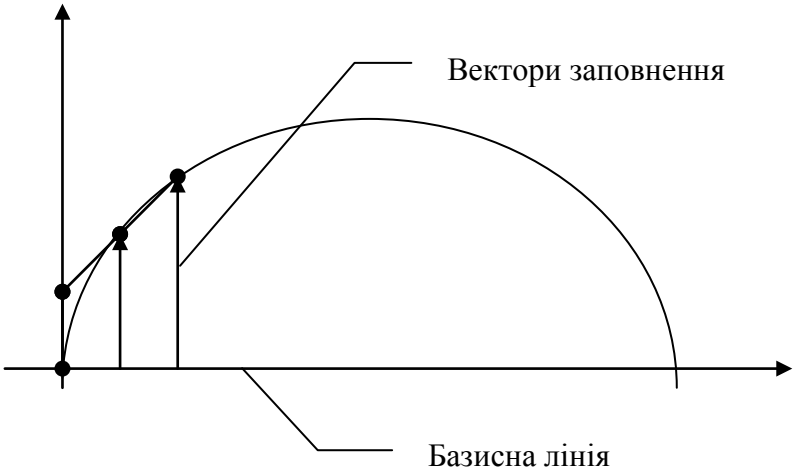


Рисунок 2.7 – Заповнення області, обмеженої контуром,
по мірі його формування

1. Проводять умовну базисну лінію, яка, як правило, поділяє область на 2 частини.
2. Формують точки траєкторії контуру до появи крокового приросту в напрямку базисної лінії.
3. Від базисної лінії проводять вертикальний (горизонтальний) вектор до визначеної точки траєкторії (можливе також заповнення вектором з кутом нахилу 45^0). Для кожної точки вектора встановлюють колір.
4. Дії 2–3 повторюють до заповнення першої частини області.
5. Дії 2–4 виконують для нижньої частини контуру.

Безумовно, базисна лінія може мати будь-яку форму або нахил, але найбільш доцільно використовувати для цього горизонтальний або вертикальний вектор, оскільки в цьому випадку обчислювальні затрати будуть найменші.

Методи заповнення можна поділити залежно від того, який принцип застосовується для встановлення внутрішніх точок багатокутника. У деяких алгоритмах застосовується перевірка на парність, в інших – критерій зв'язності.

Алгоритми заповнення області, в яких застосовують перевірку на парність, базуються на тому, що довільна пряма перетинає будь-яку замкнену криву парну кількість разів. Якщо відомо, що перша точка відповідної лінії лежить за контуром, то, виконавши обхід цієї лінії, шляхом відрахувань кількості перетинів можна встановити, які саме її відрізки розміщені в області. Якщо число перетинів непарне, то відповідний відрізок розміщений у внутрішній області (відрізки AB і CD на рис.2.8), в іншому випадку – поза областю (відрізок BC).

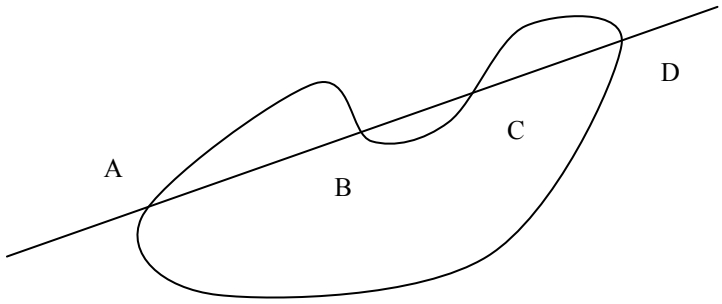


Рисунок 2.8 – Визначення межі контуру за критерієм парності

Як правило, вводять допоміжну змінну з нульовим початковим станом. При кожному перетині контуру сканувальним відрізком прямої стан змінної змінюють на протилежний (рис. 2.9).

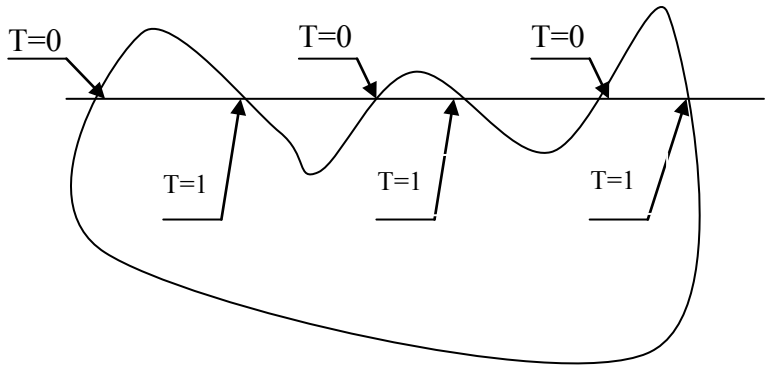


Рисунок 2.9 – Формування станів змінної Т при скануванні області

У разі, коли сканувальний відрізок прямої є дотичним до контуру, алгоритм не дає правильний результат, оскільки дотик може здійснюватись кількома точками. Останнє вказує на потребу виявлення точок-дотику.

У цьому випадку можна виконувати аналіз за двома допоміжними прямими ДП1 і ДП2 (рис. 2.10), розміщеними над і під сканувальним відрізком ОП, який використовують для перевірки на парність. Якщо одна з цих прямих не перетинає контур біля точки перетину, це означає, що відрізок ОП є дотичним.

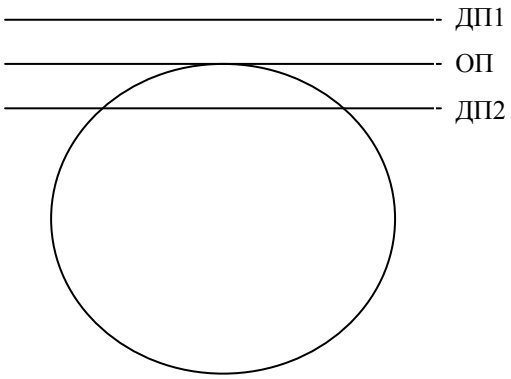


Рисунок 2.10 – Визначення наявності дотичних до контуру

Визначити критичні ситуації можна також за кількістю перетинів сканувального відрізка прямої з контуром (рис. 2.11). Якщо кількість перетинів λ для сусідніх рядків не збігаються, то потрібно виконати додатковий аналіз на предмет обробки нештатної ситуації.

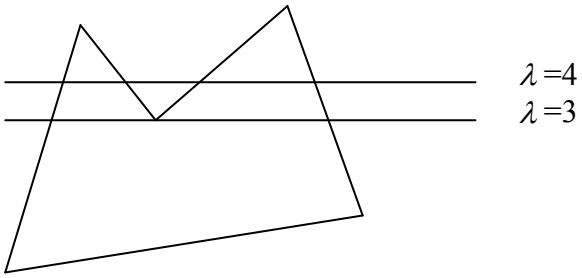


Рисунок 2.11 – Визначення критичних ситуацій за кількістю перетинів контуру сканувальним відрізком прямої

Розглянемо метод і відповідний йому алгоритм заповнення області, обмеженої сторонами багатокутника, що використовує принцип парності. Будемо вважати, що сторони багатокутника задані координатами вершин багатокутника x_i, y_i, яким відповідає максимальне значення y або мінімальне значення x, якщо сторона горизонтальна, а також прирости Δx, Δy для визначення координат іншої вершини.

Першим кроком алгоритму є сортування сторін відповідно до максимальних значень y_i, а при їхній рівності – за мінімумом x_i. У випадку рівності і x_i використовуються значення Δx, а потім Δy. При цьому вибирається сторона з найменшим алгебраїчним значенням. У результаті одержуємо впорядкований список сторін контуру – список черговості.

Другий крок алгоритму виконується для всіх суміжних сторін списку з однаковими значеннями y, що відповідає локальному максимуму. Через вершину багатокутника, що відповідає цим сторонам, проводиться пряма, паралельна осі X, і визначаються точки її перетину з іншими сторонами, що розташовані в списку вище проаналізованих. Дані прямі поділяють контур на прошарки, що містять парне число меж. Кожному прошарку відповідає спеціальний список, названий поточним, що містить межі. Для формування поточного списку список черговості містить мітки трьох типів.

Мітка *M1* визначає число меж, переданих із списку черговості в поточний список: дві – якщо точка відповідає максимуму; одна – якщо вона не є максимумом; більше двох, якщо одному значенню y відповідає більше одного максимуму.

Мітка *M2* служить для введення в поточний список наступної пари меж при переході в наступний прошарок. Її значення відповідає значенню y, при котрому цей перехід має відбутися. Зазначеними мітками мітять перші сторони зі списку черговості, з якими перетинаються прямі, проведені через вершини багатокутника, що є локальними максимумами.

Мітка *M3* визначає число нових меж, до яких провадиться звертання при завершенні роботи з поточною межею. Табл. 2.1 ілюструє формування

списку черговості і міток для одержання поточного списку на прикладі фрагмента області, наведеної на рис. 2.12.

Таблиця 2.1 – Опис меж

Сторона:	X	Y	X	Y	Мітка 1	Мітка 2	Мітка 3
AB	X_A	Y_A	X_B-X_A	Y_B-Y_A	2	Y_D	1
AC	X_A	Y_A	X_C-X_A	Y_C-Y_A	1	—	0
DC	X_D	Y_D	X_C-X_D	Y_C-Y_D	2	—	0
DE	X_D	Y_D	X_E-X_D	Y_E-Y_D	1	—	1
EF	X_E	Y_E	X_F-X_E	Y_F-Y_E	1	—	1

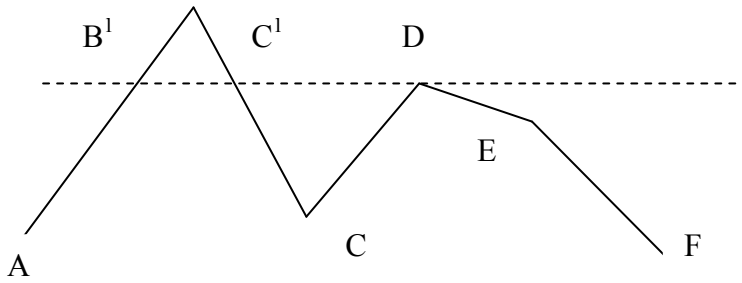


Рисунок 2.12 – Формування списку черговості

Третім кроком алгоритму є безпосереднє заповнення області послідовно для кожного прошарку. Цей процес починається з першої пари меж списку черговості шляхом формування та обновлення поточного списку, використовуючи наявні мітки, що дозволяють видаляти і вносити нові межі. Організація поточного списку дає можливість реалізувати для кожного прошарку принцип парності у зв’язку з тим, що всі локальні максимуми визначені. Потрібно відзначити, що наведений алгоритм не враховує наявність горизонтальних сторін багатокутника. Проте модифікація алгоритму для врахування цих ситуацій не має істотних трудностей.

Заповнення області за критерієм зв’язності вимагає наявності піксела-затравки, який розміщений всередині контуру.

Для його задання існує декілька засобів. У окремих режимах він може бути заданий оператором. У інших випадках за піксел-затравку можна взяти точку, безпосередньо суміжну з пікселем контуру. Що стосується задання контуру, то він, як правило, визначається сукупністю точок поелементного еквівалента зображення. У випадку задання контуру багатокутником попередньо здійснюється процедура одержання його поелементного

еквівалента за допомогою генератора векторів. Будемо вважати, що контур та затравний піксел згенеровані.

Найпростіший рекурсивний алгоритм полягає в розгляді чотирьох сусідніх до затравного піксела елементів на предмет їхньої належності контуру (рис. 2.13). Пікселі, що не належать контуру, у свою чергу можуть використовуватися як затравні.

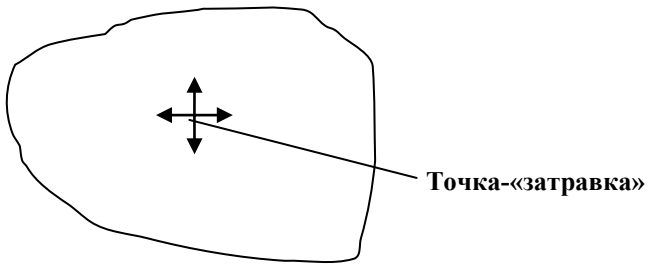


Рисунок 2.13 - Заповнення області рекурсивним алгоритмом

Розглянутий алгоритм при своїй зовнішній простоті потребує значного обсягу обчислень. Це пов’язано в першу чергу з дублюванням розгляду сусідніх елементів для сусідніх затравних пікселей.

Альтернативний нерекурсивний алгоритм полягає в нижчевикладеному. Починаючи з затравного піксела (рис. 2.14.) проводиться аналіз на належність контуру пікселей, що знаходяться зліва, а потім – справа від затравного з одночасним зафарбуванням. Далі здійснюється перехід на нижню сторону з повторенням процедури.

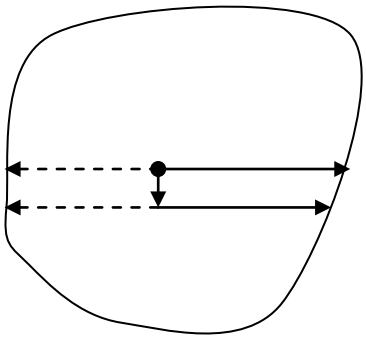


Рисунок 2.14 – Заповнення області, обмеженої контуром за критерієм зв’язності

Цей процес продовжується до повного заповнення нижньої від затравного піксела частини області. Потім проводиться заповнення верхньої частини області.

У випадку заповнення невипуклих областей необхідно реалізувати стек, у якому будуть зберігатися пікселі, розташовані поблизу максимумів і мінімумів контуру. Зазначені пікселі використовуються як затравні, послідовно зчитуються зі стека. Очевидно, що задача обчислення цих пікселів не є особливо складною.

На рис. 2.15 наведений приклад задання кількох затравних пікселів для невипуклих контурів.

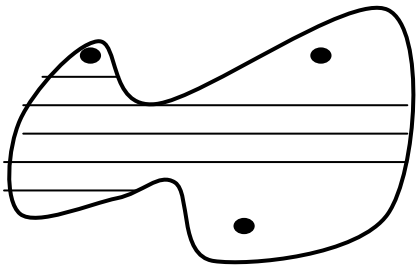


Рисунок 2.15 – Задання точок-«затравок» для невипуклого контуру

Контрольні питання

1. В чому причина появи точок просікання?
2. Як знаходять дотичні до контуру?
3. Порівняйте швидкодію алгоритмів за критерієм зв’язності та парності.
4. В яких випадках потрібно декілька точок затравлення?

3 МЕТОДИ ПОБУДОВИ ПОВЕРХОНЬ

У реальних системах моделювання об’єктів і сцен носить ієрархічний характер. Верхні рівні мають більш концептуальний характер, а нижні пов’язані з безпосереднім математичним поданням примітивів, що складають об’єкти і сцени. Можна умовно виділити зовнішнє і внутрішнє подання об’єктів у системі генерації реалістичних зображень. Зовнішнє подання може бути досить високорівневим і визначається, як правило, прикладною спрямованістю системи. Воно використовується прикладними програмістами для розробки сценаріїв і поповнення баз даних.

Внутрішнє подання об’єктів реалізується, як правило, через низькорівневі примітиви, використовувані на всіх етапах конвеєра генерації реалістичних зображень. До таких примітивів відносять плоский полігон (зазвичай, чотиристоронній): трикутник, символ, вектор.

У більшості систем використовується подання об’єктів на рівні полігонів, що відповідним чином визначає підходи до технології візуалізації. На жаль, визначення складних об’єктів, особливо природного походження, через низькорівневі примітиви досить громіздко і не завжди дає високий ступінь реалізму.

Як примітиви більш високого рівня можуть бути використані параметричні сплайни, що дозволяють подавати об’єкти у вигляді сукупності фрагментів криволінійних поверхонь. Однак безпосередня візуалізація сплайнів зіштовхується з проблемою розробки спеціальних алгоритмів і апаратних засобів. Тому звичайно використовується декомпозиція сплайна на полігони (трикутники) до або в процесі візуалізації. У САПР машинобудування може бути використаний інший підхід, де складні об’єкти формуються об’єднанням, перетинанням, накладенням таких простих об’ємних примітивів, як сфери, куби, циліндри й інші (конструктивна геометрія суцільних тіл). Візуалізація таких об’єктів за допомогою апаратури може робитися безпосередньо, без декомпозиції на полігони . При цьому досягається висока динамічність зображень, потрібна для підтримки інтерактивного процесу проєктування.

У синтезі реалістичних зображень складною задачею є моделювання таких природних об’єктів, як хмари, місцевість, флора, вогонь і результати біологічних процесів. Високий ступінь складності об’єктів, наявність дрібних деталей породжують проблеми їхнього описи і збереження в базі даних. Вихід з цієї ситуації визначений використанням процедурних стохастичних моделей. У базі даних визначається загальна форма за допомогою точного визначення декількох ключових параметрів, далі, для відтворення необхідних

деталей, використовується стохастичний процес. Точність відтворення не є дуже важливим, головне – одержання візуально прийнятних результатів. Для кожного класу об’єктів існують свої підходи до створення процедурних стохастичних моделей, що інтенсивно розвиваються.

3.1 Подання поверхні полігональною сіткою

Полігональна сітка являє собою сукупність ребер, вершин і багатокутників. Вершини з’єднуються ребрами, а багатокутники розглядаються як послідовності ребер або вершин. Сітку можна задавати декількома різними способами. Прикладному програмісту варто обрати спосіб, який найбільш підходить для його задачі. Зрозуміло, в одній задачі може з однаковим успіхом використовуватись одразу декілька подань: для зовнішньої пам’яті, внутрішнього використання і користувача. Для оцінювання цих подань використовуються такі критерії:

- Обсяг потрібної пам’яті.
- Легкість ідентифікації ребер.
- Легкість ідентифікації багатокутників, яким належить дане ребро.
- Легкість процедури пошуку вершин, що утворюють ребро.
- Легкість визначення всіх ребер, що утворюють багатокутник.
- Легкість отримання зображення полігональної сітки.
- Легкість знаходження помилок в поданні (наприклад, відсутність ребра, вершини чи багатокутника).

В загальному випадку, чим більш явно виражені залежності між багатокутниками, вершинами і ребрами, тим швидше виконують операції над ними і тим більше пам’яті потребує відповідне подання. В деяких випадках ребра полігональних сіток є спільними для більш ніж двох багатокутників (рис. 3.1.).

Розглянемо три найбільш поширені способи опису полігональних сіток.

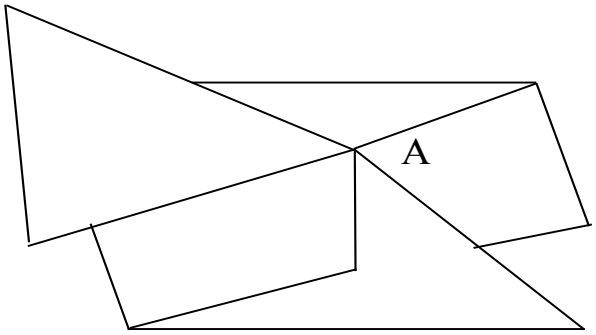


Рисунок 3.1 – Множина багатокутників зі спільним ребром

1. Явне задання багатокутників

Кожен багатокутник можна подати у вигляді списку координат його вершин:

$$P = ((X1,Y1,Z1),(X2,Y2,Z2),...,(XN,YN,ZN)).$$

Вершини запам’ятовуються в тому порядку, в якому вони зустрічаються при обході навколо багатокутника. При цьому всі послідовні вершини багатокутника, а також перша і остання, з’єднуються ребрами. Для кожного окремого багатокутника даний спосіб запису є ефективним, але для полігональної сітки дає великі втрати пам’яті внаслідок дублювання інформації про координати спільних вершин. Більш того, явного опису спільних ребер і вершин просто не існує. Наприклад, пошук всіх багатокутників, які мають спільну вершину, потребує порівняння трійок координат одного багатокутника з трійками координат решти багатокутників. Найбільш ефективний спосіб виконати таке порівняння полягає у сортуванні всіх N координатних трійок: для цього потрібно в кращому випадку N•Log2•N порівнянь. Але й тоді існує небезпека того, що одна і та ж вершина, внаслідок помилок округлення, може в різних багатокутниках мати різні значення координат, тому правильна відповідність може бути не знайдена. Полігональна сітка зображується шляхом креслення ребер кожного багатокутника, але це призводить до того, що спільні ребра рисуються двічі – по одному разу для кожного з багатокутників. Окремий багатокутник зображується тривіально.

2. Задання багатокутників за допомогою покажчиків

При використанні цього подання кожен вузол полігональної сітки запам’ятовується лише один раз в списку вершин V = ((X1, Y1, Z1), ... ,(XN, YN, ZN)). Багатокутник визначається списком покажчиків (або індексів) в списку вершин. Багатокутник, складений з вершин 3, 5, 7 і 10 цього списку, поданий як P = (3, 5, 7, 10). На рис. 3.2 наведений приклад такого подання. Воно має низку переваг порівняно з явним завданням багатокутників. Оскільки кожна вершина багатокутника запам’ятовується тільки один раз, вдається зберегти значний обсяг пам’яті. Крім того, координати вершини можна легко змінювати. Але все ще не просто знаходити багатокутники зі спільними ребрами; останні при зображенні всієї полігональної фігури рисуються двічі. Ці проблеми можна вирішити, якщо робити опис ребер в явному вигляді.

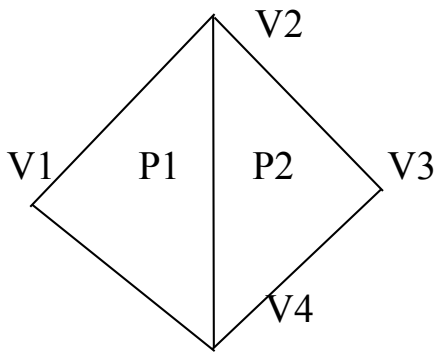
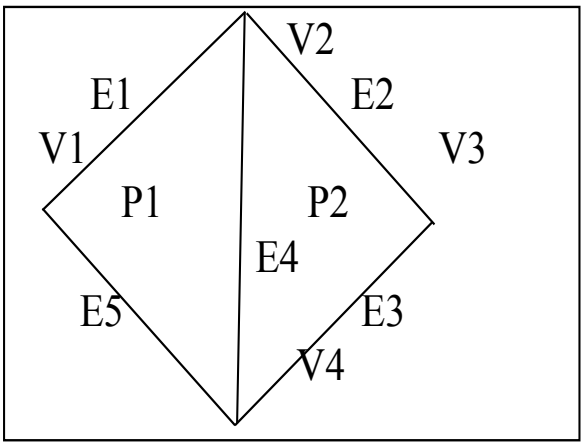


Рисунок 3.2 – Задання багатокутників за допомогою покажчиків

V = (V1, V2, V3, V4) = ((X1, Y1, Z1), ..., (X4, Y4, Z4)),
P1 = (1, 2 ,4), P2 = (4, 2, 3).

3. Явне завдання ребер

В цьому поданні є список вершин V, однак будемо розглядати тепер багатокутник не як список покажчиків на список вершин, а як сукупність покажчиків на елементи списку ребер, в якому ребра зустрічаються лише один раз. Кожне ребро в списку ребер вказує на дві вершини в списку вершин, що визначають це ребро, а також на один чи два багатокутники, яким це ребро належить. Таким чином ми описуємо багатокутник як P = (E1, .., EN), а ребро як E = (V1, V2, P1, P2). Якщо ребро належить лише одному багатокутнику, то або P1 або P2 – пусте. На рис. 3.3 наведений приклад такого подання.



V = (V1, V2, V3, V4) =
= ((X1, Y1, Z1), ..., (X4, Y4, Z4)),
E1 = (V1, V2, P1, L),
E2 = (V2, V3, P2, L),
E3 = (V3, V4, P2, L),
E4 = (V4, V2, P1, P2),
E5 = (V4, V1, P1, L),
P1 = (E1, E4, E5),
P2 = (E2, E3, E4).

Рисунок 3.3 – Явне завдання ребер полігональної сітки

При явному завданні ребер полігональна сітка зображується шляхом креслення не всіх багатокутників, а всіх ребер. В результаті вдається

запобігти багатократному кресленню спільних ребер. Окремі багатокутники при цьому також зображуються досить просто.

В деяких випадках ребра полігональних сіток є спільними для більш ніж двох багатокутників (див. рис. 3.1). Розглянемо, наприклад, випадок в картографії, коли такі підрозділи, як області, райони і т. д. можуть належати одночасно кільком багатокутникам. Якщо врахувати поділ міст на райони, виборчі дільниці, то це число істотно зросте. Аналогічно в деяких тривимірних пристосуваннях, таких як опис структури металевої кулі, ребра іноді належать трьом багатокутникам. Для таких випадків описи ребер можуть бути розширені, щоб вмістити довільне число багатокутників E = (V1, V2, P1, P2, ..., PN).

В жодному з цих подань задача визначення ребер, інцидентних вершині, не є простою – для її розв'язання потрібно перебрати все ребра. Звичайно, для визначення таких відношень можна безпосередньо використовувати додаткову інформацію.

Наприклад, в поданні, запропонованому Бомгартом, використовується розширений опис ребер, що охоплює покажчики на два сусідніх ребра кожного багатокутника, а також опис вершин, що містить покажчик на (довільне) ребро, інцидентне вершині.

При роботі з багатокутниками використовують рівняння площини, де лежить багатокутник.

Рівняння площини в просторі має вигляд $Ax + By + Cz + D = 0$. Для задання площини достатньо трьох точок.

Якщо задано 3 точки, то маємо систему рівнянь

$$Ax_1 + By_1 + Cz_1 + D = 0,$$
$$Ax_2 + By_2 + Cz_2 + D = 0,$$
$$Ax_3 + By_3 + Cz_3 + D = 0.$$

Якщо три точки не колінеарні, то рівняння має розв'язок відносно A, B, C та D, якщо присвоїти яке-небудь значення одному з коефіцієнтів. Наприклад, припустимо, що D = 1 і розв'яжемо систему рівнянь.

Більш раціональне рішення ґрунтується на тому, що, якщо точки P1, P2, P3 і (x, y, z) знаходяться в одній площині, то

$$Ax + By + Cz + D = 0,$$
$$Ax_1 + By_1 + Cz_1 + D = 0,$$
$$Ax_2 + By_2 + Cz_2 + D = 0,$$
$$Ax_3 + By_3 + Cz_3 + D = 0.$$

Для вказаної системи рівнянь запишемо:

$$\begin{vmatrix} X & Y & Z & 1 \\ X_1 & Y_1 & Z_1 & 1 \\ X_2 & Y_2 & Z_2 & 1 \\ X_3 & Y_3 & Z_3 & 1 \end{vmatrix} = 0$$

Детермінант можна подати у вигляді

$$X\begin{vmatrix} Y_1 & Z_1 & 1 \\ Y_2 & Z_2 & 1 \\ Y_3 & Z_3 & 1 \end{vmatrix} - Y\begin{vmatrix} X_1 & Z_1 & 1 \\ X_2 & Z_2 & 1 \\ X_3 & Z_3 & 1 \end{vmatrix} + Z\begin{vmatrix} X_1 & Y_1 & 1 \\ X_2 & Y_2 & 1 \\ X_3 & Y_3 & 1 \end{vmatrix} = \begin{vmatrix} X_1 & Y_1 & Z_1 \\ X_2 & Y_2 & Z_2 \\ X_3 & Y_3 & Z_3 \end{vmatrix} = 0$$

Звідки випливає, що невідомі коефіцієнти A, B, C і D дорівнюють відповідним детермінантам. Обчислення останніх дозволяє достатньо легко задати площину в координатному просторі.

Контрольні питання

1. Дайте порівняльну характеристику різних форм задання полігональної сітки.
2. Який основний недолік подання поверхні полігональною сіткою?
3. Чим визначається точність подання поверхні полігональною сіткою?
4. Яким чином задається площина?

3.2 Фактурні побудови

В машинній графіці фактурою називають деталізацію побудови поверхні. Найбільшого розповсюдження отримали два способи деталізації.

Перший полягає в тому, що на рівну поверхню наносять раніше заданий візерунок. Після цього поверхня все одно залишається рівною. Накладання візерунка на рівну поверхню виконується за допомогою функції відображення.

Другий спосіб деталізації полягає у створенні нерівностей на поверхні. Такі шорсткі поверхні реалізуються шляхом внесення зміни у параметри, котрі задають поверхню.

Вперше метод для нанесення візерунка на поверхню запропонував Кетмул.

Головним при нанесенні на поверхню є відображення, тому в даному випадку задача зводиться до перетворення системи координат.

Якщо рисунок заданий в прямокутній системі координат (u, w), а поверхня в іншій прямокутній системі координат (Q, φ), то для нанесення

рисунка на поверхню треба знайти чи задати функцію відображення одного простору на інший.

$$Q = f(u,w),\ \varphi = g(u,w), \quad \text{чи} \quad u = r(Q,\varphi),\ W = S(Q,\varphi).$$

Звичайно, хоча й необов'язково, передбачається, що функція відображення лінійна: $Q = Au + B$, $\varphi = Cw + D$, де коефіцієнти A, B, C, D знаходять зі співвідношення між двома відомими точками в системах координат.

Розглянемо конкретний приклад.

Візерунок являє собою просту сітку з прямих, які перетинаються (рис. 3.4). Параметричне подання октанти сфери

$$X = \sin Q \sin\varphi,$$
$$Y = \cos\varphi,$$
$$Z = \cos Q \sin\varphi.$$

Нехай функція відображення лінійна, тобто

$$Q = Au + B, \quad \varphi = Cw + D$$

і кути візерунка переходять в кути октанта

$$u = 0,\ w = 0 \ \text{ при } Q = 0,\ \varphi = \pi/2,$$
$$u = 1,\ w = 0 \ \text{ при } Q = \pi/2,\ \varphi = \pi/2,$$
$$u = 0,\ w = 1 \ \text{ при } Q = 0,\ \varphi = \pi/4,$$
$$u = 1,\ w = 1 \ \text{ при } Q = \pi/2,\ \varphi = \pi/4.$$

Звідки $A = \pi/2$, $B = 0$, $C = -\pi/4$, $D = \pi/2$, тобто, $Q = u\pi/2$, $\varphi = \pi/2 - \pi w/4$.

Обернене перетворення має вигляд

$$u = Q/(\pi/2),\ w = (\pi/2-\varphi)/\pi 4.$$

У цьому методі рисунок наноситься на рівну поверхню, і вона після цього залишається рівною. Для того, щоб поверхня здавалася нерівною, можна відцифрувати фотографію нерегулярною фактурою і відобразити її на поверхню.

Блінк будує нову поверхню, котра має вигляд нерівної, записуючи у напрямок нормалі функцію збурення P(u,w).

Як P можна використовувати майже кожну функцію, в якій присутні частинні похідні.

Одні з останніх методів будування нерегулярностей базуються на фрактальних поверхнях. Фрактальна поверхня будується з випадково заданих полігональних чи біполімінальних поверхонь. За допомогою фрактальних поверхонь рисуються природні об'єкти – каміння, дерева, хмари.

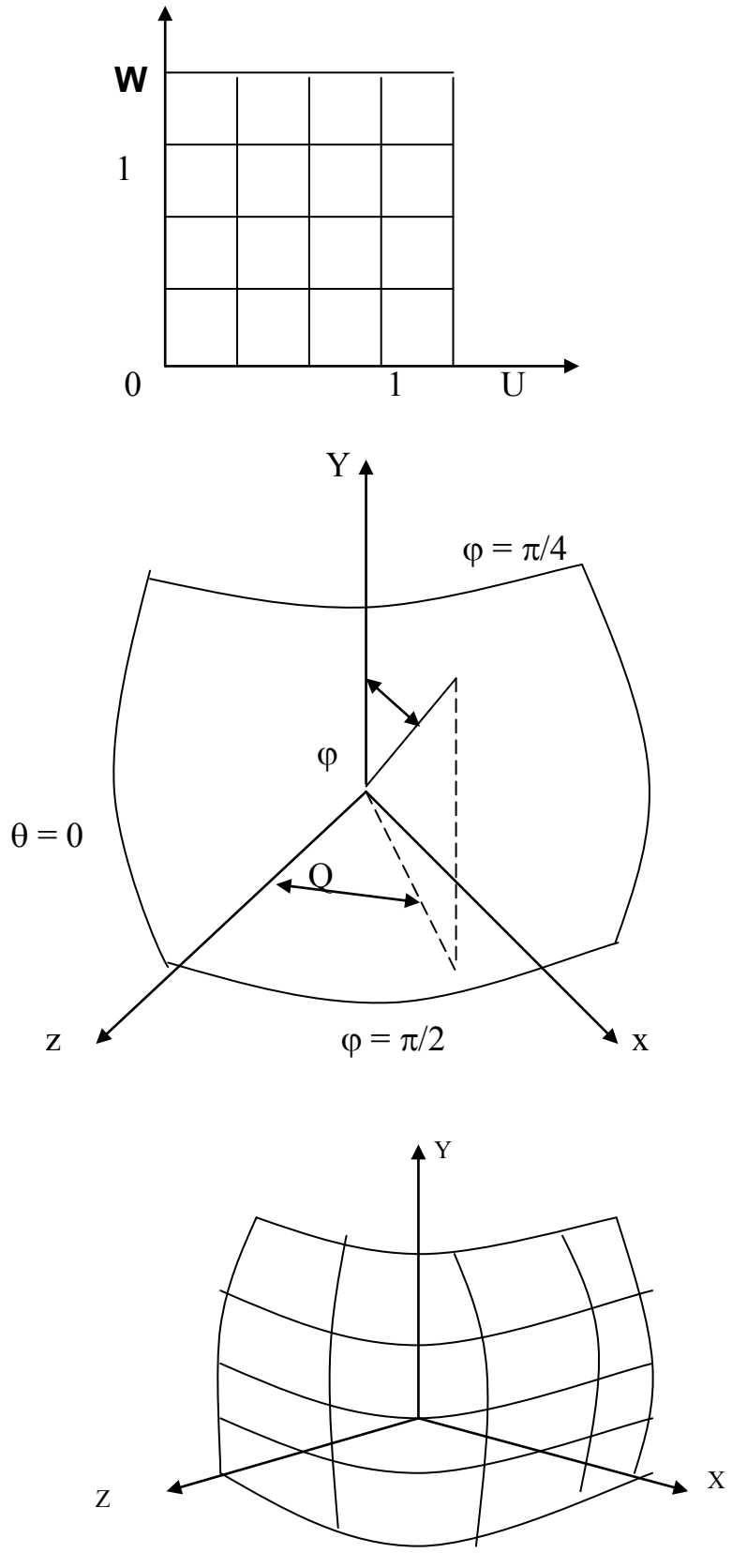


Рисунок 3.4 – Формування поверхні

Контрольні питання

1. Яку функцію найбільш доцільно вибрати для функції відображення?
2. Як задається фактурна сітка?
3. Які методи застосовують для побудови шорстких поверхонь?

3.3 Формування параметричних бікубічних поверхонь

Бікубічні поверхні задаються кубічними рівняннями від двох змінних ***s*** і ***t.*** Змінюючи обидва параметри від 0 до 1, визначають всі точки на частині поверхні.

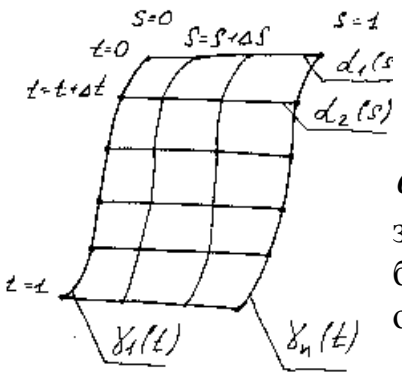
Будемо використовувати рівняння для ***x***

$$x(s,t) = a_{11}s^3t^3 + a_{12}s^3t^2 + a_{13}s^3t + a_{14}s^3 + a_{21}s^2t^3 + a_{22}s^2t^2 + a_{23}s^2t + a_{24}s^2 + \\ + a_{31}st^3 + a_{32}st^2 + a_{33}st + a_{34}s + a_{41}t^3 + a_{42}t + a_{43}t + a_{44}.$$

Запишемо його в більш зручній формі

$$x(s, t) = S\ Cx\ T^T,$$

де $S = [s^3\ s^2\ s\ 1]$, $T = [t^3\ t^2\ t\ 1]$, а T^T позначує транспоновану матрицю T. Наведений запис називають алгебраїчною формою подання, оскільки Cx задає коефіцієнти бікубічного багаточлена. Існують також Cy і Cz, які визначають коеффіцієнти $y(s, t)$ та $z(s, t)$.



Для $\alpha_i(s)$ $s = 0..1$

Для $\gamma_j(t)$ $t = 0..1$

$\gamma_1(t)$ і $\gamma_n(t)$ є граничними точками для кривих $\alpha_i(s)$. Δt и Δs вибираються довільно. Чим менші значення зазначених параметрів, тим більш точно буде визначена форма поверхні і тим більший обсяг обчислень буде потрібний.

Контрольні питання

1. Наведіть рівняння бікубічної поверхні.
2. Чим визначається точність задання бікубічної поверхні?
3. Як формується бікубічна ділянка поверхні?

4 МЕТОДИ ПОБУДОВИ РЕАЛІСТИЧНИХ ТРИВИМІРНИХ ЗОБРАЖЕНЬ

4.1 Основні типи перспективних зображень

Одне з обмежень, яке властиве моделюванню тривимірних об’єктів, полягає в тому, що показувати їх доводиться на плоскому, двовимірному екрані. Об’ємні дисплеї існують, але поки зустріти їх можна тільки в дослідницьких лабораторіях.

Для відображення тривимірного об’єкта на двовимірний екран або інший зовнішній пристрій використовується математичне перетворення, яке називається проекціюванням. Точки, що визначають відрізки прямих, криві й інші елементи відображаються на двовимірну площину.

Це досягається так. Уявну проекційну площину, яку називають картинною площиною, розміщують між об’єктом і спостерігачем, перпендикулярно до напрямку погляду. Проводяться проекційні лінії від точок об’єкта до спостерігача. Точки, де ці лінії перетинають картинну площину, є відповідними точками проекції. Для комп’ютера відносно легко обчислити ці перетинання. Отримавши точки на картинній площині, легко перенести підсумкове зображення на екран. Більш того, змінюючи місце розташування спостерігача і знову виконавши проекціювання, можна одержати вигляди об’єкта з різних сторін.

Проекції розділяють на два різних класи: паралельні і перспективні. При паралельному проекціюванні проекційні лінії йдуть паралельно погляду спостерігача. При перспективному проекціюванні ці лінії умовно перетинаються в оці спостерігача. Останній підхід створює ефект скорочення, коли розмір проекції зменшується зі збільшенням відстані вихідного об'єкта від проекційної площини. Перспективні проекції виглядають більш реалістично, ніж паралельні.

Машинні перспективні зображення поділяються на види за ознаками, які характеризують ступінь наближення їхнього сприймання до сприймання реальних об’єктів, наприклад таких, як наявність невидимих ліній, зображення півтонів, тіней. В сукупності з використаними технічними засобами вони визначають якість відтворюваних зображень.

Основні види перспективних зображень. До них відносять каркасні, (дротові), контурні та півтіньові зображення. На різних стадіях процесу проекціювання потрібні різні види перспективи. Наприклад, для оцінювання остаточного проекційного рішення використовують найбільш якісне перспективне зображення, а для корегування конструкції в процесі її

формування доцільно бачити всі лінії її каркасу, тобто використовувати дротове перспективне зображення. Сучасні растрові кольорові дисплеї в комплексі з відповідними обчислювальними засобами і програмним забезпеченням дозволяють відтворювати півтіньові кольорові зображення з власними та палаючими тінями.

Каркасні машинні перспективні зображення складаються з сукупності ліній графічної моделі відтворюваного об'єкта, включаючи також з невидимими з даної точки зору. Об'ємність зображеного простору заснована на перспективному ефекті, що створює ілюзію глибини зображення. Недоліком дротових зображень є прозорість об'єків, оскільки так як відтворюються всі лінії всіх зображених об'єктів.

Реалізація цього виду перспективних зображень вимагає найменших витрат машинних ресурсів. Відтворюються дротові зображення на графопобудовувачах, в основному, на векторних графічних дисплеях. Найбільше розповсюдження вони отримали в інтерактивних графічних системах на базі векторних дисплеїв.

Контурні зображення з усуненими невидимими лініями складаються з видимих, з даної точки зору, ліній графічних моделей відтворюваних об'єктів. На них не зображені лінії або відрізки ліній об'єктів, закриті їхніми поверхнями або поверхнями інших об'єктів. Такі машинні зображення вимагають значного обсягу обчислень, що зростає приблизно в квадратичній залежності зі збільшенням складності і кількості відтворюваних об'єктів. Контурні перспективні зображення одержують, в основному, на векторних дисплеях для оцінювання проміжних результатів проектування або на графопобудовувачах для фіксування варіантів проектних рішень.

Півтіньові перспективні зображення характеризуються відтворенням тільки видимих поверхонь з градацією їхньої яскравості. Яскравість конкретної поверхні залежить від її розташування відносно джерела світла і спостерігача (точки зору), від тону (кольору) самої поверхні, її відбивальної властивості та інших чинників. Півтіньові перспективні зображення характеризуються високою реалістичністю і відображаються пристроями, в яких є можливість управляти інтенсивністю відображення відтворюваного масиву точок. Наприклад, вони відтворюються растровими дисплеями з градацією яскравості на кожній відтворюваній точці або групі точок. Зображення тіней поліпшує наочність півтіньових зображень і дозволяє більш якісно відтворювати пластику споруд, їхню форму, взаємне розташування.

Півтіньові зображення можна отримати за різними геометричними схемами, що характеризуються, передусім, розташуванням джерела світла. Найпростіший варіант – коли джерело світла збігається з точкою зору. При цьому яскравість частин поверхонь не залежить від їхньої відстані до точки зору, не враховуються фактура і тон поверхні. Більш складна схема припускає наявність джерела світла, розташування якого не збігається з

точкою зору, і враховує відстань від нього до об’єктів. Ті ж фактори впливають і на якість кольорових перспективних зображень.

Контрольні питання

1. В чому різниця між паралельним та перспективним проекціюванням?
2. Дайте характеристику обчислювальних затрат для різних видів перспективних зображень.
3. Як досягається ілюзія об’ємності при формуванні півтонових зображень?
4. Назвіть сфери застосування різних видів перспективних зображень.

4.2 Основні моделі освітлення

Модель освітлення – це математичне подання фізичних властивостей джерел світла та поверхонь, а також їх взаємного розміщення. Для моделювання освітлення тривимірних об’єктів використовуються різні моделі освітлення.

Проста модель освітлення базується на обчисленні інтенсивності відбитого об’єктом світла точкового джерела.

Відбиття світла об’єктом може бути дифузним або дзеркальним. Дифузне відбиття має місце при рівномірному розсіюванні світла, за рахунок чого створюється ілюзія, що поверхня має однакову яскравість незалежно від кута огляду. Розсіяне світло практично завжди присутнє в реальній обстановці.

Світло точкового джерела відбивається від ідеального розсіювача за законом косинусів Ламберта

$$I = I_L \times k_d \times \cos Q,$$

де I_L – інтенсивність джерела світла,

I – інтенсивність відбитого світла,

k_d – коефіцієнт дифузного відбиття,

Q – кут між напрямком світла та нормаллю до поверхні.

При освітленні об’єкта точковим джерелом на нього падає також і світло від сусідніх об’єктів. Ці світло буде розсіяним. Для обчислення інтенсивності розсіяного світла використовується формула

$$I = I_a + I_L k_d \cos Q,$$

де I_a – інтенсивність розсіяного світла,

k_a – коефіцієнт дифузного відображення розсіяного світла (вказаний коефіцієнт змінюється від нуля до одиниці).

Інтенсивність світла обернено пропорційна квадрату відстані до джерела.

Для врахування цієї відстані вводять коефіцієнт, який визначає відстань від центра проекції до об'єкта. Якщо центр проекції лежить близько до об'єкта, то параметр $1/d^2$ змінюється дуже швидко, що призводить до значного перепаду інтенсивності. У зв'язку з цим в розрахунковій формулі

$$I = I_f k_a + I_L k_d \cos Q/(d + k)$$

використовують не обернену квадратичну залежність $1/d^2$, а лінійне згасання $1/(d + K)$, де K – константа.

Дзеркально відбите світло не розсіюється. Кут відбиття його від ідеальної поверхні дорівнює куту падіння. В будь-якому іншому положенні спостерігач не бачить дзеркально відбите світло. При дзеркальному відбитті мають місце бліки.

Формула, яка враховує як дзеркально відбите, так і дифузне світло, має вигляд

$$I = I_f k_a + I_L k_d \cos Q/(Q + k_s \cos^n (l))/(d+k),$$

де k_s – константа, яка визначається експериментально,

l – кут між вектором відбитого променя та вектором спостерігача,

n – степінь, який апроксимує просторове розподілення дзеркально відбитого світла.

В комп'ютерній графіці така модель називається функцією зафарбовування і використовується для розрахунку інтенсивності тону.

Якщо є декілька джерел світла, то ефект, який вони створюють, підсумовується.

Обчислювальна складність алгоритму побудови об'єкта з використанням простої моделі освітлення досить висока. При кольоровому зображенні розрахунок інтенсивності для кожного компонента кольору виконується окремо, що приводить до суттєвих обчислювальних затрат. Для спрощення розрахунку інтенсивності кольору в точках об'єктів використовується інтерполяція.

Контрольні питання

1. Дайте характеристику простої моделі освітлення.
2. Наведіть формули для розрахунку інтенсивності розсіяного світла.
3. Наведіть формулу для інтенсивності світла, яка враховує як дзеркально відбите, так і дифузне світло.
4. Як враховується світло від декількох джерел?

4.3 Рендеринг Гуро та Фонга

Процес відтворення (візуалізації) об’єкта або сцени називається рендерингом (від англійського «rendering» – відтворення, передача візуалізації).

Рендеринг – це обчислення для кожного піксела зображення інформації про його колір і адресу, які дають можливість відтворити глибину об’єкта на двовимірному екрані. Рендеринг заповнює всі точки на поверхні об’єкта, що були попередньо збережені як набір вершин.

Основою для опису об’єктів у більшості пакетів тривимірної графіки є полігональні моделі. Для подання об’єкта у вигляді полігональної моделі він розрізається на плоскі ділянки – грані, які, в свою чергу, діляться на трикутники (цей процес називається тріангуляцією), у котрих деякі сторони збігаються з межами грані, а деякі лежать всередині її і при візуалізації не враховуються. Наприклад, для опису куба потрібно задати 12 трикутників (шість утворювальних площин і на кожній по два трикутники), а для опису об’єктів, що мають складну форму, може знадобитися декілька тисяч трикутників.

Найреалістичніші зображення одержують при використанні півтонових методів, які створюють ефект об’ємності за рахунок інтерполювання інтенсивності кольорів. Для одержання реалістичних зображень найчастіше використовують методи рендеринга: однотонний, метод Гуро і метод Фонга. Найбільше поширення одержав метод Гуро, тому що однотонне зафарбовування не забезпечує необхідної згладженості зображення, а більш реалістичне зафарбовування Фонга вимагає занадто великих обчислювальних затрат.

Для здійснення зафарбування методом Гуро потрібно провести розбиття об’єкта на полігони. Якщо при побудові полігональної поверхні для кожної грані використовується по одній нормалі, то створюється зображення, що складається з окремих багатокутників (рис. 4.1, а). Методом Гуро можна одержати згладжене зображення (рис. 4.1, б). Нормалі до поверхні апроксимуються у вершинах багатокутників. За вибраною моделлю освітлення визначається інтенсивність кольору вершин багатокутника, а потім, за допомогою інтерполяційних формул, обчислюється інтенсивність кожного піксела всередині багатокутника.

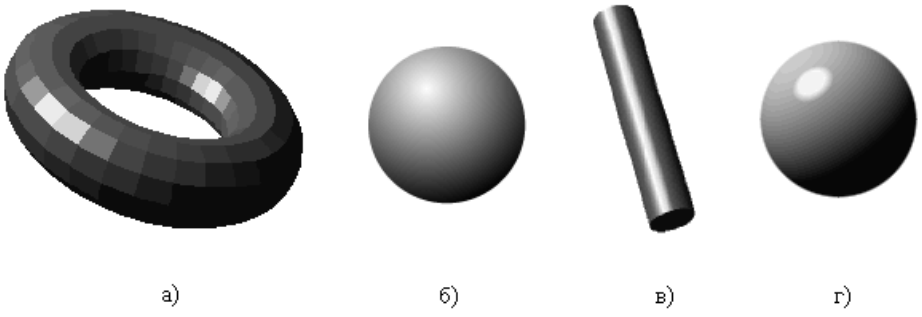


Рисунок 4.1 – Приклади синтезу об’ємних фігур:
а) однотонове заповнення; б) зафарбовування методом Гуро;
в, г) зафарбовування методом Фонга

При уважному розгляданні об’єкта, зафарбованого за методом Гуро, помітний прояв ефекту смуг Маха. Метод Гуро не завжди дозволяє уникнути різкої зміни інтенсивності на межових ребрах багатокутників, де і видно смуги Маха.

Ефект смуг Маха (рис. 4.2) зумовлений латеральним гальмуванням рецепторів ока, реакція яких на світло піддається впливу сусідніх рецепторів. Рецептори, розташовані безпосередньо на межі перепаду інтенсивностей з більш яскравої сторони, піддаються більш сильному подразненню, ніж ті, що знаходяться далі від межі. Це пояснюється тим, що вони менше загальмовуються своїми сусідами, що знаходяться з темнішого боку. Аналогічно, рецептори, розташовані безпосередньо на межі перепаду інтенсивності з більш темної її сторони, будуть піддаватися меншому впливу, ніж ті, що знаходяться в тій ж області, але далі від межі (рис. 4.2).

Рисунок 4.2 – Ефект смуг Маха

Зафарбовування Гуро найбільш доцільне при використанні простої моделі освітлення з дифузійним відбитком, тому що форма відблисків при дзеркальному відбитку суттєво залежить від вибору багатокутників, що подають об’єкт або поверхню.

У методі Фонга, як і в методі Гуро, спочатку апроксимуються нормалі до поверхонь у вершинах багатокутників. Далі переходять до інтерполяції значення вектора нормалі до поверхні всередині багатокутників.

Отримані значення вектора нормалі використовуються для визначення інтенсивності кожного піксела. Помітні поліпшення, порівняно з інтерполяцією, інтенсивності спостерігаються у випадку використання моделі дзеркального відбитка, тому що при цьому більш точно відтворюються світлові відблиски (див. рис. 4.1, в, г). Проте навіть якщо дзеркальний відбиток не використовується, інтерполяція векторів нормалі дає більш якісні

результати, ніж інтерполяція інтенсивності, оскільки апроксимація нормалі в цьому випадку здійснюється в кожній точці. Отже, утворюється більш реалістичне зображення, зокрема зменшується ефект смуг Маха.

Головним недоліком рендерингу Фонга є великі обчислювальні затрати, тому що для кожного піксела, за відомим вектором нормалі, обчислюється значення інтенсивності. Один із можливих підходів до скорочення кількості обчислень припускає використання так званої карти відбитків. Вона являє собою набір попередньо розрахованих інтенсивностей для визначеного діапазону значень вектора нормалі. У цьому випадку достатньо один раз обчислити карту відбитків, а потім, за відомим вектором нормалі, лише зчитувати з неї значення інтенсивності, не перераховуючи їх.

Вважається, що зафарбовування Фонга забезпечує вищу якість зображення, але потребує істотних обчислювальних затрат, тому для візуалізації складних сцен у реальному часі частіше застосовують алгоритм Гуро, який у даний час використовуються фактично в усіх комерційних робочих станціях.

В обох методах об’єкт, що зображується, подається у вигляді полігональної моделі. Для цього він розділяється на плоскі ділянки – грані, які, в свою чергу, як правило, діляться на трикутники. Така процедура дозволяє надалі, із застосуванням лінійної інтерполяції, легко розрахувати інтенсивності світла кожного складового піксела як всередині, так і на ребрах трикутника. Як елементарні полігони трикутники використовуються частіше за інші через простоту їхньої геометрії і розрахункових формул.

Розглянемо більш детально процес зафарбовування області, обмеже-ної одним трикутником. Геометрія трикутника задається координатами вершин (A, B і C), за значеннями яких визначають прирости координат для кожної сторони трикутника. Для методу Гуро задаються значення інтенсивності у вершинах – Ia, Ib, Ic. (рис. 4.3). Необхідно обчислити інтенсивність всіх інших точок усередині даного трикутника і на його ребрах. Для цього знайдемо інтенсивність довільно обраної точки D усередині полігона.

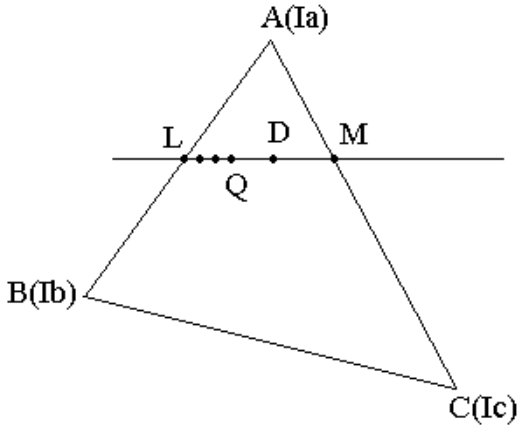


Рисунок 4.3 – Зафарбовування методом Гуро

Визначимо приріст інтенсивності світла на один піксел уздовж ребер трикутника AB, BC, AC:

$$\Delta I_{AB} = (I_A - I_B) / БП_{AB}, \quad (4.1)$$

$$\Delta I_{AC} = (I_A - I_C) / БП_{AC}, \quad (4.2)$$

$$\Delta I_{BC} = (I_B - I_C) / БП_{BC}, \quad (4.3)$$

де БП – більший з приростів координат Δx, Δy обраного відрізка прямої.

Проведемо через точку D лінію, паралельну горизонтальній осі. Вона перетне ребра трикутника в точках L і M. За приростами інтенсивності уздовж ребер визначимо інтенсивність точок, що належать цим ребрам.

Інтенсивність точок M та L може бути знайдена так:

$$I_M = I_A + \Delta I_{AB} \times БП_{AM}, \quad (4.4)$$

$$I_L = I_A + \Delta I_{AC} \times БП_{AL}. \quad (4.5)$$

Далі, за аналогією з попередніми діями, визначимо приріст інтенсивності вздовж сканувальної прямої LM (у даному випадку $LM = БП_{LM}$):

$$\Delta I_{LM} = (I_L - I_M) / LM. \quad (4.6)$$

Інтенсивність світла в точці D знайдемо у такий спосіб:

$$I_D = I_L + \Delta I_{LM} \times LD. \quad (4.7)$$

Аналогічним способом визначається інтенсивність будь-якої іншої точки, обмеженої заданим полігоном.

Найменша похибка досягається при розбитті поверхні на прямокутні трикутники з катетами, паралельними координатним осям, оскільки в цьому випадку більший приріст (БП) катета збігається з його довжиною.

При зафарбовуванні всього трикутника сканувальний рядок послідовно проходить усі точки полігона. Приріст, як уздовж сканувальної лінії, так і між сусідніми сканувальними лініями, у цьому випадку складає один піксел, що дозволяє операції множення замінити операцією накопичувального додавання.

$$I_Q = I_L + \Delta I_{LM} + \Delta I_{LM} + \Delta I_{LM}. \quad (4.1.8)$$

На рисунку 4.4, а наведений приклад попікселного зафарбовування прямокутного трикутника при Ia = 5, Ib = 9, Ic = 1. Значення інтенсивностей світла отримані за описаними вище формулами (рис. 4.4, б).

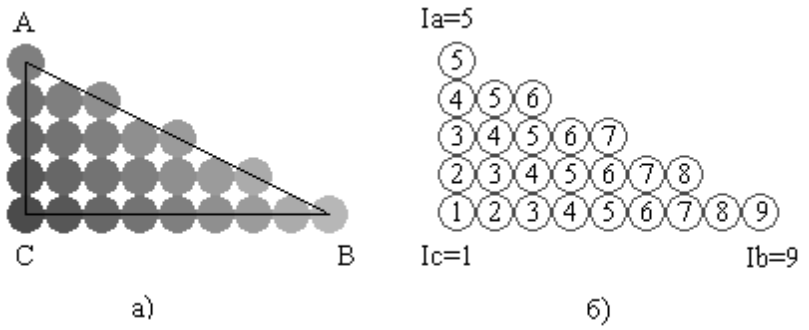


Рисунок 4.4 – Приклад реалізації рендерингу Гуро

Наведені формули інтерполяції використовуються також і в зафарбовуванні Фонга з тією лише різницею, що для кожної вершини трикутника задається не інтенсивність світла, а вектор нормалі.

4.4 Алгоритми видалення невидимих поверхонь

Формування реалістичних зображень на екрані комп'ютера передбачає видалення невидимих ліній і поверхонь тривимірних об'єктів. Ця процедура належить до найбільш поширених і вважається класичною задачею машинної графіки, але, в той же час, вона є і однією з найбільш трудомістких.

Відомо, що для формування об'ємних зображень використовуються каркасний, контурний і півтоновий методи.

У каркасних конструкціях (рис. 4.5) видимими є всі лінії конструкції. Об'ємність основана на перспективному ефекті, що створює ілюзію глибини зображення. Такий підхід вимагає найменших обчислювальних витрат, однак прийнятний тільки для відносно нескладних конструкцій, оскільки зі збільшенням числа граней наочність зображення погіршується.

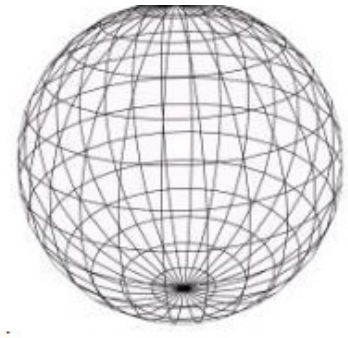
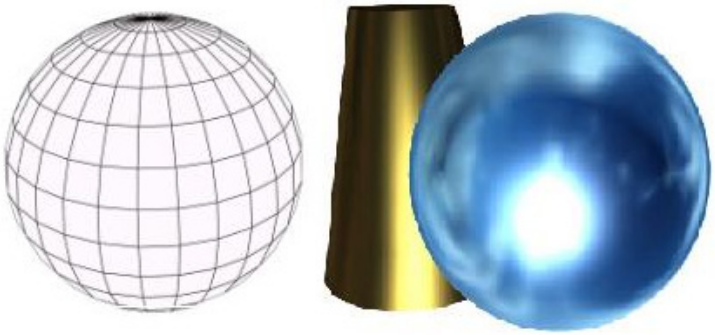


Рисунок 4.5 – Каркасна модель

Реалістичність зображень можна істотно підвищити шляхом видалення невидимих ліній і поверхонь. Причому, при видаленні невидимих ліній використовують, в основному, каркасне подання об’єктів, а в контурному (рис. 4.6) і півтоновому методах (рис. 4.6, б), як правило, мова йде про видалення невидимих поверхонь об’єктів, закритих або їх власними поверхнями, або поверхнями інших об’єктів.



а) б)

Рисунок 4.6 – Контурна (а) і півтонова (б) моделі

До теперішнього часу розроблено велику кількість алгоритмів видалення невидимих ліній і поверхонь. Серед них можна виділити придатний для різних застосувань. Вибір конкретного алгоритму залежить від низки факторів:

яка модель прийнята для опису зображуваних об’єктів;

який вид об’єктів; яка потрібна точність їх подання;

які обмеження на обсяг обчислень і пам’яті.

Скоротити обчислювальні витрати можна за рахунок низки спрощень, серед них: креслення тільки контурів граней об’єктів; апроксимація контурів ламаною лінією; використання тільки плоских граней..

Завдання також зазнає суттєвого спрощення, якщо використовувати ортогональное проєктування, оскільки при центральному проєктуванні потрібно визначати точки перетину граней об’єкта з променями, що виходять із точки спостереження, а це вимагає додаткових обчислень.

Алгоритми

Історично одним з перших був запропонований алгоритм Галімберті і Монтанарі. У ньому зіставлялося положення кожного ребра сцени з усіма гранями сцени, тобто, при видаленні невидимих ліній використовувався чисто геометричний підхід.

Основна ідея іншого алгоритму Варнока полягає в тому, що для аналізу видимості або невидимості елементів зображення здійснюється

послідовне розбиття об’єктного простору на складові частини. При такому підході з кожною ітерацією завдання видалення невидимих елементів сцени спрощується. В алгоритмі передбачено низку стандартних ситуацій для різних випадків взаємних розташувань граней об’єкта, при якому видалення невидимих частин вирішується. Об’єктний простір умовно просікають трубою прямокутного перерізу. Якщо сцена, що потрапила в трубу, не належить до жодної зі стандартних ситуацій (порожнеча в трубі, одна грань в трубі, грань, «пробита» трубою тощо), то трубу ділять ще на чотири підпростори з перерізами, що являють собою чотири частини вихідного перерізу та знову виконують зіставлення зі стандартними ситуаціями. Зрозуміло, що при поступовому зменшенні площі перерізів ймовірність зустріти стандартну ситуацію зростає. У найгіршому випадку поділ виконують до тих пір, поки розтин не зменшиться до однієї точки. Приклад виконання однієї ітерації алгоритму Варнока показаний на рис. 4.7.

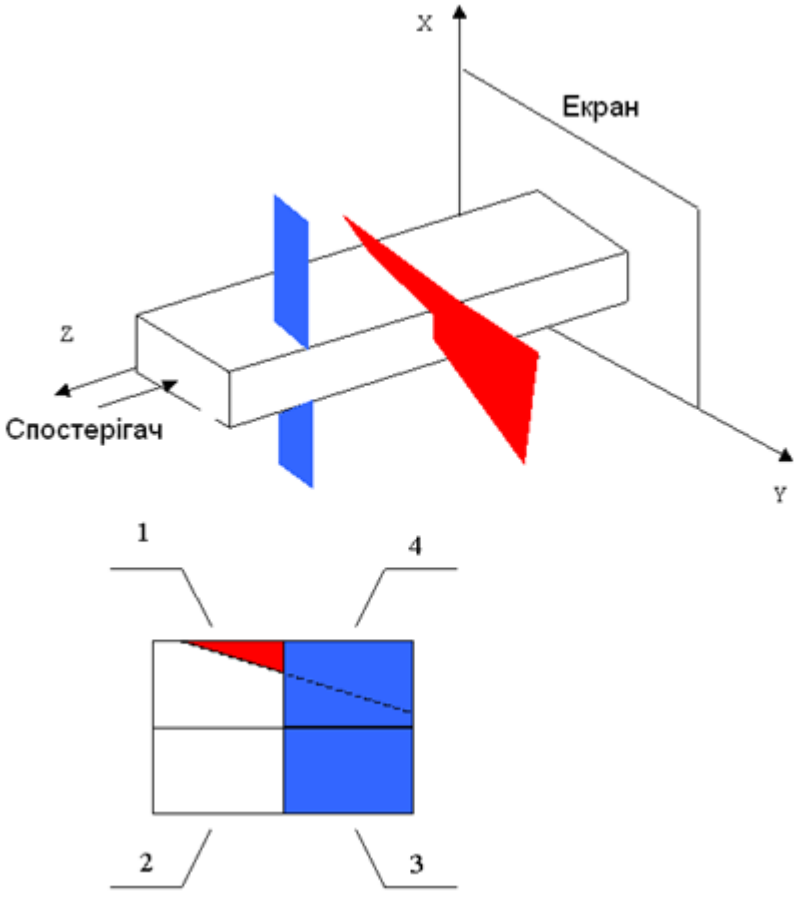


Рисунок 4.7 – Принцип алгоритму Варнока

Можна вважати, що отриманий розтин не належить до жодної зі стандартних ситуацій, тому проводиться його поділ на 4 частини (в загальному випадку розбиття можна робити на будь-яке число).

У перших трьох підвікнах ситуації стандартні. Ці вікна або порожні, або в них потрапляє тільки по одному об’єкту. Так, підвікно 2 порожнє, і воно зображується на екрані кольором фону. Аналогічно, підвікно 3 повністю «просікає» тільки одну грань об’єкта, тому воно зображується на екрані кольором цієї межі. У підвікні 1 знаходиться тільки один багатокутник, отже він є видимим. Площа поза цим багатокутником заповнюється фоновим кольором, а сам багатокутник – кольором його межі.

У перетин підвікна 2 потрапило дві грані, причому одна з них частково закриває перетин, і оскільки ситуація нестандартна, це підвікно потрібно розділити ще на 4 частини і знову виконати описані вище дії, але це буде вже наступна ітерація. Кількість ітерацій алгоритму Варнока прямо пропорційна насиченості зображення. Зі збільшенням числа стандартних ситуацій кількість поділів на підпросторі зменшується.

Алгоритм Варнока можна використовувати для роботи як з векторними, так і з растровими зображеннями.

Щоб ефективніше виконувати видалення невидимих ліній, доцільно скористатися методом сортування граней об’єктів сцени за їх віддаленістю від точки спостереження. Кінцевим результатом такого сортування є список об’єктів сцени, упорядкованих за їх віддаленістю від точки спостереження.

Елементи зображення записуються в відеопам’ять, починаючи з найбільш віддаленого. Тоді елементи, розташовані ближче до спостерігача, будуть затирати інформацію про елементи, розташованих далі.

Розглянемо для прикладу сцену, яка складається з чотирьох граней (рис. 4.8). Нехай грань під номером 4 найбільш віддалена від спостерігача, а грань під номером 1 знаходиться ближче інших. ображення будується, починаючи з межі, яка найбільш віддалена (в нашому прикладі це грань 4). Потім виводяться по черзі межі 3, 2, 1.

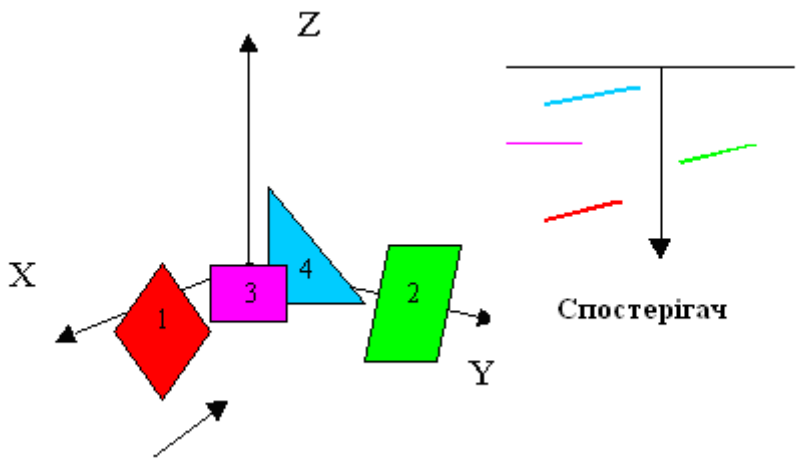


Рисунок 4.8 – Метод сортування за глибиною

При реалізації алгоритму потрібно враховувати прозорість граней. Можливі випадки, коли нова грань не перекриває попередню грань або нова грань частково чи повністю перекриває попередню грань. При варіанті, коли грань не перекривається іншими гранями, вона виводиться без зміни (у прикладі це грань під номером 2). Коли нова грань частково або повністю перекриває попередню грань, то для формування зображення замінюється загальна частина (коли нова грань непрозора), або здійснюється комбінування кольорів (якщо нова грань прозора). Алгоритм забезпечує формування реалістичних сцен за рахунок можливості комбінування кольорів в кожній точці зображення.

В алгоритмі Уоткінса тривимірна задача видалення невидимих ліній і поверхонь зводиться до двовимірної. Для цього сцену розтинають площинами, перпендикулярними до поверхні екрана, й виконується аналіз отриманих в перерізі проекцій. Алгоритм виконується по рядках розгортки розгортання екрана. Площини перерізу називають площинами розгортання. Зображення в площинах розгортки мають вигляд набору відрізків прямих, що являють собою ортогональні проекції граней тривимірних об’єктів. Для формування зображення в кожній площині розгортки здійснюється аналіз відрізків прямих і визначається їх положення відносно спостерігача. Завдання видалення невидимих граней зводиться до визначення того, який відрізок бачимо для кожної точки рядка сканування. Відрізки, розташовані ближче до спостерігача, закривають повністю або частково відрізки, розташовані далі (рис. 4.9).

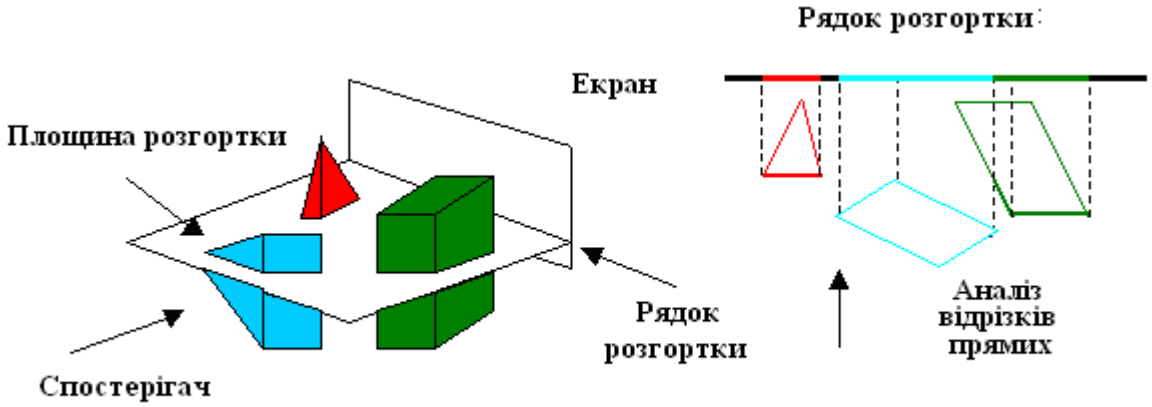


Рисунок 4.9 – Принцип роботи алгоритму Уоткінса

Відомо декілька підходів, що прискорюють роботу алгоритму. Так, в одному з них визначається видимість відрізків на кожному з інтервалів, отриманих шляхом ділення рядка сканування проекціями точок перетину ребер. Алгоритм Уоткінса одним з перших був реалізований апаратно.

Метод трасування променів, розроблений на початку 80-х років минулого століття, дозволяє отримувати найбільш реалістичні зображення шляхом моделювання прозорості, відображення, заломлення та інших оптичних ефектів. Основна ідея методу викладена нижче. Світло, що випромінюється деяким джерелом, досягає спостерігача, відбившись від деякої поверхні. За результатами розрахунку інтенсивності світла, що досягло спостерігача, можна визначити невидимі поверхні (світло не досягає спостерігача) або визначити ступінь прозорості об'єкта, визначивши різницю інтенсивностей світла від джерела і інтенсивності світла, яке досягло спостерігача. Однак повний розрахунок променів, що виходять з усіх джерел, називаний прямим трасуванням (raytracing), через величезну кількість променів застосовується досить рідко. Крім того, велика частина роботи з простежування променів, що не потрапили в око спостерігача, для визначення освітленості невидимих поверхонь виявиться проведеною даремно.

Для зниження обчислювальних витрат відстежують промені в зворотному напрямку (raycasting), тобто, від спостерігача до об'єкта, як показано на рис. 4.10. За перетином променів трасування і об'єктів визначають видимі поверхні.

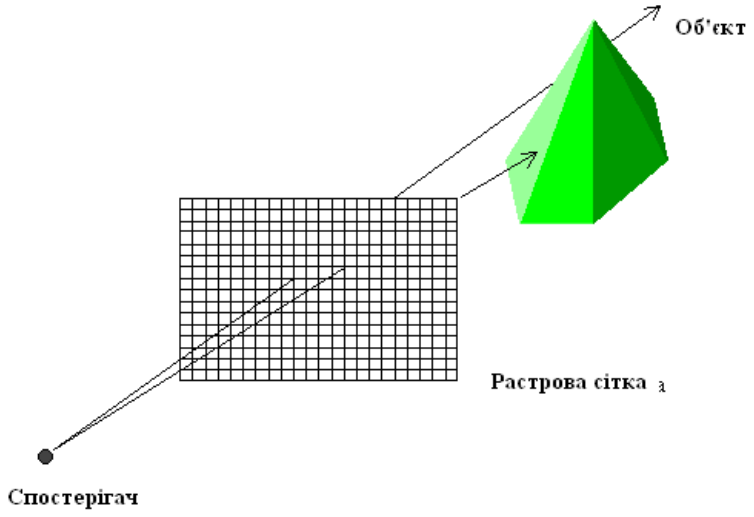


Рисунок 4.10 – Принцип роботи алгоритму трасування променів

Джерело світла знаходиться в нескінченності на осі z, тому всі промені, що йдуть від нього, паралельні їй (осі z). Кожному променю, що виходить з джерела світла, відповідає свій піксел на екрані. Промені проводяться від спостерігача через гіпотетичну растрову сітку екрана і далі у відображуваний простір.

Траєкторія кожного променя відстежується, щоб визначити, які саме об'єкти зображення, якщо такі існують, вона перетинає. Якщо промінь

перетинає об’єкт, то визначаються всі можливі точки перетину променя і об’єкта. Ці точки впорядковуються за глибиною. Точка перетину з максимальним значенням глибини вказує на видиму поверхню для відповідного піксела. Якщо жодна з поверхонь не була перерізана променем, то пікселу присвоюється колір фону.

Розрахунок променів, що проходять через всі вузли растрової сітки, виконується для кожної з трьох колірних складових: Red, Green, Blue. Його результати заносяться в кадрови буфер, звідки виводяться на екран. Оскільки 75–95% часу, що витрачається алгоритмом на трасування променів, витрачається на визначення перетинів променя трасування з об’єктом, то прагнуть зменшити обчислювальну складність цієї процедури. Наприклад, реальні об’єкти умовно поміщають в прості оболонки (паралелепіпед або сферу) і шукають точку перетину променя з цими оболонками (рис. 4.11). Якщо промінь не перетинає оболонку, значить він, не перетинає і сам об’єкт. Безумовно, задання граней оболонки істотно простіше, ніж граней самого об’єкта.

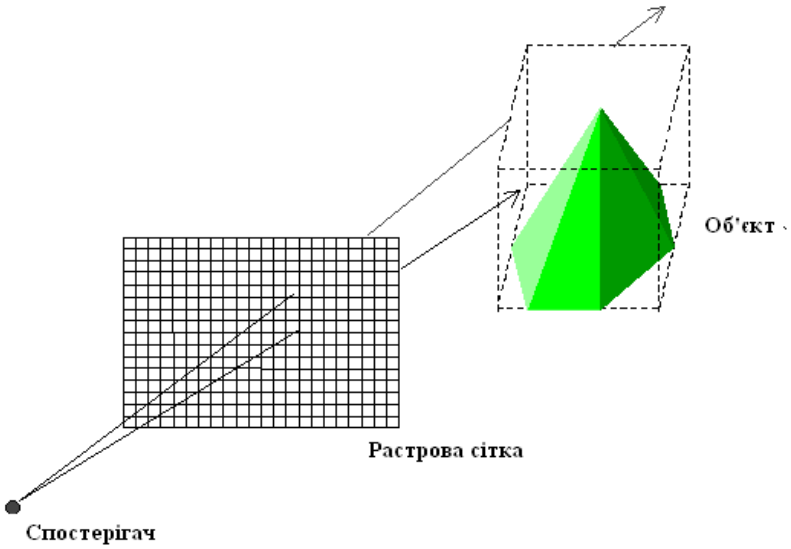


Рисунок 4.11 – Використання оболонок в методі трасування променів

Максимальна «глибина» трасування променів або задається користувачем, або визначається самою програмою-трасувальником залежно від наявності вільної пам’яті. Алгоритм трасування променів вимагає великого обсягу обчислень, однак схемотехнічна та алгоритмічна простота його реалізації, можливість виконання незалежних паралельних обчислень для всіх елементів зображення і можливість отримання реалістичних зображень із заданими оптичними властивостями визначають досить широке його використання.

У графічних акселераторах найбільшого поширення набув алгоритм видалення невидимих поверхонь з використанням Z-буфера (рис. 4.12). Для реалізації алгоритму, крім традиційної відеопам’яті для зберігання кадру зображення, використовують двовимірний масив пам’яті для зберігання глибини (z-координати) кожного видимого елемента зображення.

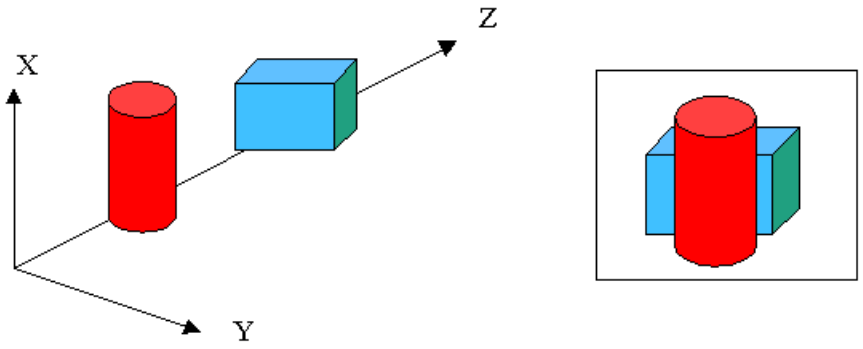


Рисунок 4.12 – Принцип роботи алгоритму
з використанням Z-буфера

При розкладанні в растр елементів зображення обчислюється значення глибини z для кожної точки. Якщо значення глибини z поточної точки зображення менше, ніж значення z точки, яка раніше зберігалася в відеопам’яті за відповідною адресою, то у відеопам’ять заносять значення поточної точки, а в Z-буфер – її глибину. В цьому випадку поточний елемент зображення перекриває вихідний. В іншому випадку модифікація відеопам’яті і Z-буфера не виконується.

Для реалізації високоякісного рендерингу потрібна велика розрядність Z-буфера. Чим вона вища, тим вище дискретність z-координат і тим точніше виконується рендеринг вилучених обєктів. Якщо 24-розрядний Z-буфер забезпечує більше 16,7 млн рівнів задання глибини, а 32-розрядний – понад 2,8 млрд, то 16-розрядний – тільки 65536. Коли при рендерингу роздільної спроможності не вистачає, може статися, що два об’єкти отримають одну і ту ж z-координату, в результаті не буде встановлено, який об’єкт ближче до спостерігача, що може викликати так званий артефакт, або triangle disposition.

У сучасних графічних акселераторах використовують виділений для Z-буфера блок пам’яті. У деяких з них для здешевлення конструкції для цієї функції використовують основний буфер, що, безумовно, позначається на продуктивності.

Z-алгоритм має просту апаратну реалізацію, не вимагає попереднього сортування примітивів, робить тривіальну візуалізацію перетинань складних

поверхонь. Недоліки алгоритму полягають в необхідності введення додаткового об’єму пам’яті, а також в складності реалізації ефектів прозорості і просвічування; усунення сходового ефекту аліайзингу.

У OpenGL, одному з найпопулярніших програмних інтерфейсів (API) для розробки додатків в області двовимірної і тривимірної графіки, закладено видалення невидимих ліній і поверхонь за алгоритмом з використанням Z-буфера.

Застосування конкретного алгоритму для видалення невидимих частин зображення визначається багатьма факторами, наприклад, типом зображуваних об’єктів, необхідним ступенем реалістичності, розташуванням джерел світла, доступними обсягами пам’яті, обчислювальною складністю, простотою апаратної реалізації і т. д. У різних випадках ефективними можуть бути ті чи інші алгоритми. У зв’язку з цим в низці програмних продуктів використовують одночасно кілька алгоритмів видалення невидимих поверхонь.

Контрольні питання

1. В чому полягає процедура рендерингу?
2. В яких випадках використовують рендиринг Гуро та Фонга?
3. Які обчислювальні дії охоплює рендеринг Гуро?
4. Як обчислюється інтенсивність кольору точок згідно з редерингом Фонга?

5 ПРИНЦИПИ ВІДОБРАЖЕННЯ ІНФОРМАЦІЇ

5.1 Растровий принцип відображення інформації

В системах відображення інформації (СВІ) на даний час найчастіше використовують растровий метод. Засоби відображення, що використовують растр, називають «СВІ растрового типу».

Сутність растрового методу формування зображення полягає в тому, що для керування електронним променем використовується розгорнення, а підсвічування елементів зображення на екрані здійснюється при русі променя по рядках у відповідні проміжки часу.

На рис. 5.1 зображений растр, утворений лінійним прогресивним розгорненням, при якому повний растр утворюється за один період кадрового розгорнення $T_к$. Розгортка зображення створюється одночасним рухом променя по горизонталі вздовж осі X і по вертикалі вздовж осі Y. Рух променя по горизонталі називають рядковим розгорненням, а накреслені при цьому лінії – рядками. Переміщення променя по вертикалі називають кадровим розгорненням, в результаті якого всі рядки розміщуються один над одним. Рядкове та кадрове розгорнення здійснюються формуванням відхиляльних сигналів X, Y (рис. 5.2) напруг для електростатичної відхиляльної системи або струмів для магнітної. Відхиляльні сигнали формуються генераторами рядкового та кадрового розгорнення.

Частота кадрового розгорнення $f_k = 1/T_k$ для ЕПТ з малим часом післявисвічування має бути більша критичної частоти мерехтіння. Як правило, частоту f_k вибирають рівною частоті мережі змінного струму, усуваючи цим ефекти переміщення по екрану утворюваних нею завад. Частота f_z і період T_z рядкового розгорнення ($f_z = 1/T_z$) вибирають з умови

$$F_z = Z \times f_k,$$

де Z – число рядків в кадрі, що визначають розподільну спожність СВІ по вертикалі. В телебаченні стандартом прийнято Z = 625. У високоякісних СВІ розповсюджене так зване багаторядкове розгорнення з Z = 1000 і більше.

Період рядкового розгорнення T_z охоплює час прямого ходу променя по рядку $T_{zп}$ і часу зворотного ходу T_{zo}. Зображення формується за час прямого ходу.

Відношення $T_{zo} / T_z = \alpha_z$ називається коефіцієнтом зворотного ходу рядкового розгорнення. Відповідно, при відомих значеннях T_z і α_z визначається $T_{zп} = T_z \times (1 - \alpha_z)$. Для стандарту телебачення $\alpha_z = 0.18$.

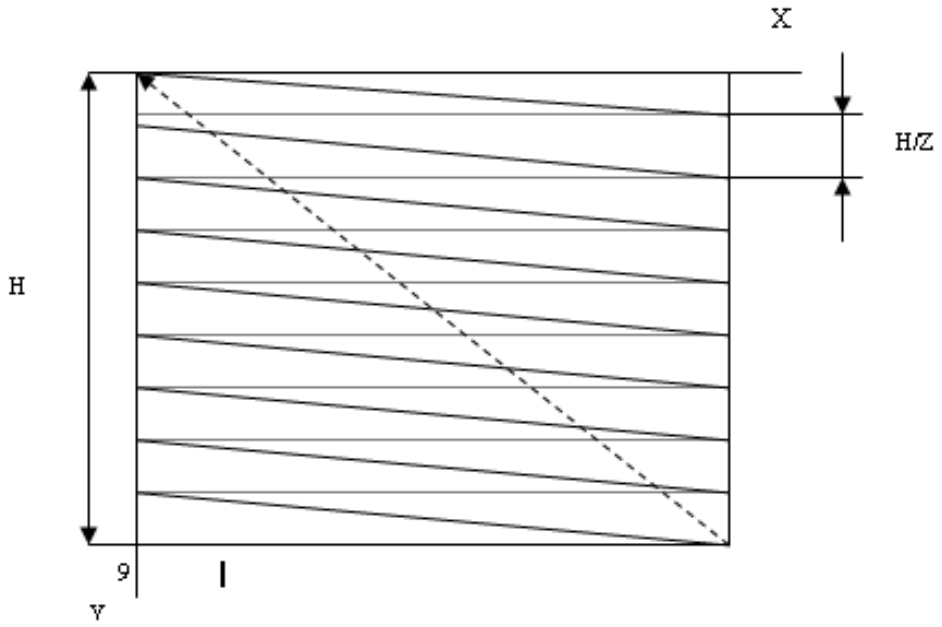


Рисунок 5.1 – Прогресивне растрове розгорнення

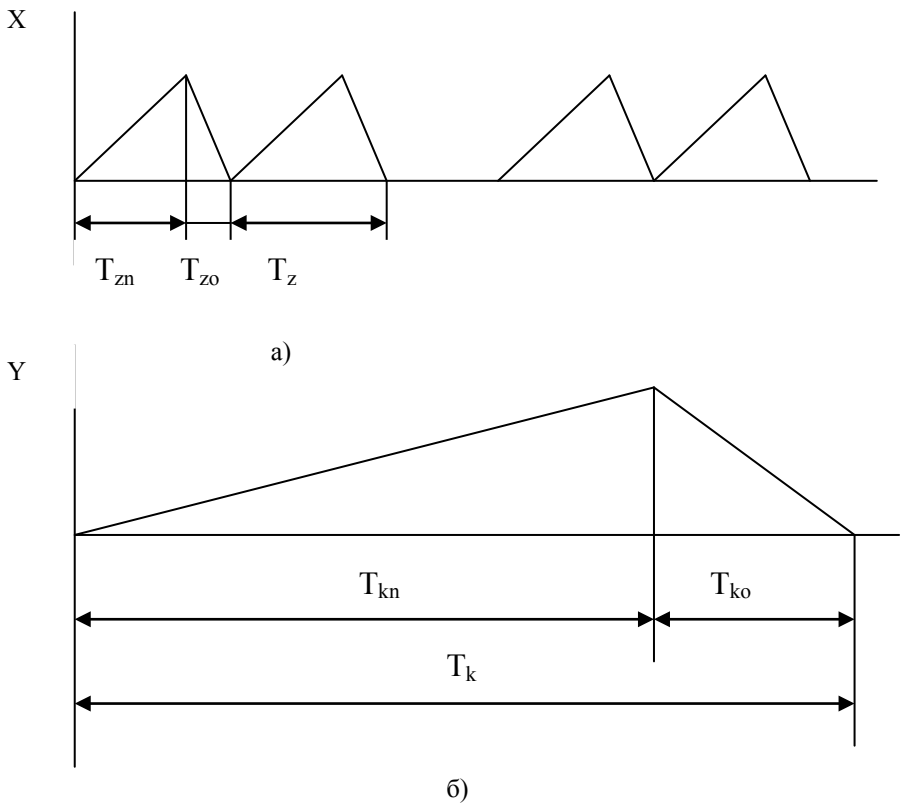


Рисунок. 5.2 – Часові діаграми рядкового X та кадрового Y розгорнення

Період кадрового розгорнення

$$T_к = T_{ко} + T_{кп},$$

де $T_{кп}$ і $T_{ко}$ – час прямого і зворотного ходів кадрового розгорнення.

Відношення $T_{ко}/T_k = \alpha_k$ називається коефіцієнтом зворотного ходу кадрового розгорнення. Число телевізійних рядків, що формуються за час прямого ходу променю, визначається за формулою

$$Z_n = (1 - \alpha_k) \times Z.$$

Для стандарту телебачення $\alpha_к = 0.08$.

На рис 5.3 показаний телевізійний растр, що утворений черезрядковим растром, який передбачає формування одного кадру зображення з двох полів, що передаються послідовно.

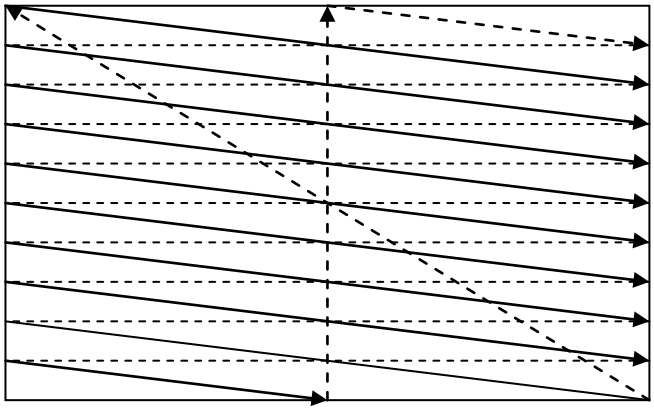


Рисунок 5.3 – Растр при черезрядковому розгорненні

В першому полі викреслюються непарні, а в другому – парні рядки растра (останні на рисунку показані штрихпунктирними лініями).

Дискретне зміщення зображення на один рядок в кожному полі не фіксується оком через інертність до сприйняття переміщення об’єктів в полі зору, якщо частота зміни зображень не менше 15–16 Гц. Тому при виборі частоти кадрів $f_k = 25$Гц забезпечується злитість сприйняття зображення двох полів. В той же час відтворення зображення в кожному полі з частотою $f_п = 2f_k = 50$Гц усуває мерехтіння яскравості, оскільки виконується вимога $f_p >= f_{кчм}$.

Зменшення частоти кадрів в два рази порівняно з прогресивним розгорненням при тому ж числі рядків в кадрі приводить до подвійного зменшення частоти рядкового розгорнення і смуги пропускання відеопідсилювача, що є необхідним. Для формування черезрядкового

розгорнення потрібно забезпечити такі умови: число рядків в кадрі має бути непарним, оскільки $Z = 2m + 1$, де m – ціле число; частоти рядкового розгорнення мають бути жорстко зв'язані між собою умовою $2f_z = Zf_п = (2m + 1)f_п$.

В результаті виконання цих умов друге поле починається з половини рядка і всі рядки виявляються зміщеними по вертикалі відносно рядків першого поля.

Черезрядкове розгорнення використовується в телевізійних трансляціях і в низці промислових телевізійних установок. В СВІ рекомендується використовувати прогресивне розгорнення, при якому відсутні черезрядкові мерехтіння, які призводять до втоми зору оператора.

До переваг СВІ растрового типу відносять: універсальність, яка дозволяє відображати всі види ІМ; можливість суміщення інформаційних моделей, які формуються методом електронного синтезу (знакогенерації), з півтоновими зображеннями, що отримуються за допомогою телевізійних камер.

При формуванні зображення найменшими елементами є точки. Вони розміщуються вздовж кожного рядка на фіксованих місцях. Для монохромних індикаторів кожній точці зображення відповідає комірка запам'ятовувального пристрою, куди заносять 1 або 0 залежно від того, має точка підсвічуватись чи ні. Кількість рядків і точок в рядку, як правило, вибирають кратним $2^к$, що відповідає структурі мікросхем пам'яті та суттєво спрощує схему інтерфейсу.

Зображення на екрані формують відеосигналами, які для кожної точки передають інформацію, що її зчитують з відеопам'яті синхронно з розгорненням.

Для отримання півтонового зображення відеопам'ять має кілька однакових шарів. При формуванні зображення дані зчитуються паралельно зі всіх шарів і подаються на цифроаналоговий перетворювач з числом розрядів, яке відповідає кількості шарів.

Розподільна спроможність кольорових моніторів визначається, на відміну від чорно-білих, фізичними розмірами тріад люмінофорних зерен на екрані.

При синхронному формуванні зображення значення прямих кодових елементів (ПКЕ) формуються синхронно з ходом рядкового розгорненням і подаються на входи керування (R, G, B) ТВ монітора. Система може виконувати зміну кадрів зображення з частотою ТВ розгорнення, однак цей спосіб характеризується значними апаратними затратами.

Асинхронний спосіб передбачає буферизацію масиву значень ПКЕ елементів зображення в відеопам'яті (ВП). Розподіл процедур формування і регенерації зображення знімають обмеження на складність зображень, що формуються. Друга важлива перевага – можливість реєстрації і відображення швидкоплинних процесів. В більшості випадків інтерес являє не сам процес,

а кінцевий результат чи траса станів, яка може бути зображена зі значно більшою точністю, ніж в синхронних системах.

Висока продуктивність асинхронних систем генерації зображень забезпечується: використанням високопродуктивних графічних процесорів (ГПР); додатковою обробкою ПКЕ безпосередньо перед виведенням на монітор.

Контрольні питання

1. Які типи розгорток вам відомі?
2. Вкажіть основні параметри прогресивного розгорнення.
3. В яких випадках використовують синхронне та асинхронне растрові розгорнення?
4. Як організовується модифікація відеопам’яті при растровому розгорненні?

5.2 Векторний принцип відображення інформації

У векторних пристроях відображення інформації зображення отримують у вигляді набору векторів, дуг, точок і символів, які адресуються в заданій системі координат. При цьому використовується автономна пам’ять, яка зберігає координати точок, кінців векторів або коди знаків, а також команди, які визначають режим роботи пристрою. Інформація з пам’яті послідовно надходить в блок керування дисплеєм, причому координати, які зчитуються, вказують точку, в яку необхідно встановити електронний промінь. При цьому промінь переміщується в нове положення від точки, яка була вказана попередніми кодами. Залежно від заданої команди під час руху електронного променя його траєкторія висвічується на екрані (тоді відображається відповідний вектор, дуга або висвічується тільки кінцева точка). Можливий режим переміщення променя без підсвічування.

Основний недолік наведеного підходу полягає в обмеженій складності зображення, яке може бути виведеним на екран без мерехтіння.

При векторному принципі формування зображення виконується довільне сканування. Це означає, що відрізок прямої може бути накресленим безпосередньо від однієї точки до іншої.

У векторному дисплеї на запам’ятовувальній електронно-променевій трубці використовують люмінофор з великим часом післясвітіння. Щоб нарисувати відрізок на екрані, інтенсивність променя збільшують до такої величини, яка призводить до запам’ятовування на люмінофорі. Для стирання зображення на всю трубку подають спеціальну напругу, яка знімає свічення люмінофору. Витерти окремі елементи зображення неможливо як і організувати динамічний рух чи анімацію.

В дисплеї на ЗЕПТ через деякий час виконують перерисування зображення.

У векторному (рисує відрізки чи вектори) дисплеї з регенерацією зображення на базі ЕПТ використовується люмінофор з дуже коротким часом післясвітіння. Такі дисплеї часто називають дисплеями з довільним скануванням. Через те, що час післясвітіння люмінофора малий, зображення на ЕПТ за секунду має багаторазово перерисовуватись чи регенеруватись. Мінімальна швидкість регенерації має складати принаймні тридцять (1/с), а краще від 40 до 50 (1/с). Швидкість регенерації, що менша 30 (1/с), приведе до того, що зображення буде мерехтіти, як це буває, коли кінофільм прокручується надто повільно. На таке зображення неприємно дивитись і його важко використовувати.

Для векторного дисплея з регенерацією потрібно, крім ЕПТ, ще два елементи: дисплейний буфер та дисплейний контролер. Дисплейний буфер – це неперервна ділянка пам’яті, яка містить всю інформацію, необхідну для виведення зображення на ЕПТ. Функція дисплейного контролера полягає в тому, щоб циклічно обробляти цю інформацію зі швидкістю регенерації. Складність (число зображуваних векторів) рисунка обмежується двома факторами – розміром дисплейного буфера і швидкістю дисплейного контролера. Ще одним обмеженням є швидкість обробки геометричної інформації.

На рис. 5.4 наведено блок-схему двох високопродуктивних векторних дисплеїв.

Рисунок 5.4 – Концептуальні блок-схеми векторних дисплеїв з регенерацією

В обох випадках припускається, що такі геометричні перетворення, як поворот, перенесення, масштабування, перспективне проекціювання та відсікання, реалізовані апаратно в геометричному процесорі. В першому випадку (рис. 5.4, а) геометричний процесор працює повільніше, ніж це необхідно при регенерації зображень (від 4000 до 5000 векторів). Таким чином, геометричні дані, що посилаються центральним процесорним пристроєм (ЦПП) графічному дисплею, обробляються до зберігання в дисплейному буфері. Отже, в ньому зберігаються тільки ті інструкції, які необхідні генератору векторів та літер для виведення зображення.

Дисплейний контролер зчитує інформацію з дисплейного буфера і посилає її генератору векторів та літер. При досягненні кінця дисплейного буфера контролер повертається на його початок, і цикл повторюється знову.

В другій схемі (рис. 5.4, б) геометричний процесор працює швидше, ніж необхідно для регенерації достатньо складних зображень. В цьому випадку вихідна геометрична база даних, послана з ЦПП, зберігається безпосередньо в дисплейному буфері, а вектори, як правило, задаються в координатах користувача у вигляді чисел з плаваючою точкою. Дисплейний контролер за один цикл регенерації зчитує інформацію з дисплейного буфера, пропускає її через геометричний процесор і результат передає генератору векторів. При такому способі обробки геометричні перетворення мають виконуватися протягом одного циклу регенерації.

Матричний екран, як правило, утворений незалежними дискретними елементами, що дозволяє підсвічувати довільно вибраний елемент. Розглянемо для прикладу газорозрядну індикаційну панель (ГІП) постійного струму.

Конструктивно ГІП постійного струму являє собою діелектричну платівку 4 з отворами-комірками (рис. 5.5). З двох сторін від решітки розташовані системи рівнобіжних електродів *2* і 5*,* що перехрещуються під прямим кутом. Панель має захисні стекла 1 і 6, а корпус – герметизований. Комірки 3 розташовані в місцях перехрещування електродів, заповнені інертним газом (неон, суміш неону з азотом або неону з аргоном і т. д.) і утворять мініатюрні газорозрядні прилади, у яких одна система електродів виконує функцію катодів, а інша – функцію анодів. Якщо на одну з пар «анод–катод» подати напругу, величина якої перевищує напругу пробою, то в комірці, розташованій в місці їх перехрещування, виникає тліючий розряд, і комірка світиться. Напруга горіння комірки менша напруги запалювання. Якщо напругу подавати по черзі до в одній горизонтальній шині, то комірки цього рядка запалюються по черзі, і точка переміщається з одного краю ГІП до іншого.

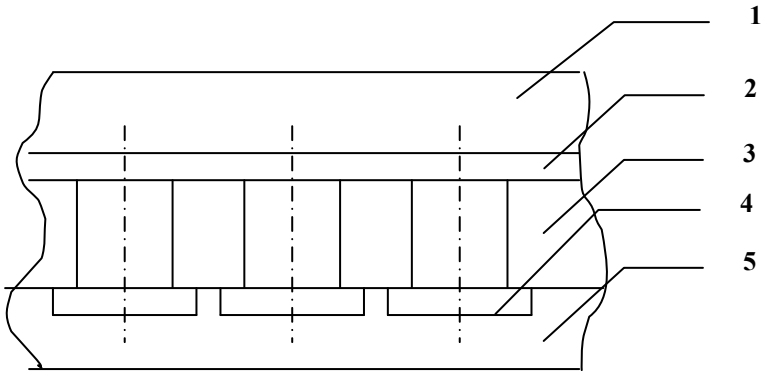


Рисунок 5.5 – Газорозрядна панель постійного струму

Таким чином, підключаючи періодично з визначеною частотою необхідні комірки, можна одержати зображення потрібного знака. ГІП постійного струму не мають пам’яті, внаслідок чого для одержання зображення потрібно періодично подавати імпульси керування послідовно на всі рядки панелі.

ГІП змінного струму відрізняється від ГІП постійного струму тим, що в ній електроди відділені від газового проміжку прошарком діелектрика, на якому при проходженні струму через проміжок часу утворяться електричні заряди, що гасять розряд і полегшують його запалювання при зміні полярності напруги. Електричне поле вказаних зарядів на стінках комірки зумовлює її пам’ять.

Контрольні питання

1. В чому суттєва відмінність растрового та векторного принципів відображення інформації?
2. Які індикатори застосовують при векторному принципі відображення інформації?
3. Наведіть концептуальні блок-схеми векторних дисплеїв з регенерацією.

6 ПРИНЦИПИ ПОБУДОВИ ПРОГРАМНИХ ЗАСОБІВ МАШИННОЇ ГРАФІКИ

6.1 Класифікація проблемно-орієнтованих графічних мов

Проблемно-орієнтовані графічні мови, що використаються в системах машинної графіки, прийнято класифікувати за такими ознаками:

- оперативність;
- наявність засобів для опису операцій обробки;
- зв’язок з універсальними алгоритмічними мовами програмування;
- метод задання команд (операторів) мови;
- місце в процесі обробки графічних даних.

За оперативністю мови поділяють на діалогові (оперативні) і пасивні. Діалогові забезпечують роботу в реальному масштабі часу шляхом обробки операторів мови в режимі інтерпретації, що дозволяє оперативно отримувати результат виконання програми в графічній формі. В діалогових мовах для задання операторів поряд з алфавітно-цифровими даними використовуються і графічні побудови, що виконуються на графічному терміналі за допомогою пристроїв введення. Пасивні мови дозволяють задавати сукупність графічних операцій у вигляді деякого символічного опису з наступною компіляцією цих описів і виконанням в режимі пакетної обробки.

За наявністю засобів для опису операцій обробки виділяють інформаційні та алгоритмічні мови. Інформаційні дозволяють описувати тільки графічні дані, алгоритмічні призначені для опису графічних даних і операцій над ними, охоплюючи обчислювальні операції, операції управління введенням/виведенням і зберігання даних.

За зв’язком з універсальними алгоритмічними мовами розрізняють автономні та поширювальні мови. Перша має власну граматику, відповідний транслятор і може застосовуватися незалежно від інших мов програмування. Поширювальні мови будуються на основі граматики іншої мови і є його графічним доповненням. Базою розширення найчастіше служать універсальні алгоритмічні мови. Такий підхід дозволяє використати всі наявні в базовій мові потужні засоби обробки даних і спростити зв’язок машинної графіки з проєктувальними компонентами системи, а також забезпечити, значною мірою, незалежність мови від типу ЕОМ, яку використовують.

За способом задання операторів мови поділяються на символічні (алфавітно-цифрові), цифрові, і графосимволічні мови. Програма символічною мовою являє собою послідовність текстових рядків фіксованого або

довільного формату. Цифрові мови – це сукупність кодових комбінацій, в яких числами задаються як коди графічних команд, так і їх параметри. Графосимволічні мови, як правило, діалогові і дозволяють задавати графічну інформацію у формі комбінації текстових директив і графічних побудов.

За місцем в процесі обробки графічних даних розрізняють вхідні, внутрішні і вихідні графічні мови. Вхідні графічні мови (ВГ-МОВИ) призначені для опису і введення в ЕОМ графічних даних і задання дій над ними. Звичайно, ВГ-МОВИ містять деякий базисний набір графічних операторів і передбачають можливість розширення їх залежно від специфіки сфери та умов застосування. Загальними для багатьох ВГ-МОВ є підмножини команд, що забезпечують: побудову графічних примітивів; задання атрибутів графічних примітивів; побудову графічних зображень довільної конфігурації; побудову зображення з обмеженої множини елементів, які мають типову конфігурацію; перетворення зображення (афінні та інші перетворення); документування інформації в графічному, текстовому вигляді або запис на машинні носії; прийом і передачу інформації; управління приладами введення.

Підмножини цих команд можуть бути поширені або скорочені залежно від сфери та умов використання конкретної мови.

Внутрішні мови призначені для програмної обробки даних, накопичення та зберігання їх в системі, забезпечення протоколів зв’язку між різноманітними компонентами системи. В сучасних системах внутрішні мови використовуються також для запису даних в так звані графічні метафайли, в яких зображення зберігаються в форматах, незалежних від команд конкретних графічних приладів.

Вихідні мови призначені для виведення даних з ЕОМ з метою графічного відображення та документування. Формати і набір операторів вихідних мов суттєво залежать від використовуваних приладів графічного виведення.

Контрольні питання

1. Назвіть сфери застосування оперативних і пасивних графічних мов.
2. Як графічні мови підрозділяються за місцем в процесі обробки графічних даних?
3 Які команди властиві вхідним, внутрішнім та вихідним графічним мовам?

6.2 Основні підходи до розробки програмних засобів машинної графіки

Графічні мови реалізують шляхом розширення універсальних мов програмування або шляхом розробки автономних мов.

В структурі програмного забезпечення проблемно-орієнтованих графічних систем на основі розширення існуючих алгоритмічних мов загального призначення можна виділити принаймні п’ять рівнів.

Нульовий рівень складають системні програми управління введенням/виведенням (драйвери) графічних приладів, що створюються, як правило, з використанням мов програмування низького рівня (автокоди, асемблери).

Програмне забезпечення *першого рівня* являє собою графічні автокоди приладів, з використанням яких організовується формування файлів виведення на графічні прилади.

Другий рівень складає базове програмне забезпечення машинної графіки (БПЗ МГ) – ядро розширення алгоритмічних мов. БПЗ МГ створюється як незалежне від властивостей конкретних приладів і проблемного застосування, реалізує найбільш розповсюджені функції введення, виведення та зберігання графічної інформації. Воно є предметом уніфікації як в рамках однієї мови, так і в цілому в системах машинної графіки.

Третій рівень складає програмне забезпечення, що містить найбільш представницькі функції конкретної сфери (галузі) застосування, наприклад, програми побудови графіків для систем автоматизації експерименту, програми формування елементів креслеників для систем автоматизованого проєктування і т. п. Ці програми використовують програми другого рівня і є доповненням його для конкретних застосувань.

Останній, *четвертий рівень* програмного забезпечення складають програми користувачів системи і призначені для орієнтованих систем.

Програми другого і третього рівнів, як правило, об’єднуються в пакети прикладних програм машинної графіки (ППП МГ) і складають основу розширення алгоритмічних мов.

Виходячи зі ступеня залежності пакетів прикладних програм МГ від конкретних типів графічних приладів і трудоємкості їх налаштування на роботу з приладами інших типів, розрізняють чотири рівня інваріантності ППП МГ до графічних приладів.

Рівень 0 – пакет орієнтований на застосування конкретного набору технічних засобів і практично не має засобів налаштування на інші прилади. Перевагами ППП цього рівня є можливість досягнення високої ефективності використання можливостей приладів і мінімальні витрати процесорного часу на перетворення зображення з систем координат користувача в приладну систему координат. Основний недолік – необхідність перепрограмування всіх базисних функцій при зміні приладів.

Рівень 1 – пакети прикладних програм побудовані за принципом «перевернутої піраміди». Вихідні дані в таких пакетах є загальним інтерфейсом для всіх графічних приладів, оскільки набір вихідних графічних примітивів дуже обмежений, але достатній для генерації будь-якого зображення. Цей набір може містити, наприклад, такі примітиви: точка, відрізок прямої, алфавітно-цифровий символ.

Істотній недолік цього підходу полягає в тому, що він суттєво обмежує можливість використання (за умови збереження інваріантності) функцій, що реалізувалися апаратно (наприклад, побудова дуг і кіл, ліній різних типів, текстів і т. п.). Це збільшує обсяг програм пакета, що моделюють ці функції, знижується ефективність використання ЕОМ, зростає час формування зображення.

Перевага такої організації ППП МГ полягає в інваріантності прикладних програм користувача до графічних приладів і в мінімальних трудоємності налаштування самого пакета на нові пристрої шляхом написання дуже обмеженої кількості програм формування вихідних примітивів пакета в кодах пристрою, що підключається.

Рівень 2 – рівень віртуального графічного пристрою, тобто пристрою, який дає користувачу деякий уніфікований набір вхідних і вихідних функцій незалежно від того, є фізичний еквівалент такого пристрою в системі чи немає. ППП МГ налаштовується на такий віртуальний пристрій (графічний протокол). Підключення конкретного графічного пристрою до системи з таким пакетом у випадку різниці між ним і віртуальним пристроєм потребує розробки відповідних програм, які доповнюють даний фізичний пристрій до віртуального графічного пристрою.

Такий підхід ефективний для багатотермінальних систем, коли одна і та ж програма має формувати одночасно зображення для різних графічних пристроїв. При цьому збільшується час ЕОМ, що затрачається на виконування перекодувань і перетворення координат, але досягається інваріантність до графічних пристроїв при одночасному максимальному використанні їх апаратних можливостей.

Рівень 3 – рівень віртуального графічного пристрою з напівавтоматизованим налаштуванням на фізичний пристрій. ППП МГ цього рівня відрізняється від рівня 2 тим, що в своєму складі має програму загального інформаційного погодження, яка на основі таблиці можливостей пристрою виконує програмне моделювання тих функцій віртуального пристрою, які не реалізовані апаратно у фізичному пристрої. Такий підхід дозволяє зменшити трудомісткість створення програм, що доповняють фізичний пристрій до віртуального графічного пристрою системи.

Контрольні питання

1. На якому рівні розширення алгоритмічних мов розробляють драйверні програми?
2. Який рівень розширення алгоритмічних мов складає ядро розширення?
3. В чому полягають недоліки принципу «перевернутої піраміди»?
4. Вкажіть дії, характерні різним рівням інваріантності ППП МГ до графічних приладів.

6.3 Стандарти машинної графіки

Основною метою стандартизації у сфері машинної графіки є розробка методичних і базових програмних засобів для створення транспортабельних прикладних програм, що використають графічні засоби, тобто програм, які відносно легко трансформувати з однієї конфігурації технічних засобів на іншу. Це пов'язано з тим, що графічні термінали відрізняються як системою графічних команд, так і кількістю приладів введення/виведення.

Одні термінали обладнані майже повним набором приладів, в той час як інші мають лише один або два (наприклад, світлове перо і клавіатура або тільки одне світлове перо). Стандартизація програмного забезпечення машинної графіки засновується на поданні всіх приладів графічних систем і їхніх характеристик в деякому уніфікованому вигляді з метою отримання достатньо гнучкого апарату еквівалентної заміни одного фізичного приладу іншим без зміни суті програми.

Інша мета стандартизації – це досягнення структурної і технологічної єдності розробки прикладних графічних програм, що полегшує роботу програмістів з різноманітними реалізаціями стандартного графічного пакета в різних мовах програмування і на різних обчислювальних системах.

Стандартизація програмного забезпечення машинної графіки ставить метою забезпечити перший або другий рівень модифікації графічних програм при їх перенесенні з однієї системи на іншу.

З появою мереж ЕОМ і розподілених графічних систем, в яких відбувається розподіл функцій між локальним графічним процесором і обслуговувальною ЕОМ, виникла проблема стандартизації графічних протоколів, які регламентують зв'язок обчислювальних машин при розподіленій обробці графічних даних.

Класифікація

У основі розробки графічних стандартів лежить принцип віртуальних ресурсів, що дозволяє розділити графічну систему на декілька шарів – прикладний, базисний і апаратно-незалежний. При цьому кожний шар є віртуальним ресурсом для верхніх шарів і може використовувати можливості нижніх шарів за допомогою стандартизованих програмних інтерфейсів. Крім того, графічні системи можуть обмінюватися інформацією з іншими системами або підсистемами за допомогою стандартизованих файлів або протоколів.

Згідно з цим виділяють три основних напрямки стандартизації – базисні графічні системи, інтерфейси віртуального пристрою, формати обміну графічними даними.

Стандартизація базисних графічних систем спрямована на забезпечення мобільності прикладних програм і заснована на концепції ядра, що містить універсальний набір графічних функцій, загальних для більшості застосувань.

Найбільш відомими проєктами зі стандартизації базисних систем є Core System, GKS, GKS-3D, PHIGS, PHIGS+.

Основний напрямок розвитку цих проєктів полягає в посиленні образотворчих можливостей для візуалізації геометричних об’єктів (2D, 3D, видалення прихованих ліній і граней, півтонового зафарбування, текстурування й ін.). Стандарт на базисну графічну систему містить у собі функціональний опис і специфікації графічних функцій для різних мов програмування.

Концепція віртуального пристрою почалась розроблятися з моменту появи апаратно-незалежних графічних систем. Інтерфейс віртуального пристрою розділяє апаратно-залежну й апаратно-незалежну частини графічної системи. Він забезпечує можливість заміни графічних пристроїв (термінальну незалежність), а також можливість роботи з декількома пристроями одночасно. Інтерфейс віртуального пристрою може існувати у формі програмного інтерфейсу і/або протоколу взаємодії двох частин графічної системи. Найбільш чітко концепція віртуального пристрою подана в проєкті CGI.

Розвиток цієї концепції збігся з активним застосуванням графічних засобів на персональних комп’ютерах і графічних станціях. При цьому основними інтерактивними пристроями стали растрові дисплеї, а пристроями для одержання твердих копій – растрові принтери. Це привело до необхідності виділення окремого набору растрових функцій, що дозволяють використовувати функціональні можливості растрових пристроїв.

Подальший розвиток растрових функцій пов’язаний з появою багатовіконних графічних систем X Window і MS Windows (а також NeWS і Display Postscript), що забезпечили зручні засоби для маніпулювання растровими зображеннями. Ці засоби є основою для розвитку систем опрацювання зображень і для організації ефективного багатовіконного інтерфейсу користувача з використанням меню, діалогових панелей, смуг перегляду й ін. Відзначимо, що традиційні засоби виведення геометричних примітивів (ліній, дуг, багатокутників) і текстів також є в цих системах.

Сьогодні найбільш розвинуті проєкти PEX і OpenGL непогано поєднують основні досягнення як геометричного, так і растрового напрямків.

Графічні системи класу 2D

GKS. Базисна графічна система GKS забезпечує функціональний інтерфейс між прикладною програмою і деяким набором вхідних і вихідних графічних пристроїв.

Графічна коренева система (ГКС) (GKS – Cгaphical KernelSystem) розроблена фахівцями ФРГ і прийнята міжнародною організацією зі стандартизації (ISO) як міжнародний графічний стандарт (Draft International Standart – ISO/DIS).

Однієї з найважливіших концепцій, покладених в основу графічних стандартів, є ідея розподілу функцій графічного програмного забезпечення на модельні і власне графічні. Відповідні частини програмного забезпечення називаються модельною і графічною системами.

В *модельній системі* графічні об'єкти визначаються у власних локальних координатах, і система має засоби перетворення для роботи з цими координатами. Після застосування до об'єкта деякої сукупності таких перетворень формується опис об'єкта у світовій системі координат, що є машинно-незалежною декартовою системою координат.

Графічна система, використовуючи світові координати точок об'єкта як вхідну інформацію, генерує зображення об'єкта і виводить його на графічний пристрій. Зображення будується шляхом звернення до програм рисування примітивів – відрізків прямих ліній, маркерів і рядків тексту. Як проміжна між світовою і екранною системами координат вводиться нормалізована приладно-незалежна система координат. Вихідні примітиви піддаються двом перетворенням на шляху між прикладною програмою і виведенням на графічний пристрій: перетворення типу вікно/робоча область, при якому простір світових координат відображається на просторі нормалізованих координат; перетворення на пристрої, що використовується для відображення простору нормалізованих координат на простір екранних координат індивідуально для кожного графічного пристрою.

Під час першого перетворення графічний об'єкт можна повернути, промасштабувати, перемістити і т. д. перед тим, як передати його графічній системі для візуалізації.

Вихідні примітиви графічних систем подані набором операцій типу: позиціювання, відрізків прямих ліній, послідовності відрізків (ламана), маркери і їх послідовність, рядки тексту, пікселів і заповнення площі для використання в растровій графіці. CORE SYSTEM припускає дво- і тривимірні примітиви.В ГКС введений узагальнений примітив креслення у вигляді послідовності координат точок і засобів їхнього сполучення (дуга, коло, еліпс, сплайн-апроксимація і т. п.).

Вигляд вихідного примітива залежить від таких значень атрибутів примітива, як: тип і товщина лінії, яскравість, колір, тип шрифту, розмір символів, напрямок рядка символів, площина обрису символів, розміщення символів в рядку, відстань між символами, центрування рядка символів, якість обрису символів, вид маркера. Для ідентифікації кожний вихідний примітив може мати приписане йому ім'я.

Сегментація зображення. Всі вихідні примітиви, що утворять графічний об'єкт, можуть бути записані в виді сегмента, що задається прикладною програмою. Групуючи примітиви об'єкта в пойменовані сегменти, програміст може селективно змінювати окремі частини повного зображення шляхом вилучення і повторного формування сегмента (заміна сегмента). Характер зображення, що складається з сегментів, залежить від

значень динамічних атрибутів сегмента. Атрибути видимості використовуються для управління виведенням зображення, яке визначається сегментом. На відміну від атрибутів примітива значення динамічних атрибутів сегмента можуть змінюватися програмою після закінчення формування сегмента. Роздільна ідентифікація сегментів і примітивів дає можливість при одному рівні сегментації отримати додатковий рівень «назви», наприклад, набір світлових кнопок як єдиний сегмент має ім'я і кожна кнопка має ідентифікатор примітиву.

Ім'я означеної кнопки, що вертається в прикладну програму, буде складатися з імені набору і ідентифікатора примітиву.

Вхідні примітиви призначені для опису даних, що вводяться з віртуальних приладів введення типу локатора положення (LOCATOR), введення цифрових даних (VALUATOR), вказання на частину зображення (PICK), вибору (COINCH), введення рядків символів (STRING). Введення даних з цих приладів може здійснюватися в трьох режимах: запит, переривання і ПДП.

Всі віртуальні прилади в конкретних системах реалізуються тими або іншими реально існуючими вхідними приладами. Кожний клас приладів має свій системно-заданий зворотний зв'язок («ехо– відображення»), що може бути ввімкнутий чи вимкнутий прикладними програмами.

Однією з центральних в ГКС є концепція робочого місця. Виведення графічної інформації може бути здійснено на одне або декілька робочих місць. Робоче місце ГКС являє собою термінал, що містить тільки один пристрій відображення і також один або декілька приладів введення.

Для кожного робочого місця визначаються специфічне перо і текстова таблиця, що дозволяє керувати характером всіх примітивів (атрибутами примітивів) на відповідному робочому місці. Для довгострокового зберігання графічних даних введена концепція метафайлу. Метафайл також використовується для організації стандартного інтерфейсу між ГКС і іншими графічними системами.

Система ГКС надає тільки незалежну від мови програмування комірку графічної системи. Для можливості використання ГКС в складі мови програмування має бути передбачений мовно- орієнтований шар, що забезпечує виконання всіх команд конкретної мови програмування, наприклад надання значень параметрам і іменам.

PostScript (Adobe Systems) – мова опису сторінок для растрових пристроїв друкування. Відмінна риса – широкі образотворчі можливості при мінімальному наборі графічних функцій. Багато графічних систем і настільних видавничих систем підтримують PostScript. Деякі виробники лазерних принтерів забезпечують його апаратну підтримку. PostScript використовують для виконання графічних функцій у багатовіконних системах NeWS і Display PostScript. Привабливі властивості цієї мови сприяли появі її тривимірних розширень.

Широкі образотворчі можливості мови PostScript забезпечені поняттям траєкторії, що може бути складена з ліній, дуг, сегментів кривої Безьє і текстових символів. У процесі виведення траєкторії можуть піддаватися довільним лінійним перетворенням. Замкнуті траєкторії можуть бути зафарбовані, заповнені растровим зразком або заштриховані іншими траєкторіями. Заповнення може проводитися за різними правилами (even-oc, nonzero-winding-number). Лінії можуть бути різного типу, змінної товщини і мати округлення в точках з'єднання. Робота з текстами відбувається на основі багатої бібліотеки шрифтів. Підтримується декілька колірних моделей – RGB, CMY і HSV.

CGI – проєкт стандарту (ISO, 1986) на інтерфейс віртуального пристрою. CGI орієнтований не на прикладних, а на системних програмістів, що займаються розробкою графічних систем. Функціональні можливості CGI сформовані з урахуванням розроблених раніше проєктів GKS і CGM (Computer Graphics Metafile).

Функції виводу підтримують роботу з лініями, багатокутниками, прямокутниками, маркерами, текстами, дугами, секторами та сегментами кола й еліпса, а також замкнутими фігурами, складеними з цих примітивів. Замкнуті об'єкти можуть зафарбовуватися, заштриховуватися або заповнюватися растровим зразком. Набір атрибутів CGI аналогічний наборові атрибутів GKS. Конвеєр перетворення обмежений перетворенням робочої станції. Функції сегментації аналогічні наявним у GKS.

Растрові функції підтримують роботу з відображуваними і віртуальними бітовими картами. Перші є частиною відеопа'мяті пристрою. Другі можуть бути повноколірними матрицями пікселів у пам'яті. Двоколірні віртуальні бітові карти можуть служити як маски для операції заповнення областей, а також для задання символів, маркерів, курсорів і ін. Атрибутами карт є прозорість, основний і фоновий колір. Введено різні режими накладення кольорів при виведенні пікселів (and, or, xor, …).

Функції введення аналогічні наявним у GKS із деякими доповненнями. Введено поняття тригера, що дозволяє встановити режим спрацьовування окремих пристроїв залежно від деякої події. Більш чітко визначені поняття підказування, відлуння і підтвердження. Введено два нових логічних пристрої введення – растрова область і узагальнений пристрій введення.

X Window System – багатовіконна графічна система, розроблена в Массачусетському Технологічному інституті. Одна з основних цілей розробки – забезпечення мережної прозорості і можливості використання широкого спектра кольорових і монохромних графічних станцій.

Система розділена на дві частини, клієнт і сервер, що взаємодіють за допомогою X-протоколу. Прикладному програмісту надана бібліотека базисних функцій і бібліотека інструментальних засобів. Функції керування забезпечують можливість маніпулювання системою вікон і контролю за діями користувача. Параметри графічних функцій містять у собі ідентифікатори

дисплея і вікна, а також графічний контекст, що містить значення атрибутів і інші параметри відображення.

Функції виведення забезпечують зображення точок, ліній, дуг, кіл, прямокутників, а також заповнення багатокутників, секторів, сегментів і прямокутників. Аналогічно, як в мові PostScript, є атрибути, що визначають спосіб округлення ліній і правила заповнення. Функції виведення текстів підтримуються багатою бібліотекою шрифтів. Конвеєр перетворення координат відсутній. Структуризація або сегментація даних не підтримується.

Растрові функції забезпечують широкі можливості для маніпулювання з пікселними матрицями. Пікселні матриці можуть використовуватися як зразки заповнення, а бітові – як маски відсікання. Застосована колірна модель – RGB.

Функції введення на базисному рівні забезпечують механізм опрацювання подій від миші і клавіатури. Функції більш високого рівня забезпечують роботу з меню, діалоговими панелями, смугами перегляду й ін.

Microsoft Windows – багатовіконна надбудова над операційною системою MS DOS на IBM PC. Версія Windows NT трансформувалася в повноцінну операційну систему. Забезпечує багатозадачний режим. Графічні функції системи аналогічні наявним у X Window, однак у параметрах функцій немає ідентифікатора дисплея. Підтримується метафайл.

Графічні системи класу 3D

GKS-3D – розширений варіант GKS (ISO, 1987), що дозволяє працювати з тривимірними графічними об’єктами. У цей проєкт внесені додаткові (стосовно GKS) можливості

Функції виведення доповнені сімома 3D-примітивами – ті ж, що в GKS із приставкою 3D і набір функцій для заповнення 3D-областей. Для останніх функцій введені атрибути контуру, аналогічні атрибутам ліній. Введено атрибут для керування алгоритмами видалення прихованих ліній і граней. Введено 3D-перетворення, 3D-нормалізація, видове перетворення, 3D-перетворення робочої станції. Видове перетворення дозволяє робити паралельне та центральне проекціювання .

Функції сегментації розширені можливістю роботи з 3D-сегментами. Введено перетворення 3D-сегментів.

Функції введення доповнені двома логічними пристроями для введення 3D-координат і 3D-ліній.

PHIGS – альтернативний стосовно GKS-3D стандарт (ANSI-1986, ISO-1989), що забезпечує можливість інтерактивних маніпуляцій з ієрархічно структурованими графічними об’єктами. Одержав подальший розвиток у проєктах PHIGS+ і PEX. Порівнянні з GKS-3D характеристики наведено нижче.

Набір примітивів і атрибутів аналогічний наявним у GKS-3D. Підтримується декілька колірних моделей – RGB, CIE, HSV, HLS. Замість 3D-перетворення нормалізації введене модельне перетворення.

Замість сегментів введені *ієрархічні структури даних*. Структури можуть містити в собі примітиви, атрибути, перетворення, неграфічні дані. Засоби редагування дозволяють видаляти і копіювати елементи структур. Внесено механізм фільтрації, що здійснює вибіркове відображення елементів, їхнє виділення й ін.

PHIGS+ (або **PHIGS-PLUS**) – проєкт розширення PHIGS (ISO/ANSI Draft 1990), спрямований на забезпечення основних вимог прикладних програм у сфері освітлення, півтонового зафарбовування й ефективного опису складних поверхонь. Для цих цілей у PHIGS+ внесено:

- набір поліліній;
- крива B-сплайна;
- полігональна область;
- набір полігональних областей з даними;
- набір трикутників;
- оверхня B-сплайна.

Примітиви, що мають суфікс «із даними» дозволяють внести додаткову інформацію, що є частиною визначення примітива. Наприклад, у випадку набору трикутників для кожної грані і/або вершини можна задати комбінації кольору, нормаль і прикладні дані. Існує механізм керування, що дозволяє визначити, які дані варто використовувати, а які – пропустити під час відображення. PHIGS+ розрізняє передню і задню поверхні грані на основі геометричної нормалі. Різні значення кольору й інших атрибутів можуть бути визначені для передньої та задньої граней. Для обчислення освітленості крім геометричних характеристик задаються відбивні властивості поверхні, а також розташування джерел кольору і їхньої характеристики.

PEX (MIT X Consortium) – проєкт розширення системи X Window для підтримки PHIGS+. Це одна з двох систем (інша – OpenGL), що забезпечують найбільш розвинуті на сьогоднішній день інструментальні засоби для побудови реалістичних зображень. Суть проєкту PEX полягає в описі механізму розширення X-протокола і X-сервера для забезпечення функцій PHIGS+, що, у першу чергу, призначено для системних програмістів. З погляду прикладного програміста функціональні можливості PEX у частині зображення просторових об’єктів відповідають системі PHIGS+. Однак, починаючи з версії 5.2, у PEX з’явилися нові можливості, що забезпечують усунення ступінчастості (antialiasing) і текстурування поверхонь. Засоби роботи з растровими зображеннями підтримуються за допомогою X Window і додаткових розширень.

OpenGL – стандарт, запропонований компанією Silicon Graphics у 1993 році, що регламентує інтерфейс прикладного програміста. Попередником цього проєкту є IRIS GL (SGI 1988 р.). Орієнтований на роботу в системі X Window. Про підтримку OpenGL повідомляли майже всі головні фірми-виробники, зокрема ОС Windows NT має цей стандарт у своєму комплекті. За функціональними можливостями OpenGL приблизно відповідає системі PEX останніх версій, але дещо відрізняється за стилем програмування. Крім того, на відміну від PEX, має власні розвинуті засоби для роботи з растровими зображеннями.

Контрольні питання

1. Яка мета стандартизації в машинній графіці?
2. Наведіть класифікацію стандартів машинної графіки.
3. Перечисліть основні концепції стандарту GKS.
4. Перечисліть основні положення найбільш поширених стандартів машинної графіки.

7 РЕАЛІЗАЦІЯ ІНТЕРАКТИВНОГО РЕЖИМУ

7.1 Діалог і його основні характеристики

Діалогом людини з ЕОМ вважають такий процес обміну повідомленнями між ними, при якому виконуються такі умови:

- існує мета, для досягнення якої ініціюється діалог. При цьому розрізняють основну мету, що її ставить завжди людина, і проміжні, що можуть бути поставлені також машиною. Наприклад, ЕОМ може поставити перед користувачем задачу: опанувати знання, необхідні для досягнення основної мети;

- ролі інформатора і споживача інформації (реципієнта) по черзі виконуються людиною й ЕОМ;

- знання й уміння, необхідні для досягнення основної мети, є у партнерів принаймні «у сумі»;

- між партнерами досягнуте взаєморозуміння, що проявляється в знанні кожним із них мови, за допомогою якої відбувається обмін інформацією;

- основним або побічним результатом усякого діалогу є поповнення, хоча б одним із партнерів, своїх знань, умінь, навиків.

До основних характеристик, що визначають процес діалогової взаємодії, відносять:

оперативність;

спроможність до керування;

- готовність партнерів до самостійного виконання дій з досягнення основної мети діалогу;

- спроможність партнерів до навчання.

Оперативність визначається часом відповіді (реакцією) партнерів на поставлене питання. Розрізняють такі значення цієї характеристики: «двостороння оперативність», «оперативність із боку машини». Більш критичним є значення оперативності з боку ЕОМ, тому що людині властиво очікувати миттєвої реакції системи на її дії або її відповідь із деякою затримкою на досить складні (із погляду людини) питання. Ієрархія значень часу реакції ЕОМ на дії користувача, запропонована Міллером, містить у собі три основних рівні відчуття нормального завершення роботи:

- лексичний, з часом реакції системи до 50 мс. Це, наприклад, час появи ехо-відображення на екрані дисплею для символу, що вводиться з клавіатури термінала, або початку мерехтіння виділеного графічного елемента. Затримка або відсутність відповіді на дії даного класу дратує користувачів і знижує ефективність роботи;

- синтаксичний, з часом відповіді системи від 0.5 до 4 с, наприклад, відповідь на запит зробити заміну деякого графічного елемента, отримання довідкової інформації або синтаксичний аналіз введеної команди;

- семантичний – це питання, на яке користувач очікує серйозних, розумних відповідей. Час реакції більше 10 с. Затримка відповіді в цьому випадку використовується людиною для обдумування можливих стратегій поведінки залежно від відповіді. Для запобігання панічному сприйманню паузи, машина приблизно кожні 10 с має видавати користувачу підтвердження нормальної роботи.

Спроможність до керування виражається у спроможності до видачі таких команд і (або) питань партнеру, що забезпечували б виконання останнім деяких дій, спрямованих на досягнення цілей діалогу .

Готовність партнерів до самостійного виконання дій для досягнення цілей діалогу залежить від досвіду користувача, а також від комплексу програмних засобів, які забезпечують діалогову взаємодію. Важливим є надання допомоги зі сторони обчислювальної машини.

Здатність партнерів до навчання оцінюється двостороннє: здатність до навчання ЕОМ, здатність до навчання людиною. Ця характеристика діалогу проявляється в тому, що в одного з партнерів (або в обох) в процесі взаємодії підвищується рівень готовності до самостійного досягнення мети діалогу або підвищується здатність до керування.

Контрольні питання

1. Назвіть основні умови реалізації діалогу.
2. Назвіть основні характеристики, які визначають процес діалогової взаємодії.
3. Які дії можна виконати на лексичному, синтаксичному та семантичному рівнях?

7.2 Психофізіологічні фактори взаємодії оператора з ЕОМ

При розробці діалогових графічних систем (ДГС) потрібно враховувати психологічні та фізіологічні можливості людини. В загальному випадку це зводиться до виконання вимог інженерної психології до графічного пристрою як пристрою відображення інформації, що сприймається людиною, і до ДГ-термінала як робочого місця людини-оператора. Програмне забезпечення ДГС і ДГ-МОВ не має викликати нудьгу, паніку та розчарування.

Нудьга і паніка виникає у користувача при порушенні оперативності діалогу, що виникає у невідповідності відклику системи необхідному темпу діалогу або виконуваним користувачем діям.

Розчарування користувача пов’язане з обмеженістю (на його думку) можливостей ДГС або ж з труднощами роботи з нею. Як правило, ці

психологічні бар'єри виникають у ДГС з суттєво обмеженими можливостями щодо вибору методів введення інформації, діагностики помилок користувача або вибору підказок і інших засобів.

Серед вимог до мови графічного діалогу, що враховують психофізіологічні аспекти діяльності людини за дисплейним пультом, особливо виділяють забезпечення візуальної, тактильної і контекстуальної неперервності.

Візуальна неперервність означає, що при роботі в певному інтервалі часу увага користувача має бути постійно зосереджена на потрібній ділянці екрана або на певній групі клавіш, що з цією метою групуються за смисловим призначенням і можуть бути однакової форми або кольору.

Тактильна неперервність передбачає таку організацію послідовності введення команд, при якій унеможливлюються часткові чергування різноманітних дій хоча б при введенні одного речення. Наприклад, послідовність дій <миша – клавіатура – миша> не задовольняє вимоги тактильної неперервності. Більш прийнятна послідовність <миша – миша – клавіатура>.

Контекстуальна неперервність вимагає зосередження уваги на частині загальної задачі, що вирішується в процесі діалогу. Це досягається виданням машиною таких відповідей, що сприймаються користувачем негайно і підтверджують результат даного кроку.

Для врахування психологічних чинників і поліпшення процесу взаємодії людини з ЕОМ в ДГС використовують низку прийомів в організації мови зображень і сервісних програмних засобів.

Поле екрана графічного дисплея звичайно поділяють на окремі ділянки за функціональним призначенням: головну (робочу), в якій відтворюється власне графічне подання об'єкта проєктування; ділянку процесів, призначену для відображення ключових слів команд користувача, допустимих в даному стані системи; графічних даних для відображення стандартних або побудованих раніше графічних об'єктів, що використовуються як елементарні для побудови складних зображень; супроводу діалогу, в якому виводяться системні вказівки користувачу, питання системи і діагностичні повідомлення.

Контрольні питання

1. Чим пояснюється необхідність урахування психофізіологічних факторів взаємодії оператора з ЕОМ при розробці діалогових графічних мов?
2. Охарактеризуйте візуальну, тактильну та контекстуальну неперервності.
3. Які прийоми використовують для врахування психологічних чинників і поліпшення процесу взаємодії людини з ЕОМ в ДГС?

7.3 Класифікація діалогових графічних систем

Діалогові графічні системи (ДГС) класифікують за структурою технічних і програмних засобів і за орієнтацією на певний клас користувачів або задач.

За структурою технічних і програмних засобів ДГС поділяють на однопультові, багатопультові і розподілені.

Однопультові ДГС орієнтовані на монопольне використання ЕОМ одним користувачем, який працює за діалоговим графічним терміналом. Однопультові ДГС неефективно використають ресурси ЕОМ, тому застосовуються або на ЕОМ з невеликими ресурсами або в системах з багатопрограмним режимом роботи, в яких час процесора ділиться між програмою обслуговування термінала і програмами, що виконуються на ЕОМ в пакетному режимі.

Багатопультові ДГС містять декілька ДГ-терминалів і використовують ЕОМ в режимі розподілу часу.

Операційна система ЕОМ періодично виділяє кожному користувачу невеликий інтервал (квант) часу так, що за рахунок високої продуктивності процесора у користувачів створюється враження монопольної роботи. Кожному користувачу виділяється також деякий обсяг оперативної і зовнішньої пам’яті ЕОМ, де містяться його програми і масиви даних. В той же час всі користувачі мають доступ до загальних банків даних і загальних програм.

Розподілені ДГС базуються на розподілі обчислювальних функцій між центральною ЕОМ великої продуктивності і локальної (термінальної) ЕОМ, що управляє роботою ДГ-ТЕРМІНАЛА. Розподілені ДГС забезпечують:

- оперативну підготовку, редагування та корегування даних в автономному режимі роботи ДГ-ТЕРМІНАЛА до введення їх в центральну ЕОМ;
- економію ресурсів як центральної ЕОМ, так і лінії зв’язку завдяки зменшенню кількості запитів на передачу та обсяг даних, що передаються;
- скорочення часу відповіді на запити порівняно з суттєво завантаженими операційними системами, що працюють в режимах мультипрограмування або розподілу часу.

За орієнтацією на певний клас користувачів ДГС поділяють на три групи:

ДГС редагування зображення використовують тільки для створення зображення, його корекції і компонування. Графічний дисплей з приладами графічного введення фактично використовується як «електронна» креслярська дошка. Неграфічні дії над зображенням в таких системах суттєво обмежені або відсутні взагалі.

Користувачами таких систем можуть бути будь-які фахівці, які намагаються автоматизувати процес створення й введення в ЕОМ ескізів і креслеників.

ДГС спеціалізованого використання застосовуються для розв'язання обмеженого класу задач з поданням результатів в графічній формі. Питома вага програм машинної графіки в таких системах невелика порівняно зі спеціалізованим програмним забезпеченням вирішення досліджуваної проблеми. З цієєї причини спеціалізовані ДГС називають «системами з використанням інтерактивної графіки», а не просто «графічними системами».

ДГС загального призначення, або інструментальні, призначені для використання при розробці проблемно-орієнтованих графічних систем.

В цих ДГС реалізовані всі основні засоби і загальні програми графічного введення/виведення, існують мови високого рівня для розробки прикладних діалогових графічних програм. Вони дозволяють звести до мінімуму витрати на розробку структури даних, на організацію графічного введення/виведення та скоротити терміни розробки проблемно-орієнтованих систем.

Контрольні питання

1. Які діалогові графічні системи Вам відомі?
2. Охарактеризуйте рівні розподілу обчислювального процесу в розподільних ДГС.
3. Як розділяються ДГС за орієнтацією на певний клас користувачів?

7.4 Програмна імітація пристроїв введення

Для ефективної реалізації діалогових взаємодій використовують різні пристрої введення графічної інформації.

Передекранна сенсорна панель містить пари оптоелектроних елементів, які розміщені по периметру екрана. Кожна з таких пар містить джерело світла та його приймач (рис. 7.1).

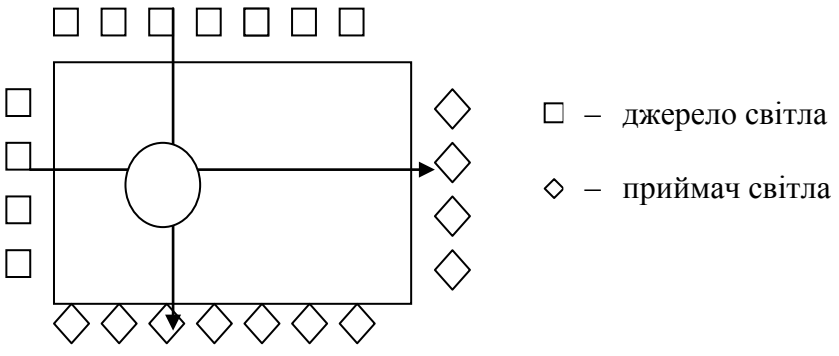


Рисунок 7.1 – Передекранна оптоелектронна сенсорна панель

При дотику оператором панелі виконується введення в ЕОМ координат X та У положення об’єкта на екрані.

Режим позиціювання полягає у встановленні графічного маркера в задану точку екранної системи координат. Враховуючи, що для цього використовується обмежена кількість оптоелектронних пар, процедура позиціювання охоплює два етапи: «грубого» позиціювання та «точного» позиціювання.

Під час першого етапу оператор встановлює реєструвальний орган в потрібну точку екрана. При цьому графічний маркер переміщається в початок макрозони, що її утворюють оптоелектронні пари. Під час другого етапу (переключення можливе різними шляхами, наприклад, з використанням кнопки або автоматично – відразу після переміщення реєструвального органу з активної зони) оператор переміщає реєструвальний орган в напрямку отрібної позиції графічного маркера. При цьому кожне переміщення макрозони зумовлює переміщення маркера на один піксел в тому ж напрямку.

Позначимо через n кількість оптоелектронних пар, яку потрібно забезпечити для однієї зі сторін екрана, яка містить N точок. Тоді відношення N/n визначає розмір макрозони для режиму позиціювання. Для забезпечення ідентифікації кожної точки макрозони відношення N/n має дорівнювати n, тобто,

$$N/n = n.$$

$$N = n^2.$$

Звідси

$$n = \left\lceil \sqrt{N} \right\rceil.$$

Пристрій ввеедення типу «трекбол» (рис. 7.2) забезпечує переміщення графічного маркера в напрямку перекочування кулі зі швидкістю, яку надав кулі оператор. Ввважається, що куля має дуже великий момент інерції.

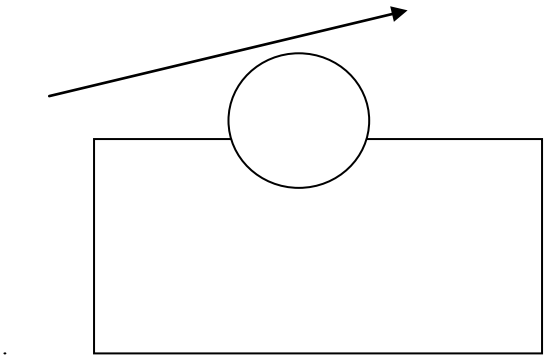


Рисунок 7.2 – Трекбол

У режимі програмної імітації режиму трекболу оператор виконує переміщення реєструвального органу в координатному просторі передекранної сенсорної панелі в напрямку необхідного переміщення графічного маркера. При цьому за допомогою передекранної панелі через виділений проміжок часу ΔT знімаються значення координат реєструвального органу на початку інтервалу та в його кінцевій точці. Обчислювальними засобами розраховують різницю координат, яка, спільно зі знаками, задає напрямок руху графічного маркера, а модуль різниці – швидкість його переміщення.

При досягненні маркером вибраної заданої зони оператор поміщає в зоні передекранної сенсорної панелі орган реєстрації й утримує його нерухомо. Оскільки в даному випадку переміщення немає, то різниця координат реєструвального органу дорівнює нулю, що призведе до зупинки переміщення графічного маркера на екрані.

Пристрій введення типу «джойстик» (рис. 7.3) забезпечує переміщення графічного маркера в напрямку нахилу рукоятки зі швидкістю, яка пропорційна куту нахилу.

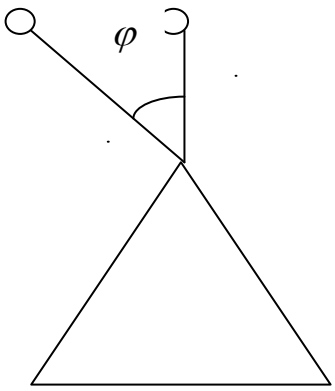


Рисунок 7.3 – Джойстик

В режимі програмної імітації джойстика умовний центр останнього розміщують в центрі екранної системи координат. При необхідності переміщення графічного маркера в необхідному напрямку оператор поміщає реєструвальний орган на таку позицію, яка відносно умовного центра джойстика має такий же напрямок. В цьому випадку віддалення Δ реєструвального органу від центра екранної системи координат визначає швидкість переміщення графічного маркера (рис. 7.4). При необхідності зупинки маркера оператор поміщає реєструвальний орган в умовний центр джойстика. Оскільки в цьому випадку $\Delta = 0$, то швидкість переміщення також буде дорівнювати нулю і, як наслідок, переміщення маркера буде припинено.

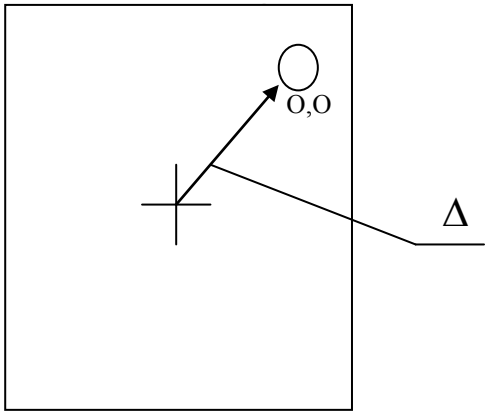


Рисунок 7.4 – Задання для переміщення маркера

Контрольні питання

1. Дайте характеристику основних пристроів введення графічної інформації.
2. Як визначається необхідна кількість оптоелектронних пар для реалізації режиму точного позиціювання?
3. Як здійснюється програмна імітація режимів «трекбол», «джостик» з використанням передекранної сенсорної панелі?

8 ОБРОБКА ТА ФОРМУВАННЯ ГРАФІЧНИХ ФАЙЛІВ

8.1 Робота з кольорами та півтонами

8.1.1 Колір. Системи змішування кольорів

Колір має психофізіологічну і психофізичну природу. Сприймання кольору залежить від фізичних властивостей світла, тобто електромагнітної енергії, від його взаємодії з фізичними речовинами, а також від їхньої інтерпретації зоровою системою людини.

Зорова система людини сприймає електромагнітну енергію з довжинами хвиль від 400 до 700 нм як видиме світло. Світло сприймається або безпосередньо від джерела, наприклад електричної лампочки, або безпосередньо при відображенні від поверхні об’єкта або заломленні в ньому.

Джерело або об’єкт є ахроматичним, якщо світло, що спостерігається, містить всіх видимі довжини хвиль в приблизно однаковій кількості. Ахроматичне джерело здається білим, а відбите або заломлене ахроматичне світло – білим, чорним або сірим. Білими здаються об'єкти, що ахроматично відбивають більше 80% світла білого джерела, а чорними – менше 3%. Проміжні значення дають різноманітні відтінки сірого. Інтенсивність відбитого світла зручно розглядати в діапазоні від 0 до 1, де 0 відповідає чорному, 1 – білому, а проміжні значення – сірому кольору.

Яскравість об’єкта залежать від відносної чутливості ока до різних довжин хвиль. При денному світлі чутливість ока максимальна при довжині хвиль порядку 550 нм, а на краях видимого діапазону спектра вона різко падає.

Якщо світло, що сприймається, містить довжини хвиль в довільній нерівній кількості, то воно називається хроматичним. Якщо довжини хвиль сконцентровані біля верхнього краю видимого спектра, то світло здається червоним, тобто, домінуюча довжина хвиль лежить в червоній області видимого спектра. Якщо довжини хвиль сконцентровані в нижній частині видимого спектра, то світло здається синім, тобто домінуюча довжина хвиль лежить в синій частині спектра. Проте сама по собі електромагнітна енергія певної довжини хвилі не має ніякого кольору. Відчуття кольору виникає в результаті перетворення фізичних явищ в оці й мозку людини. Колір об’єкта залежить від розподілу довжин хвиль джерела світла і від фізичних властивостей об’єкта. Об’єкт здається кольоровим, якщо він відбиває або пропускає світло лише у вузькому діапазоні довжин хвиль і поглинає всі інші.

Психофізіологічне подання світла визначається кольоровим тоном, насиченістю і світлотою. Кольоровий тон дозволяє розрізняти кольори, а насиченість – визначати ступінь ослаблення (розбавлення) даного кольору білим. Насиченість чистого кольору складає 100% і зменшується зі збільшенням білого. Насиченість ахроматичного кольору складає 0%, а його світлота рівна інтенсивності цього світла.

Зазвичай зустрічаються не чисті монохроматичні кольори, а їхні суміші. В основі трикомпонентної теорії світла лежить припущення про те, що в центральній частині сітківки знаходяться три типи чутливих до кольору колбочок. Перші сприймають довжини хвиль, що лежать в середині видимого спектра, тобто зелений колір; другі – довжини хвиль біля верхнього краю видимого спектра, тобто червоний колір, треті – короткі хвилі нижньої частини спектра, тобто сині. Відносна чутливість ока максимальна для зеленого кольору і мінімальна для синього. Якщо на всі три типи колбочок вплине однаковий рівень енергетичної яскравості (енергія в одиницю часу), то світло здається білим. Природне біле світло містить всі довжини хвиль видимого спектра, однак відчуття білого світла можна отримати, змішуючи будь-які три кольори, якщо жоден з них не є лінійною комбінацією двох інших. Це можливо завдяки фізіологічним властивостям ока, яке має три типи колбочок. Такі три типи кольорів називаються основними.

В машинній графіці застосовуються дві системи змішування основних кольорів: адитивна – червоний, зелений, синій (RGB) і субтрактивна – блакитний, пурпурний, жовтий (CMY). Вони зображені на рис. 8.1.

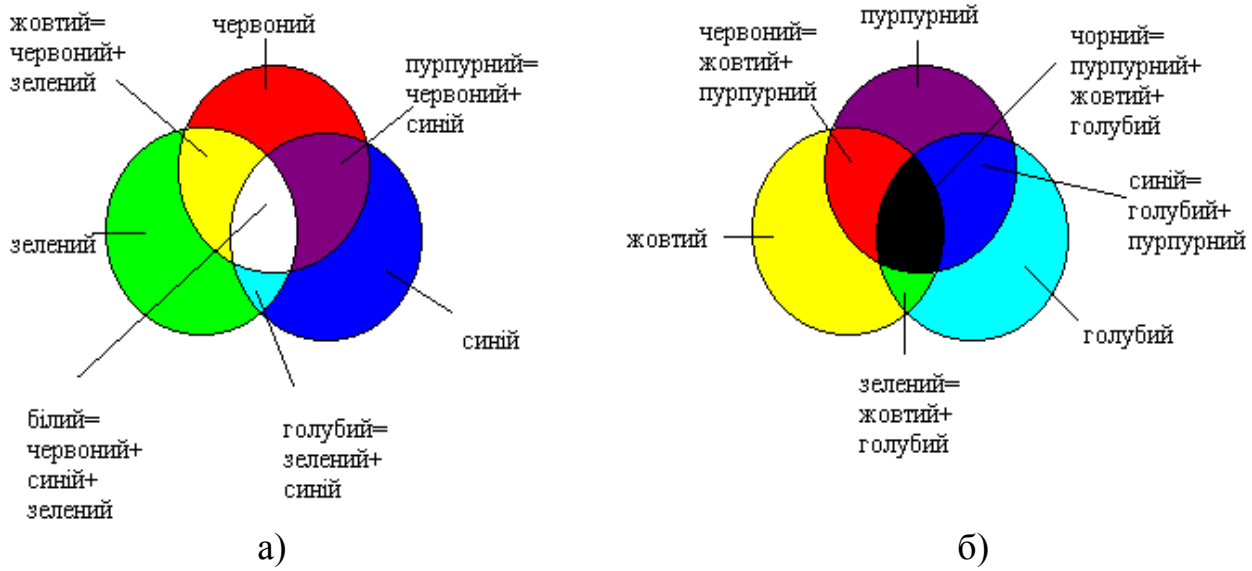


Рисунок 8.1 – Адитивна (а) і субтрактивна (б) системи змішування кольорів

Кольори однієї системи є додатковими до іншої: блакитний – до червоного, пурпурний – до зеленого, жовтий – до синього. Додатковий колір – це різниця білого і даного кольору: блакитний – це білий мінус червоний, пурпурний – білий мінус зелений, жовтий – білий мінус синій. Хоча червоний можна вважати додатковим до блакитного, за традицією червоний, зелений і синій вважаються основними кольорами, а голубий, пурпуровий і жовтий – їх додатками. Цікаво, що в спектрі веселки або призми пурпурного кольору немає, тобто він породжується зоровою системою людини.

Для відбивальних поверхонь, наприклад типографських фарб, плівок і екранів, що не світяться, застосовується субтрактивна система CMY. В субтрактивних системах зі спектра білого кольору віднімаються довжини хвиль додаткового кольору. Наприклад, при відбитті або пропусканні світла крізь пурпуровий об’єкт поглинається зелена частина спектра. Якщо отримане світло відбивається або заломлюється в жовтому об’єкті, то поглинається синя частина спектра і залишається лише червоний колір. Після його відбиття або заломлення в блакитному об’єкті колір стає чорним, бо при цьому вилучається весь видимий спектр. За цим принципом працюють фотофільтри.

Адитивна колірна система RGB зручна для поверхонь, які мають здатність світитися, наприклад екранів ЕПТ або кольорових ламп.

Контрольні питання

1. Які довжини хвиль сприймає зорова система людини?
2. Чим визначається психофізіологічне подання світла?
3. В яких випадках застосовуються адитивна колірна система RGB та субтрактивная колірна система CMY?

8.1.2 Палітри й оптимізація палітр

Палітра визначає кольори, що будуть використовуватися в картині. Якщо виникне необхідність, то додаткові кольори можна одержати змішуванням фарб, уже внесених у палітру.

У комп’ютерах теж використовуються палітри, але комп’ютерні колірні палітри більш обмежені. Дійсно, ілюстративна програма, що підтримує максимум 256 кольорів, ніяк не зможе використовувати більше 256 кольорів одночасно. Залишається єдина надія на псевдотонування. Але навіть ювелірно налаштоване дифузійне псевдотонування буде навряд чи ефективно, якщо кольори палітри погано підібрані. У кінцевому підсумку програмними засобами підбирають фарби палітри, які якнайкраще відповідають відтінкам зображення.

Однієї з проблем, що виникають час від часу в комп’ютерній графіка, є проблема вибору колірної палітри, за допомогою якої можна відобразити

картинку з кількістю відтінків, що перевищують кількість кольорів у палітрі. Яка комбінація кольорів дасть найкращий результат? Відповідь на це питання дає процес, який називають оптимізацією палітри. Найпростіший підхід полягає в тому, щоб, перебравши всі пікселі в зображенні, порахувати скільки разів зустрічається кожний колір і скласти палітру з тих кольорів, що зустрічаються найчастіше. Якщо деякий відтінок синього кольору зустрічається 100 разів, а відтінок червоного тільки 20, то перевага віддається синьому кольору. Але цей метод має декілька недоліків. Один із них полягає в тому, що деякі кольори будуть вилучені повністю. Уявіть собі зображення заміської дороги, де переважають сині, коричневі, жовті, зелені тони, і десь в одному куту виявився червоний, дорожній знак «Стій». Якщо червоний колір більше ніде на цій картинці не зустрічається, те він не потрапить у палітру, і, отже, буде пофарбований у якийсь інший колір.

Можливо, краще було б вибрати комплект кольорів для палітри з рівномірно розподіленими червоною, зеленою і синьою компонентами. Такий підхід забезпечує широкий вибір кольорів, але при цьому не враховується той факт, що в більшості зображень немає рівномірного колірного розподілу.

Інше вирішення проблеми – це метод квантування кольорів медіанним перетином. Колірний простір розглядається як тривимірний куб. Кожна вісь куба відповідає одному з трьох основних кольорів: червоному, зеленому або синьому (розглянемо для прикладу випадок, коли використовують 256 кольорів). Кожна з трьох сторін розбивається на 255 рівних частин, Розподіли на осях нумеруються від 0 до 255, причому більше значення відповідає більшій інтенсивності кольору. Точки всередині куба, що відповідають кольорам, відмічаються так само, як точки тривимірного графіка, якби були задані x, y і z координати точки. Приміром, якщо значення червоної, зеленої і синьої компонент є 128, 64 і 192 відповідно, то потрібно відкласти на осі Ч 128, на осі З – 64, на осі С – 192 і одержати точку всередині куба, що відповідає даного кольору. Чорний колір із компонентами 0, 0, 0 потрапить на одну вершину куба, а білий – з компонентами 255, 255, 255 – в іншу, діагонально протилежну вершину. Якщо відзначити точки всередині куба, що відповідають кольорам пікселів у звичайному повноколірному зображенні, то виявиться, що точки нерівномірно розташовані по всьому кубові. Очевидна тенденція групування точок в окремих регіонах.

Метод медіанного перетину поділяє куб на 256 паралелепіпедів, кожний з який містить приблизно однакову кількість пікселів. При такому розбитті куба центральна точка кожного паралелепіпеда є оптимальним вибором для колірної палітри. У тій ділянці куба, що густо заповнена точками, буде більше паралелепіпедів і, відповідно, у палітру потрапить більше кольорів. А там, де точок менше, і кольорів буде взято менше. Жодний колір не буде відкинутий повністю. Тим же кольорам, що зустрічаються частіше, буде віддана перевага.

Звернемося ще раз до прикладу з заміською дорогою. Кольори в палітрі, отриманій медіанним перетином, будуть концентруватися навколо синього, коричневого, жовтого і зеленого, але, принаймні, знайдеться один відтінок із достатньою червоною компонентою, щоб апроксимувати колір знака «Стій».

Метод квантування кольорів медіанним перетином

Метод квантування кольорів за допомогою медіанного перетину застосовується при виборі 256 кольорів, щоб подати повноколірне зображення, що містить декілька тисяч кольорів. Щоб зрозуміти як працює метод медіанного перетину, уявімо колірний простір як куб. Кожна вісь відповідає одному з трьох основних кольорів і розподіли на осі нумеруються від 0 до 255; більшому номеру відповідає більша інтенсивність кольору. Кольори в зображенні відзначаються всередині куба так само, як точки на тривимірному графіку.

Перший крок полягає у відсіканні «країв» куба, що не містять пікселів. Приміром, якщо у всіх пікселів значення червоної компоненти не менше, ніж 8 і не більше, ніж 250, то відкидаються частини куба від Ч = 0 до Ч = 7 і від Ч = 251 до Ч = 255.

Другий крок полягає в розрізуванні отриманого паралелепіпеда на два в середній точці (медіані) найдовшої сторони. Якщо найдовша сторона паралельна осі С, то комп’ютер вибирає серединне синє значення з усіх синіх значень, поданих у паралелепіпеді, і розрізає в цій точці. Тепер паралелепіпед розділений на два паралелепіпеди меншого розміру, що містять однакову кількість пікселів.

Весь попередній процес – відсікання порожніх «країв» і розрізування найдовшої сторони в середній точці – повторюється для двох менших паралелепіпедів. Тепер вихідний куб розділений на чотири паралелепіпеди, що містять приблизно однакову кількість пікселів.

Медіанний перетин повторно застосовується для того, щоб розділити куб на 8, 16, 32, 64, 128 і 256 паралелепіпедів. Вони містять приблизно ту саму кількість пікселів, а їх об’єми обернено пропорційні щільностям пікселів.

Маючи простір, розділений таким чином, легко вибрати палітру. Кожний із 256 паралелепіпедів містить піксели приблизно однакового кольору, і центр кожного паралелепіпеда подає оптимальне значення кольору для палітри. Маючи координати вершин, дуже просто обчислити координати центральної точки. (Деякі графічні програми замість того, щоб обчислювати центральну точку, усереднюють значення всіх пікселів, які знаходяться всередині паралелепіпеда; на обчислення піде більше часу, але отримана палітра буде кращою). Вирахувавши Ч, З и С координати для всіх 256 центральних точок у паралелепіпедах, одержимо 256 кольорів, що і будуть складати палітру.

Контрольні питання

1. Що таке палітра?
2. В чому полягає оптимізація палітри?
3. Характеристика методу квантування кольорів медіанним перетином.

8.1.3 Апроксимація півтонами

Апроксимація півтонами – це метод з використанням мінімальної кількості рівнів інтенсивності для отримання великої кількості півтонів.

Стефаном Хагеном в 1880 році запроваджений метод півтонового друкування, в якому велика кількість фотографічних півтонів сірого кольору реалізується за допомогою дворівневого середовища: чорна фарба на білому папері.

Півтоновий друк використовує дискретні клітини. Розмір клітини вибирається залежно від мілкозернистості решітки і тривалості експозиції. Для газетного друку використовується 50–90 точок на дюйм.

Метод півтонів базується на властивості зорової системи інтегрувати, тобто згладжувати дискретну інформацію.

На противагу півтоновому друку, в якому використовуються змінні розміри клітин, в методі конфігурування розміри кліток фіксовані. Для зображення з фіксованою розподільністю декілька пікселів об’єднуються в конфігурації. На рис. 8.2 показана одна з можливих груп конфігурації для дворівневого чорно-білого дисплею. Для кожної клітини використовуються чотири піксели. При такій організації отримуємо п’ять можливих рівнів або тонів сірого кольору. При виборі конфігурацій потрібно пильнувати, щоб не виникли несприятливі дрібномасштабні структури. Наприклад, не належить застосовувати ні одну з конфігурацій, зображених на рис. 8.2, b, c., оскільки це може призвести до того, що для великої ділянки з постійною інтенсивністю на зображенні проявляться небажані горизонтальні або вертикальні смуги.

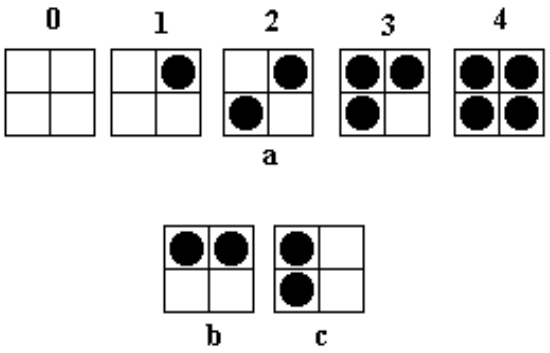


Рисунок 8.2 – Дворівневі конфігурації 2×2

Кількість доступних рівнів інтенсивності можна збільшити за допомогою збільшення розміру клітини. Конфігурації для клітини 3×3 забезпечують десять рівнів інтенсивності.

Клітини не обов'язково мають бути квадратними. На рис. 8.3 зображена клітина 3×2 пікселів, яка дає сім рівнів інтенсивності.

Якщо є можливість використовувати різні розміри точок, то кількість рівнів інтенсивності збільшується.

Використання конфігурацій веде до втрати просторового розподілення, що прийнятно у випадку, коли роздільна спроможність зображення менша, ніж дисплея.

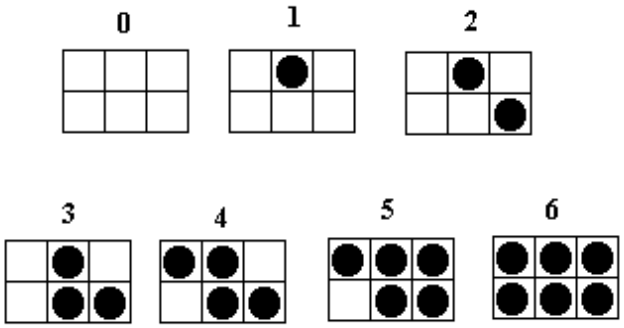


Рисунок 8.3 – Дворівневі конфігурації 3*2

Розроблені методи покращання візуального розподілення. Найпростіший з них полягає у застосуванні порогового значення для кожного піксела. Якщо інтенсивність зображення перевершує деяку порогову величину, то піксел вважається білим, в протилежному випадку – чорним. орогову величину, як правило, встановлюють рівною половині максимальної інтенсивності.

При пороговому методі втрачається велика кількість дрібних деталей. Особливо це характерно для волосся та рис обличчя. Дрібні деталі втрачаються через відносно великі помилки відображуваної интенсивності для кожного піксела.

В методі, розробленому Флойдом та Стейнбергом, ця похибка розподіляється на сусідні пікселі. Розподіл похибки виконується завжди вниз і вправо. Таким чином, при генерації зображення в порядку сканування повертатись зворотно не потрібно. Зокрема, в алгоритмі Флойда–Стейнберга 3/8 похибки розподіляється вправо, 3/8 – вниз і 1/4 – по діагоналі. Поріг інтенсивності вибирають рівним T = (білий + чорний)/2 (рис. 8.4).

Розподіл похибки на сусідні пікселі покращує вигляд деталей зображення, оскільки інформація про мілкі не втрачається.

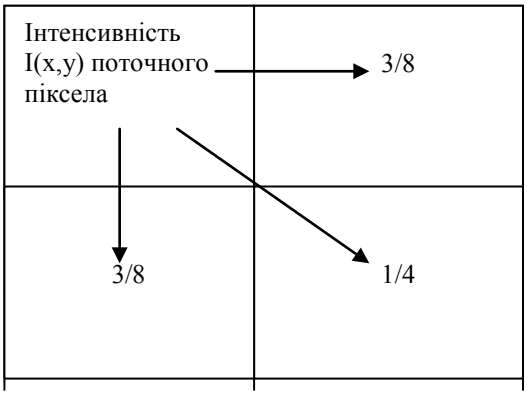


Рисунок 8.4 – Розподілення інтенсивності в алгоритмі Флойда–Стейнберга

Контрольні питання

1. В яких випадках застосовується апроксимація півтонами?
2. Дайте характеристику методу Хагена.
3. Чим визначається кількість півтонів в методі Хагена?
4. В чому суть методу Флойда–Стейнберга?

8.1.4 Методи псевдотонування

Піксел може приймати різні відтінки. Будь-яке зображення, незалежно від його складності, – це сукупність пікселів заданих відтінків.

Змінювати колір кожного піксела можна незалежно, але кількість відтінків, що одночасно можуть бути присутніми на екрані, обмежена і залежить від графічного устаткування. До однієї межі діапазону відносять монохромні системи, що дозволяють відображати тільки два кольори. До протилежної границі відносяться повноколірні системи, які відображають 16,7 мільйона кольорів. Максимальна кількість кольорів, які одночасно відображаються на екрані, визначаються кількістю бітів, виділених для кожного піксела у відеобуфері. У повноколірних системах кожному пікселу виділяється 24 біти колірної інформації: вісім – для червоної компоненти кольору, вісім – для зеленої і вісім – для синьої. Більшим числам відповідають більш яскраві кольори. 24-бітове число може бути в межах від 0 до 16777215. Це означає, що відеоадаптер може відобразити більше 16,7 мільйона кольорів. Змішуючи різні інтенсивності червоної, зеленої і синьої компонент можна одержати практично будь-який колір.

Біти у відеобуфері, що відповідають тому чи іншому пикселу, не обов’язково вказують його колір безпосередньо. У системах із 256 кольорами (тільки 8 бітів на піксел) значення з відеобуфера вказує на один з 256 рядків у таблиці, яка отримала назву *колірної палітри*. Число, що знаходиться в цьому

рядку палітри, визначає колір піксела. Якщо палітра складається з 24-бітових значень, то відеоплата може відобразити будь-який зі 16,7 млн кольорів.

Як правило, чим більше пікселів на экрані, тим вища якість зображення. Часто користувач поставлений перед вибором кількості кольорів і роздільності. Той самий відеоадаптер дозволяє одержати 256 кольорів при роздільності 1024 на 768, або 16 кольорів при роздільності 1280 на 1024. Так що ж важливіше: вища роздільність чи більше відтінків? Якщо на екрані потрібно одержати зображення фотографічної якості, то важливіші відтінки. Зображення низької роздільності, що містить 256 кольорів, виглядає більш реалістичним, ніж зображення з 16 кольорів.

Один із шляхів, що дозволяють компенсувати нестачу наявних кольорів, – це *псевдотонування* (dithering) комп'ютерного зображення. Існує багато варіантів псевдотонування, але всі вони базуються на одному принципі, а саме: замінити піксели з кольорами, які відсутні в палітрі, конфігураціями пікселів із кольорами з палітри. Псевдотонування ґрунтується на тому, що людське око змішує кольори двох сусідніх пікселів, що знаходиться поряд, сприймаючи деякий третій колір. Використовуючи алгоритм псевдотонування, можна було б замінити блок зелених пікселів конфігурацією (візерунком) із жовтих і синіх пікселів, які чергуються. Цей процес змішування кольорів називається візерунковим псевдотонуванням. Проблема, однак, полягає в тому, що іноді групи незалежних пікселів у сукупності утворять вторинні візерунки, які отримали назву артефактів. Більш сприйнятним є дифузійне псевдотонування, в якому не використовуються заздалегідь підготовлені колірні конфігурації (візерунки). В даному випадку аналізується кожний піксел зображення. Його новий колір вибирається так, щоб відмінність нового кольору від вихідного було мінімальною. Потім обчислюється внесена похибка, тобто різниця між новим і попереднім кольорами, і ця похибка розподіляється між сусідніми пікселами, злегка змінюючи їхні відтінки. Наприклад, якщо новий колір піксела містить менше червоного та зеленого, ніж попередній, то дифузійне псевдотонування додасть трохи червоного та зеленого відтінку сусіднім пікселам. Такий адаптивний підхід унеможливлює появу артефактів і, як правило, приводить до задовільних результатів.

Псевдотонування можна також використовувати для одержання чорно-білих копій кольорових зображень на таких монохромних пристроях, як принтерах. Подібний процес отримав в поліграфії назву *автотипії*. Він використовується для одержання півтонових зображень при друкуванні газет.

При застосуванні візерункового псевдотонування результат виглядає менш привабливим, ніж при використанні дифузійного псевдотонування. Зображення здається зернистим і видимі значні артефакти. Візерункове псевдотонування вимагає значно менших обчислювальних затрат. В деяких

графічних програмах візерункове псевдотонування використовується для попереднього перегляду, а дифузійне псевдотонування для виведення кінцевого результату.

Дифузійне псевдотонування розподіляє колірну помилку – різницю між фактичним кольором піксела і бажаним – на всі пікселі так, щоб сумарна колірна похибка була нульовою для всього зображення.

Перший крок у процесі псевдотонування – це вибір колірної палітри. Якщо над зображенням виконується 16-колірне псевдотонування, то відповідна програма має вибрати палітру з тих 16 кольорів, що найкраще підходять для вихідного діапазону кольорів. Один із способів – це порахувати скільки разів зустрічається кожний колір і вибрати ті 16, які найчастіше зустрічаються

Починаючи з першого піксела зображення в лівому верхньому кутку, комп'ютер вибирає з палітри колір, який найменше відрізняється від вихідного кольору піксела. Будемо вважати, що в палітрі значення червоної, зеленої і синьої компонент кольору дорівнюють, відповідно, 192, 64 і 64, а колір піксела вихідного зображення – 202, 96, 58. Колірна похибка обчислюється для кожної компоненти. У даному випадку похибка для червоної компоненти буде дорівнювати 202 – 192 = 10, для зеленої 32 і для синьої – 6.

Значення помилок, які були обчислені на попередньому кроці, мають бути розподілені серед сусідніх пікселів, використовуючи фільтр Флойда – Стейнберга. Піксел справа одержить 7/16 помилки, пиксел зліва знизу одержить 3/16, піксел знизу одержить 5/16 і пиксел справа знизу одержує 1/16. Сума цих дробів дорівнює одиниці. Ця необхідна умова, якщо похибка має розподілятись повністю. Ці дроби збільшуються на помилку і додаються до відповідних пікселів. Наприклад, червона компонента правого піксела має збільшитися на 10*7/16, тобто на 5. Зелена компонента збільшується на 32*7/16 (14) і синя компонента зменшується на 6*7/16 (3). Коли крок завершено, пікселу в лівому верхньому кутку встановлюється значення з палітри, а значення кольорів трьох його сусідніх пікселів змінюються.

Цей процес повторюється для кожного піксела на екрані. Коли закінчено рядок, сканування починається з лівого піксела наступного рядка.

Контрольні питання

1. В чому полягає сутність псевдокодування?
2. Дайте зіставну характеристику дифузійного та візерункового псевдотонування.
3. Як виконується дифузійне псевдокодування?

8.2 Основні режими занесення інформації в відеопам'ять

Серед режимів занесення інформації в відеопам’ять найбільш поширеними є режими заміщення, накладання, зворотного читання, стирання (режими ReGis), а також режими, які використовують різні логічні операції.

Режими занесення інформації в відеопам’ять використовують регістри переднього плану (зберігає код кольору, яким виконується кресленик), регістр фону (зберігає код кольору фону), регістр маски (зберігає структуру зіставних компонент графічних елементів), а також регістр даних відеопам’яті (рис. 8.5).

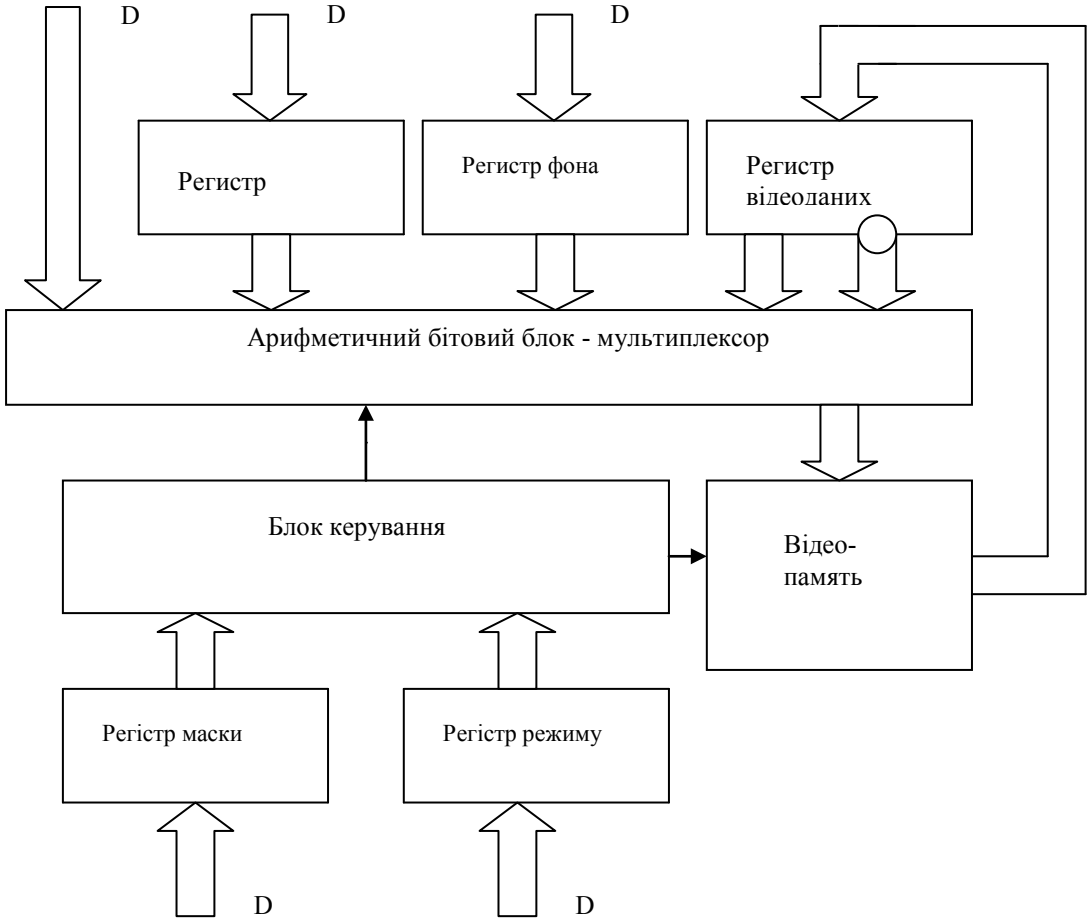


Рисунок 8.5 – Структурна схема блока обробки відеоданих

Регістр маски виконують як зсувний. Одиничний розряд, який отримують після зсуву, визначає видимість поточного піксела на екрані.

В режимі заміщення при одиничному значенні розряду регістра маски в комірку відеопам’яті заносять вміст регістра переднього плану, а при нульовому – вміст регістра фону. Такі дії призводять до стирання тих

графічних компонент, які адресно співвідносились до проміжків часу, коли на виході регістра маски формувався рівень логічного нуля. Вказані компоненти заміщаються кольором фону.

В режимі накладання таке заміщення відсутнє, для чого при нульовому значенні старшого розряду регістра маски запис у відеопам'ять забороняють. Таким чином на попереднє зображення буде накладено зображення компонент, видимість яких визначається вмістом регістра маски.

В режимі стирання при одиничному значенні старшого розряду регістра маски в відеопам'ять заноситься вміст регістра фону.

В режимі зворотного читання виконується зчитування вмісту комірки відеопам'яті, на місце якого заносять інверсію отриманого коду. Вказаний режим можна застосувати для анімації.

Для того, щоб змусити об'єкт рухатись по екрану, потрібно за допомогою операції «зворотне читання» записати його в відеопам'ять двічі. Як тільки об'єкт буде поміщений туди перший раз, його зображення з'явиться на екрані. При виконанні операції другий раз об'єкт з екрана буде стерто. Якщо повторно змінювати розміщення об'єкта, то створиться враження, що він рухається по екрану.

Контрольні питання

1. Які регістри використовуються в різних режимах занесення інформації у відеопам'ять?
2. Вкажіть основні дії при занесенні інформації у відеопам'ять для різних режимів.
3. В якому з режимів не використовується регістр маски?

8.3 Стиснення графічних зображень

Растрові файли мають дуже великі розміри. Якщо знехтувати заголовками файлу та іншими неграфічними даними, то його розмір пропорційний кількості пікселів у зображенні і кількості бітів, потрібних для подання кожного піксела. Повноколірне зображення розміром 1024×768 пікселів займає більше двох мегабайтів пам'яті, а одна секунда відеофільма телевізійної якості в растровому вигляді вимагає біля тридцяти мегабайтів. Тому жорсткий диск можна заповнити миттєво. Навіть компакт-диск, що вміщає біля 700 мегабайтів даних, не настільки великий, щоб помістити такий об'єм інформації.

Використовуючи метод, який називається стисненням зображень, можна різко зменшити в розмірі графічні файли. При стисненні графічної інформації використовуються спеціальні прийоми, що зменшують кількість байтів, необхідних для подання зображення. Ступінь стиснення залежить від

методу стиснення і вмісту графічного файлу. Як правило, графічний файл стискається в п’ять і більше разів. Існують методи, що стискують ще сильніше, але з втратами якості. При відновленні зображення втрачається деяка частина колірної інформації. У підсумку, розпаковане зображення може стати злегка розмитим або знебарвленим.

Методи стиснення растрової інформації діляться на дві великі групи: стиснення з втратами і стиснення без втрат. *Методи стиснення без втрат* дають більш низький коефіцієнт стиснення, але зате зберігають точне значення пікселів вихідного зображення. *Методи з втратами* дають більш високі коефіцієнти стиснення, але не дозволяють відтворити початкове зображення з точністю до піксела. Для файлів, які формуються програмами автоматизованого проєктування, дуже важливо зберегти всю інформацію, тому що втрата хоча б одного біта може змінити зміст усього файлу. Зовсім інша справа з растровими даними. Людське око не сприймає всі відтінки кольору в звичайному растровому зображенні. Таким чином, деякі деталі можуть бути опущені без видимого порушення інформаційного змісту зображення.

Розглянемо два найбільш розповсюджені методи стиснення зображень. Спочатку познайомимося з одним із варіантів групового кодування (run-lenght encoding – RLE). Ідея методу полягає в тому, що послідовність значень, які повторюються, заміняється парою чисел: одне з них вказує на довжину групи (число повторень даного значення), а інше – на власне це значення. Це дуже загальний і дуже простий метод без втрат. Він використовується в багатьох популярних сьогодні форматах графічних файлів і, зокрема, у PCX і BMP. У його основі лежить той факт, що багато зображень надлишкові, оскільки містять велику кількість суміжних пікселів одного кольору. Розглянемо, наприклад, як за допомогою групового кодування стискається зображення, у якому зустрічається підряд 100 пікселів із нульовим значенням. Ця послідовність зі 100 нулів кодується парою чисел (100, 0). Отже, такий фрагмент картинки скоротиться у п’ятдесят разів.

Інший метод, яким користуються досить часто, – JPEG (метод, що стискує з втратами) одержав свою назву від абревіатури об’єднаної групи експертів у сфері фотографії (Joint Photographic Expert Group – JPEG), що його і розробила. JPEG широко використовується при стисненні статичних зображень. Цей метод істотно складніший, ніж RLE. Основна ідея методу перебуває в поділі інформації в зображенні за рівнем важливості, і потім відкиданні менш важливої її частини, зменшуючи тим самим загальний обсяг збережених даних. Це досягається перетворенням матриці колірних значень у матрицю амплітуд, що відповідають визначеним частотам розкладання зображення. (Звукові коливання, наприклад, можна розкласти математичними методами на прості синусоїдальні гармоніки різних амплітуд і частот, що при додаванні відтворюють вихідний сигнал). Рядок або стовпець пікселів зображення теж можна подати амплітудами і частотами.

Мова в даному випадку йде не про спектральний склад світла, а про форму подання кривих, що утворять графіки, якщо значення пікселів служать ординатами. Відзначимо, що формула перетворення матриці пікселів у матрицю амплітуд непроста. JPEG-стиснення відкидає частину високочастотних компонент зображення, залишаючи компоненти з низькими частотами. Людське око менш критичне до високочастотних варіацій кольору, оскільки загальний вигляд зображення визначається низькими частотами. Значення піксела, отримане при відновленні зображення, дещо відрізняється від вихідного значення, тому що частина інформації була загублена, хоча звичайно вони дуже близькі.

У методу JPEG є дуже цікава особливість: користувач може задавати коефіцієнт якості. Високий коефіцієнт якості дозволяє зберегти більше деталей, але при цьому зменшується ступінь стиснення. При низькому коефіцієнті якості ступінь стиснення збільшується, але зображення стає менш чітким. Чим нижчий коефіцієнт якості, тим більша кількість інформації відкидається. Питання в тому, як знайти розумний баланс між ступенем стиснення і якістю зображення. Ефект JPEG-стиснення неоднаковий для різних зображень. Одні зображення можна стиснути в десять разів без особливого погіршення якості, в інших же, навіть при вдвічі меншому коефіцієнті стиснення, виникають неприпустимі спотворення.

Коли будь-який з методів (RLE або JPEG) застосовується до повноколірного зображення, то червона, зелена і синя компоненти стискуються незалежно. Якщо в растровому зображенні використовується палітра або просто відтінки сірого, то значення пікселів можна закодувати в один прохід.

Розглянемо більш детально методи RLE і JPEG.

Алгоритм групового кодування

Якщо уважно проаналізувати растрове зображення, то виявляється, що пікселі одного кольору часто розміщені поруч один з одним. Якщо почати з лівого верхнього кутка зображення і досліджувати пікселі кожного рядка, виписуючи послідовно їхні значення зліва направо, то можна помітити, що зображення складається з множини відрізків, у яких повторюється одне і те число. Кількість пікселів у відрізку будемо називати довжиною відрізка.

Починаючи з першого рядка, програма групового кодування переглядає значення пікселів зліва направо і шукає відрізки пікселів, що повторюються. Кожен раз, коли зустрічаються три або більше пікселів, що йдуть підряд, з однаковим значенням, програма замінює їх парою чисел: перше число вказує на довжину відрізка, друге – на значення пікселів. Число, що визначає довжину відрізка, називають міткою відрізка.

Щоб ідентифікувати серії значень пікселів, що не повторюються, програма також вставляє мітки, що вказують на кількість таких значень у серії. Зарезервований біт необхідний для того, щоб можна було відрізнити

мітку відрізка від мітки серії значень, що не повторюються. Наприклад, у 8-ми бітах можна задати послідовності довжиною до 127 пікселів; восьмий біт у кожній мітці може відрізняти відрізок від серії пікселів, що не повторюються. Точно так само обробляється кожний рядок пікселів і відрізки однакових значень пікселів стискуються в усьому зображенні.

Графічна програма декодує зображення, зчитуючи стиснений файл і відновлює відрізки повторюваних значень пікселів. Зауважимо, що відновлене зображення цілком збігається з оригіналом.

Алгоритм JPEG

Насамперед програма поділяє зображення на блоки – матриці розміром 8×8 пікселів. При використанні методу JPEG час, що затрачається на стиснення зображення, пропорційний квадрату числа пікселів у блоці. Обробка декількох блоків меншого розміру виконується значно швидше, ніж обробка всього зображення повністю.

До значень пікселів застосовується формула, названа дискретним косинусоїдальним перетворенням (Discrete Cosine Transform – DCT). DCT переводить матрицю значень пікселів 8×8 у матрицю значень амплітуд тієї ж розмірності, що відповідає визначеним частотам синусоїдальних коливань. Лівий верхній кут матриці відповідає низьким частотам, а правий нижній – високим.

Коефіцієнт якості, введений користувачем, використовується в простій формулі, що генерує значення елементів іншої матриці 8×8, яка називається матрицею квантування. Чим нижчий коефіцієнт якості, тим більші значення будуть мати елементи матриці.

Кожне значення в матриці, яка була сформована після DCT-перетворення, ділиться на відповідне значення з матриці квантування, потім округляється до найближчого цілого числа. Оскільки великі числа знаходяться в правій нижній половині матриці квантування, то основна частина високочастотної інформації зображення буде відкинута. Тому нижня права частина матриці пікселів буде складатися, в основному, з нулів.

Далі програма зчитує елементи матриці і кодує їх послідовно методами без втрат. Зауважимо, що стиснення істотно залежить від нулів у правій нижній половині матриці. Чим нижчий коефіцієнт якості, тим більше нулів у матриці і, відповідно, тим вищий ступінь стиснення.

Декодування JPEG-зображення починається з кроку оберненого кодування без втрат, у результаті чого відновлюється матриця квантування пікселів.

Значення з матриці пікселів перемножуються на значення з матриці квантування, щоб відновити, наскільки це можливо, матрицю, що була обчислена на кроці застосування DCT. На етапі квантування була загублена деяка частина інформації, тому числа в матриці будуть близькі до початкових, але не буде абсолютного збігу.

Обернена до DCT формула (IDCT) застосовується до матриці для відновлення значень пікселів вихідного зображення. Ще разом відзначимо, що отримані кольори не будуть цілком відповідати початковим через втрату інформації на кроці квантування. Відновлене зображення, при порівнянні з оригіналом, буде виглядати дещо розмитим і знебарвленим.

Контрольні питання

1. Для чого виконують стиснення графічних файлів?
2. Які основні підходи до стиснення Вам відомі?
3. Дайте характеристику обчислювальному процесу при стисненні методом JPEG.

8.4 Корекція графічних зображень

В машинній графіці для покращення або корекції зображень виконують низку маніпуляцій. За потреби червону, зелену і синю компоненти можна змінювати окремо, щоб одержати найкращий колірний баланс. У розпливчастих зображеннях можна збільшити різкість, і, навпаки, чіткі, контрастні зображення можна розмити, імітуючи ефект пом’якшувальних фотофільтрів.

Розглянемо чотири ефекти для корекції зображень: розмивання, збільшення різкості, тиснення і акварельний ефект. При розмиванні перерозподіляються кольори в зображенні і пом’якшуються різкі межі. При збільшенні різкості підкреслюються розходження між кольорами суміжних пікселів і виділяються непомітні деталі. Тиснення перетворить зображення так, що фігури всередині зображення мають вигляд, начебто вони видавлені на металевій поверхні. Акварельний ефект перетворює фотографічне зображення в картинку, начебто б написану аквареллю.

З алгоритмічної точки зору одержання цих ефектів не викликає особливих труднощів. Складається матриця чисел, яку називають ядром згортки. Матриця розміром 3×4 містить три рядки по три числа в кожному. Щоб перетворити один піксел у зображенні перемножуються значення його кольорів на число в центрі ядра. Потім перемножуються вісім значень кольорів пікселів, що оточують центральний піксел, на відповідні їм коефіцієнти ядра. Підсумовуються всі дев’ять значень. В результаті отримуємо нове значення кольору центрального піксела. Цей процес повторюється для кожного піксела в зображенні. Таким чином зображення фільтрується. Коефіцієнти ядра визначають результат процесу фільтрації. Ядро розмивання може, наприклад, складатися із сукупності коефіцієнтів, кожний з який менший 1, а їхня сума складає 1. Це означає, що кожний піксел поглине щось із кольорів сусідів, але повна яскравість зображення залишиться незмінною. (Якщо сума коефіцієнтів більша 1, яскравість збільшиться; якщо менша 1, яскравість зменшиться.) У ядрі різкості

центральний коефіцієнт більший 1, а оточений він від'ємними числами, сума яких на одиницю менша центрального коефіцієнта. У такий спосіб збільшується будь-який існуючий контраст між кольором піксела і кольорами його сусідів.

Розмивання і збільшення різкості

При підготовцу до розмивання цифрове зображення зчитується у пам'ять комп'ютера у вигляді червоної, зеленої і синьої компонент кольору для кожного піксела.

Ядро розмивання розміром 3×3 застосовується до червоної, зеленої і синьої компонент кольору кожного піксела в зображенні. Значення кольору піксела, що знаходиться під центром ядра, обчислюється множенням вагових коефіцієнтів ядра на відповідні значення кольору в зображенні і підсумовуванням результатів.

Підсумкове зображення буде розмитим порівняно з оригіналом, тому що колір кожного піксела поширився серед сусідів. Ступінь розмивання можна збільшити або використовуючи ядро більшого розміру, щоб розподілити кольори серед більшого числа сусідів, або, підбираючи коефіцієнти ядра і зменшуючи вплив центрального коефіцієнта, або фільтруючи зображення ще разом із ядром розмивання.

Збільшення різкості досягається точно так само, як і розмивання, за винятком того, що використовується інше ядро.

При опрацюванні кожного піксела в зображенні використовується ядро різкості розміром 3×3. Червона, зелена і синя колірні складові обробляються окремо і пізніше об'єднуються, щоб сформувати 24-бітове значення кольору. Від'ємні ваги навколо центра ядра збільшують контраст між центральним пікселом і сусідами.

Кінцеве зображення значно чіткіше, ніж оригінал. Процес збільшення різкості просто підвищив існуючий контраст між пікселами. При повторному опрацюванні зображення чіткість може збільшитися ще більше.

Тиснення

Тиснення робиться майже так, як і розмивання та збільшення різкості.

Кожний піксел у зображенні обробляється ядром тиснення розміром 3×3. На відміну від ядер розмивання і різкості, у яких сума коефіцієнтів дорівнює 1, сума ваг у ядрі тиснення дорівняє 0. Це означає, що «фоновим» пікселам (пікселам, що не знаходяться на межах переходу від одного кольору до іншого) присвоюються нульові значення, а нефоновим пікселам – значення, відмінні від нуля.

Після того, як значення піксела оброблено ядром тиснення, до нього додається число 128. У такий спосіб значенням фонових пікселів стане середній сірий колір (червоний = 128, зелений = 128, синій = 128). Суми, що

перевищують 255, можна округлити до 255 або взяти залишок за модулем 255, щоб значення виявилося між 0 і 255.

У такому варіанті зображення контури здаються видавленими над поверхнею. Напрямок підсвічування зображення можна змінювати, змінюючи позиції 1 і -1 у ядрі. Якщо, наприклад, поміняти місцями значення 1 і -1, то має місце реверсування напрямку підсвічування.

Акварелизація

Акварельний фільтр перетворить зображення, і після перетворення воно виглядає так, начебто написано аквареллю.

Перший крок у застосуванні акварельного фільтра – згладжування кольорів у зображенні. Одним із способів згладжування є процес медіанного усереднення кольору в кожній точці. Значення кольору кожного піксела і його 24 сусідів поміщають у список і сортують від меншого до більшого. Медіанне (тринадцяте) значення кольору в списку присвоюється центральному пікселу.

Після згладжування кольорів комп’ютер обробляє кожний піксел у зображенні ядром різкості, щоб виділити межі переходів кольорів.

Остаточне зображення нагадує акварельний живопис. Це лише один приклад, що показує, як можна об’єднувати різні методи опрацювання зображень і домагатися незвичайних візуальних ефектів.

Контрольні питання

1. Які дії з коригування зображень Вам відомі?
2. Як виконується акварелізація?
3. Як виконується розмивання і збільшення різкості зображень?
4. Які дії виконуються при реалізації процедури тиснення?

9 Задачний практикум

В задачному практикуму наведені приклади розв'язання типових задач.

1. Визначити мінімальну кількість інтерполяційних тактів, необхідну для формування вектора ортогональними кроковими приростами.

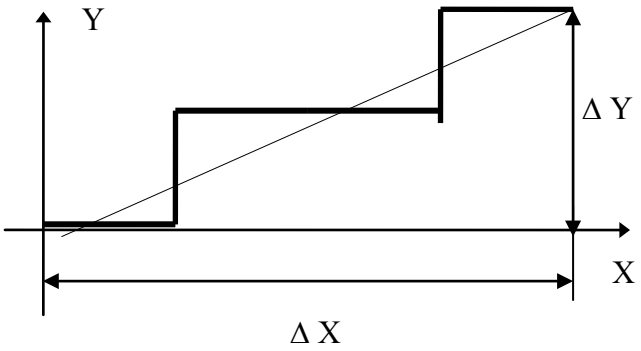


Рисунок 9.1

Формування вектора здійснюється вертикальними та горизонтальними кроковими приростами (рис. 9.1). Кожний з ортогональних кроків унеможливлює переміщення по іншій координаті. Оскільки переміщення в напрямку осі абсцис здійснюється тільки горизонтальними кроками, то їх кількість дорівнює Δ X.

Аналогічно, кількість інтерполяційних тактів в напрямку осі ординат дорівнює Δ Y. Загальна кількість інтерполяційних тактів Δ X + Δ Y.

2. Виконайте інтерполювання відрізка прямої з приростами ΔX = 5, ΔY = 2 за алгоритмом А. М. Пєтуха, Д. Т. Обідника

Інтерполювання за алгоритмом А. М. Пєтуха, Д. Т. Обідника здійснюється за такими формулами:

$$\begin{cases} ОФ_0 = \lfloor БП / 2 \rfloor, \Delta = БП - МП, \\ ОФ_{I+1} = ОФ_i - МП \text{ при } ОФ_i >= 0, \\ ОФ_{I+1} = ОФ_i + \Delta \text{ при } ОФ_i < 0, \end{cases}$$

де БП та МП, відповідно, більше та менше значення приростів ΔX, ΔY.

БП = 5, МП = 2, Δ = 3.

$ОФ_0 = \lfloor 5/2 \rfloor = 2$
$ОФ_1 = 2 - 2 = 0$
$ОФ_2 = 0 - 2 = -2$
$ОФ_3 = -2 + 3 = 1$
$ОФ_4 = 1 - 2 = -1$
$ОФ_5 = -1 + 3 = 2$

При $ОФ_i >= 0$ виконується горизонтальний крок, а при $ОФ_i < 0$ – діагональний. Сформована траєкторія наведена на рис. 9.2.

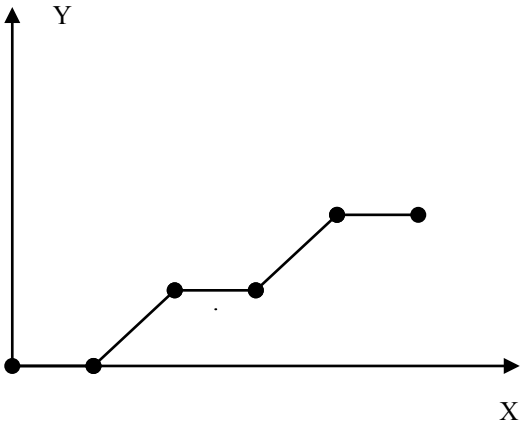


Рисунок 9.2

3. Визначити тип вектора, який формується вертикальними та діагональними кроками, для якого знак оцінювальної функції в кожному інтерполяційному такті змінюється на протилежний, а значення адрес пікселів зменшується.

Оскільки крокова траєкторія формується вертикальними та діагональними приростами, то можна зробити висновок, що у вектора приріст ΔY більший за приріст ΔX.

При додатному знаку оцінювальної функції виконується вертикальний крок, а при від`ємному – діагональний. За умовою задачі знак оцінювальної функції змінюється в кожному інтерполяційному такті, тобто за кожним вертикальним кроком формується діагональний і навпаки.

Оскільки діагональний крок містить приріст по осі Y, то можна зробити висновок, що на кожне переміщення по осі X припадає два переміщення по осі Y, тобто $\Delta Y = 2\Delta X$.

Оскільки адреси пікселів при формуванні вектора зменшуються, то це означає, що він формується в напрямку до початкової точки екрана.

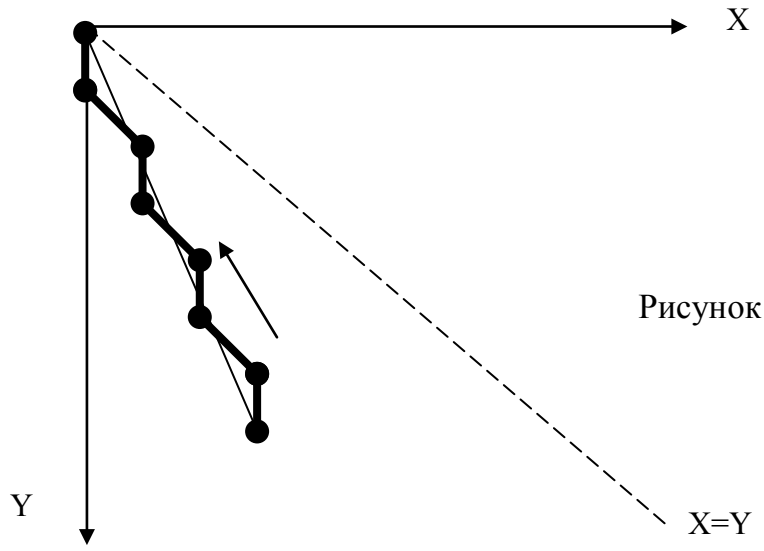


Рисунок 9.3

4. Довести, що максимальна похибка алгоритмів лінійної інтерполяції з восьмивекторним напрямком крокових приростів не може бути меншою половини кроку дискретизації

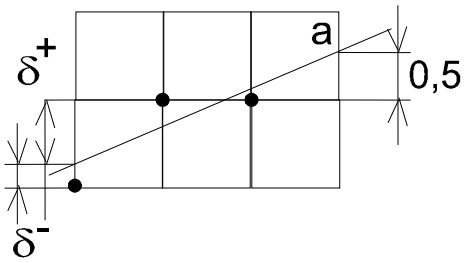


Рисунок 9.4

Крокова траєкторія, яка формується алгоритмами з восьмивекторним напрямком крокових приростів, охоплює горизонтальні, вертикальні та діагональні прирости, що дозволяє вибрати найближчі точки решітки відносно ідеального відрізка прямої.

Відрізок прямої має, щодо найближчих точок решітки дискретного простору, дві похибки: δ^+ та δ^- (рис. 9.4). З двох можливих точок решітки, одна з яких знаходиться вище, а друга – нижче відрізка прямої, доцільно вибрати ту, для якої похибка δ^+ чи δ^- менша за модулем.

У випадку (а), коли ідеальний відрізок прямої проходить через середину дискрети, то $\delta^+ = |\delta^-| = 0{,}5$; тому найближча точка решітки віддалена від вектора на 0,5 кроку дискретизації. В цьому випадку маємо два ідентичних варіанти вибору найближчих точок решітки, які найближче знаходяться до відрізка прямої. Кожна з цих точок віддалена від відрізка на половину кроку дискретизації.

Наведений приклад показує, що досягти меншої похибки в цьому випадку неможливо, тому максимальна похибка алгоритмів лінійної інтерполяції не може бути меншою половини кроку дискретизації.

5. Визначити значення оцінювальної функції в точці (6, 2) при формуванні відрізка прямої згідно з алгоритмом Пєтуха, Обідника за умови, що $\Delta x = 8$, $\Delta y = 3$, $x_n = 0$, $y_n = 0$.

За умовою виконано 4 горизонтальних і 2 діагональних координатних прирости. При формуванні горизонтального кроку значення оцінювальної функції визначається за формулою

$$ОФ_{i+1} = ОФ_i - МП,$$

де МП – менший із приростів. В даному випадку МП = 3.

При виконанні діагонального кроку

$$ОФ_{i+1} = ОФ_i + \Delta,$$

де Δ = БП – МП, $\Delta = 5$.

Підсумовуючи вказане, знаходимо

$$ОФ_6 = ОФ_0 - 4МП + 2\Delta.$$

Для алгоритму Пєтуха, Обідника $ОФ_0 = \lfloor БП/2 \rfloor = 4$. Враховуючи початкове значення оцінювальної функції знаходимо

$$ОФ_6 = 4 – 12 + 10 = 2.$$

6. Визначити тип вектора, який, за алгоритмом лінійної інтерполяції А. М. Пєтуха, Д. Т. Обідника, має всі значення останньої, рівні початковому значенню.

Оцінювальна функція обчислюється за формулами:

$$\begin{cases} ОФ_0 = \lfloor БП/2 \rfloor \\ ОФ_{i+1} = ОФ_i - МП \quad \text{при } ОФ_i \geq 0, \\ ОФ_{i+1} = ОФ_i + \Delta \quad \text{при } ОФ_i < 0, \end{cases}$$

де МП та БП – відповідно, менший та більший прирости відрізка прямої, $\Delta = БП - МП$.

Оскільки початкове значення оцінювальної функції завжди більше нуля, то

$$ОФ_1 = ОФ_0 – МП.$$

При $МП = 0$, $ОФ_1 = ОФ_0 = ОФ_i$, $i = 1, БП$.

7. *Довести, що початкове та кінцеве значення оцінювальної функції при формуванні вектора за алгоритмом лінійної інтерполяції Пєтуха – Обідника рівні.*

Оцінювальна функція розраховується за формулами:

$$ОФ_{i+1} = \begin{cases} ОФ_i – МП \text{ при } ОФi >= 0 \\ ОФ_i + \Delta \text{ при } ОФi < 0 \end{cases}$$

де МП, БП – відповідно, менший та більший прирости,

$\Delta = БП – МП$, $ОФ_0 = int(БП / 2)$.

При $ОФ_i >= 0$ формується горизонтальний (вертикальний) крок, а при $ОФ_i < 0$ – діагональний.

Кількість діагональних кроків дорівнює МП, а кількість горизонтальних (вертикальних) — (БП – МП). Оскільки виконується БП інтерполяційних тактів, кінцеве значення оцінювальної функції дорівнює

$$ОФ_к = ОФ_{БП} = ОФ_0 – МП*(БП-МП) + \Delta*МП = ОФ_0 – МП*\Delta + МП*\Delta = ОФ_0.$$

8. Визначити мінімальну кількість інтерполяційних тактів алгоритмів лінійної інтерполяції з восьмивекторним напрямком крокових приростів.

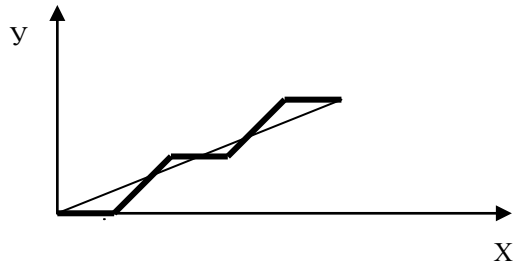


Рисунок 9.5

Формування крокової траєкторії за вказаними алгоритмами здійснюється горизонтальними (вертикальними) та діагональними кроками. Діагональний крок містить в собі одночасне переміщення по обох координатах. При $\Delta x > \Delta y$ крокова траєкторія складається з горизонтальних та діагональних приростів. Враховуючи, що діагональний крок охоплює переміщення в горизонтальному напрямку, робимо висновок, що кількість кроків для досягнення кінцевої точки дорівнює Δx.

Аналогічно можна показати, що при $\Delta y > \Delta x$ потрібно виконати Δy кроків. Таким чином, кількість інтерполяційних тактів для алгоритмів з восьмивекторною орієнтацією крокових приростів дорівнює більшому приросту.

9. Розробити структурну схему лінійного інтерполятора, який формує крокову траєкторію за методом оцінювальної функції.

За базовий алгоритм використаємо алгоритм оцінювальної функції, згідно з яким виконуються такі дії

$$\Delta = \text{БП} - \text{МП}; \quad \text{ОФ}_\text{О} = [\text{БП}/2],$$

де БП, МП – відповідно, більший та менший прирости.

$$ОФ_{I+1} = \begin{cases} ОФ_I - МП & при \;\; ОФ_I \geq 0 \\ ОФ_I + \Delta & при \;\; ОФ_I > 0 \end{cases}$$

Цикл інтерполювання закінчується через БП тактів.

Структурна схема лінійного інтерполятора наведено на рис.9.6.

В регістр БП та МП з вхідної шини D заносяться, відповідно, більший (БП) та менший (МП) прирости. Регістр накопичувального суматора обнуляють. Значення БП заноситься в RG накопичувального суматора (утворений комбінаційним суматором Sm та регістром RG). Через мультиплексор MX на вхід накопичувального суматора подається значення

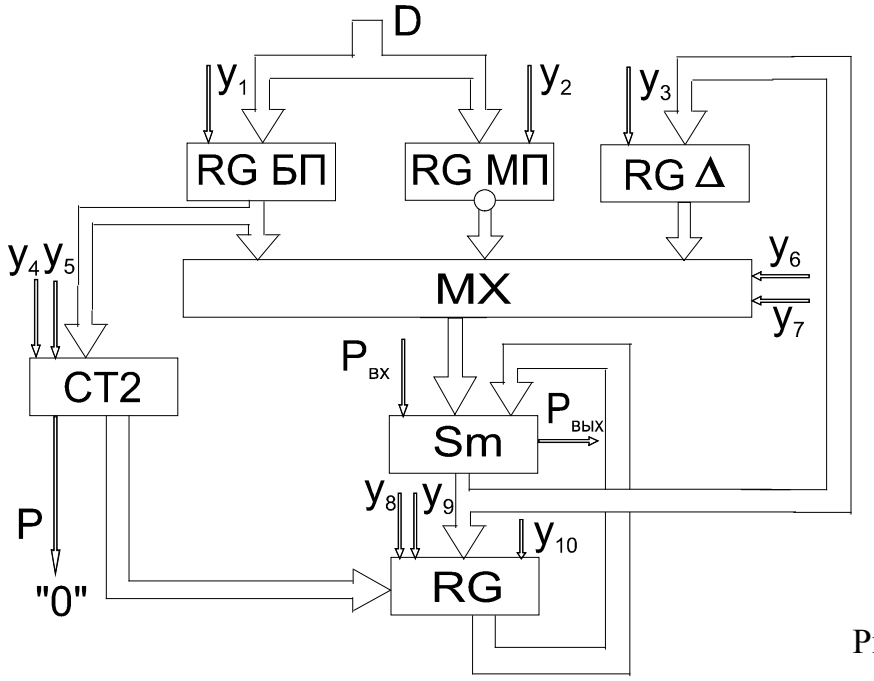


Рисунок 9.6

МП в інверсному коді (оскільки операція віднімання для даного випадку виконується в доповняльному коді, то при її реалізації на вхід переносу накопичувального суматора подається рівень логічної одиниці). Значення БП – МП з виходу суматора Sm заноситься в регістр Δ. В лічильник СТ2 з виходу регістра БП подається значення більшого приросту, яке під дією сигналу $У_4$ записується в лічильник. В регістр RG накопичувального суматора подається значення [БП/2], яке отримуємо монтажним шляхом на виході лічильника СТ2. На цьому закінчується цикл підготовки.

В циклі інтерполяції в кожному такті знаходиться значення оцінювальної функції. Для цього на вхід накопичувального суматора з виходу мультиплексора подається значення Δ або МП.

Знак оцінювальної функції визначає сигнал переносу $P_{вих}$ суматора. З кожним інтерполяційним тактом значення лічильника зменшується на 1. При досягненні лічильником нульового стану інтерполяція закінчується.

10. Визначити значення інтенсивностей кольору точок для забезпечення антиаліайзингу векторної межі з приростами ΔX = 5, ΔY = 2 за умови, що інтенсивність кольору I = 10.

Коефіцієнт похилу вектора дорівнює $\frac{\Delta Y}{\Delta X}$. Задамо кут нахилу відношенням $\frac{I}{I_X}$, тобто, $\frac{\Delta Y}{\Delta X} = \frac{I_X}{I}$. $I_X = \frac{\Delta Y \times I}{\Delta X} = \frac{2 \times 10}{5} = 4$.

Таким чином, $I_X = 4$, $I = 10$. Виконаємо інтерполювання з параметрами I_X, I при кількості інтерполяційних тактів $\Delta X = 5$.

$$БП = 10, \quad МП = 4,$$

$$\Delta = БП - МП = 6,$$

$$ОФ_О = [БП/2] = 5;$$

$$ОФ_1 = ОФ_О - МП = 5 - 4 = 1,$$

$$ОФ_2 = ОФ_1 - МП = 1 - 4 = -3,$$

$$ОФ_3 = ОФ_2 + \Delta = -3 + 6 = 3,$$

$$ОФ_4 = ОФ_3 - МП = 3 - 4 = -1,$$

$$ОФ_5 = ОФ_4 + \Delta = -1 + 6 = 5.$$

Встановлення $I_О = 5$, $I_1 = 1$, $I_2 = 3$, $I_3 = 3$, $I_4 = 1$, $I_5 = 5$ забезпечує згладження ступінчастої форми вектора.

11. Визначити кількість інтерполяційних тактів для формування траєкторії кола ортогональними кроковими приростами.

Розглянемо перший квадрант (рис. 9.7). Крокова траєкторія формується горизонтальними та вертикальними елементарними переміщеннями. Кількість крокових приростів в напрямку координат X дорівнює радіусу R, оскільки вертикальні кроки не забезпечують переміщення по осі абсцис. Аналогічна ситуація має місце і для ординатного напрямку. Таким чином, кількість крокових приростів для формування траєкторії в першому квадранті дорівнює 2R, а для всього кола $2R \times 4 = 8R$.

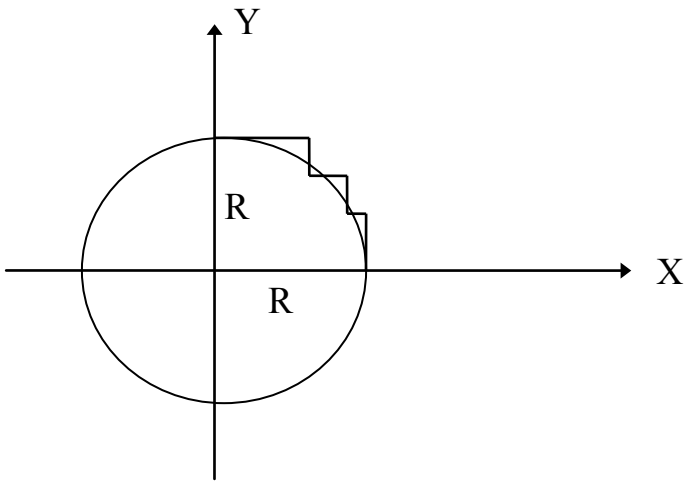


Рисунок 9.7

12. Визначити п'ять точок траєкторії дуги кола, яка починається з точки (6, 0) за умови, що інтерполювання здійснюється за методом оцінювальної функції проти часової стрілки, а радіус кола дорівнює 6.

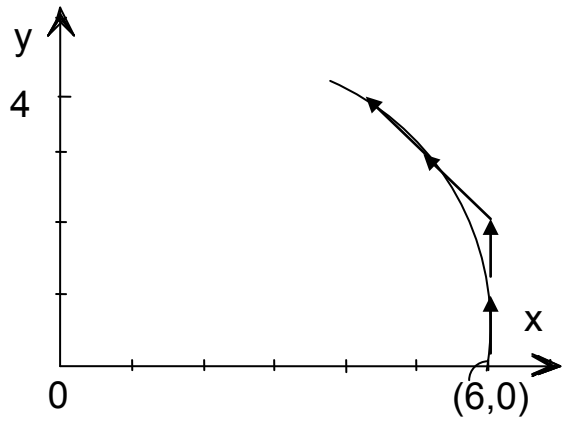


Рисунок 9.8

Оскільки інтерполювання здійснюється проти часової стрілки, то координати X точок траєкторії будуть зменшуватись, а координати У – збільшуватись (див. рис. 9.8). Формули для розрахунку оцінювальної функції мають вигляд:

$$ОФ_{i+1} = \begin{cases} ОФ_i + 2У_i + 1 & \text{при } ОФ < 0 \\ ОФ + 2(y_i - x_i) + 2 & \text{при } ОФ \geq 0 \end{cases}$$

$$ОФ_0 = -R.$$

$$ОФ_0 = -6.$$

$$ОФ_1 = -6 + 2 \times 0 + 1 = -5$$

$$ОФ_2 = -5 + 2 \times 1 + 1 = -2$$

$$ОФ_3 = -2 + 2 \times 2 + 1 = +3$$

$$ОФ_4 = 3 + 2 * (3 - 5) + 2 = +5$$

Згідно зі знаками оцінювальної функції робимо висновок, що в перших двох тактах формуються вертикальні кроки, а в двох наступних – діагональні. Початкова точка дуги має координати (6, 0).

13. Визначити та обґрунтувати напрямки траєкторії в режимі доведення в кінцеву точку дуги.

В першому секторі (рис. 9.9, а) крокова траєкторія формується горизонтальними і діагональними (комбінованими) кроковими приростами.

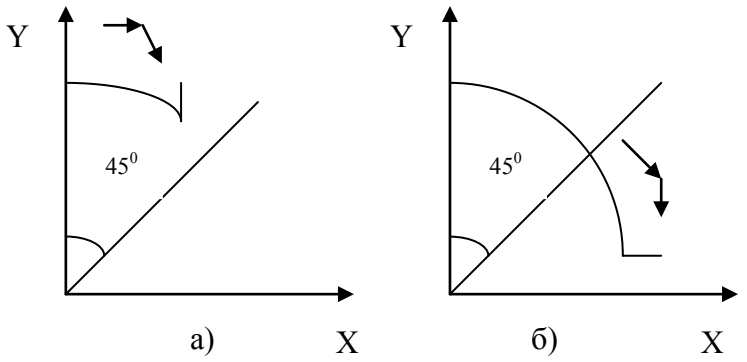


Рисунок 9.9

Враховуючи, що діагональний приріст містить переміщення по осі X, можна констатувати, що всі елементарні кроки до точки X = Y містять переміщення по осі абсцис.

Звідки випливає, що при формуванні траєкторії гарантовано буде досягнуто координату X_k, а доведення буде необхідне до координати Y_k.

Аналогічно можна показати, що при формуванні дуги в другому секторі доведенню підлягає координата X_k.

14. Розрахувати 4 крокові прирости для параболи $Y = 2X^2$ за методом оцінювальної функції.

Оцінювальна функція визначається за формулою $ОФ_i = y – 2x_i^2$.

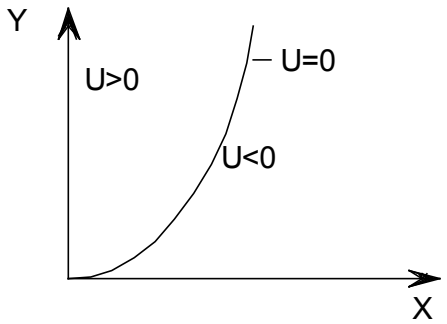


Рисунок 9.10

При ОФ ≥ 0 виконується крок по осі X, а при ОФ < 0 – по осі Y. Після кроку по осі X нове значення $x_{i+1} = x_i + 1$.

Тоді

$$ОФ_{i+1} = y_i – 2 \times (x + 1)^2 = y_i – 2 \times x_i^2 – 2 \times x + 2 = y_i – 4x_i - 2.$$

При виконанні кроку по осі Y нове значення аргументу буде розраховуватись як $y_i = y_i + 1$.

Тоді

$$ОФ_{i+1} = y_i + 1 - 2 \times x_i^2 = ОФ_i + 1.$$

Розрахуємо 4 крокові прирости за умови, що $ОФ_0 = 0$.

1. $ОФ_1 = 0 - 4 - 2 = -6$, $y = 0, x = 1$.
2. $ОФ_2 = -6 + 1 = -5$, $y = 1, x = 1$.
3. $ОФ_3 = -5 + 1 = -4$, $y = 2, x = 1$.
4. $ОФ_4 = -4 + 1 = -3$, $y = 3, x = 1$.

15. Навести вираз для кривої Ерміта, початкова та кінцева точки якої мають координати (0, γ), (1, β), а похідні в початковій та кінцевій точках, відповідно рівні α, γ.

Кубічна крива в формі Ерміта задається рівнянням

$$x(t) = at^3 + bt^2 + ct + d.$$

Знайдемо значення невідомих a, в, c, d.

$$x(0) = d, \quad x(1) = a + b + c + d,$$
$$x'(0) = c, \quad x'(1) = 3a + 2b + c.$$

Таким чином:

$$\gamma = c,$$
$$\beta = a + b + c + d,$$
$$\alpha = c,$$
$$\nu = 3a + 2b + c.$$

Розв'язання системи дає невідомі a, в, c, d.

16. Визначити геометричний вектор Безьє для кривої $X(t) = 3t^3 + 15t^2 + 4t + 1$ за умови, що $P_2 = 7$, $P_3 = 9$.

Геометричний вектор Безьє має вигляд:

$$G = \begin{vmatrix} P1 \\ P4 \\ R1 \\ R4 \end{vmatrix}, \quad \text{де } P1 = x(0),\ P2 = x(1),\ P3 = x'(0),\ P4 = x'(1).$$

Підставляючи в X(t) значення 0 та 1, одержимо:

$$P1 = 1, \qquad P4 = 23.$$

Згідно з алгоритмом Безьє значення R1 та R4 визначаємо за формулами:

$$R1 = 3 \times (P2 - P1),$$
$$R4 = 3 \times (P4 - P3),$$
$$R1 = 18, \quad R2 = 42.$$

Таким чином геометричний вектор Безье для даного прикладу має вигляд:

$$G = \begin{vmatrix} 1 \\ 23 \\ 18 \\ 42 \end{vmatrix}$$

17. Знайти вираз для полінома Лагранжа за умови, що базові точки мають координати: (7, 10), (9, 30), (11, 50).

Інтерполяційний поліном Лагранжа n-го степеня для даної множини точок має вигляд

$$y = \sum_{s=0}^{n} \left(\frac{(x - x_0)...(x - x_{s-1})(x - x_{s+1})...(x - x_n)}{(x_s - x_0)...(x_s - x_{s-1})(x_s - x_{s+1})...(x_s - n)} \right) y_s .$$

В даному випадку маємо:

$$y = \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)} y_1 + \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)} y_2 + \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)} y_3 =$$
$$= \frac{(x-9)(x-11)}{(7-9)(7-11)}10 + \frac{(x-7)(x-11)}{(9-7)(9-11)}30 + \frac{(x-7)(x-9)}{(11-7)(11-9)}50 =$$
$$= \frac{(x-9)(x-11)}{(-2)(-4)}10 + \frac{(x-7)(x-11)}{2(-2)}30 + \frac{(x-7)(x-9)}{4(2)}50 =$$
$$= (x^2 - 11x - 9x + 99)\frac{10}{8} + (x^2 - 11x - 7x + 77)(-\frac{30}{4}) + (x^2 - 7x - 9x + 63)\frac{50}{8} =$$
$$= 1,25x^2 - 25x + 123,75 - 7,5x^2 + 135x - 577,5 + 6,25x^2 - 100x + 393,75 =$$
$$= 10x - 60$$

Висновок. В даному випадку крива вироджена у відрізок прямої, який проведений через три точки.

18. Визначити тип трикутника, який найбільш доцільний для рендерингу Гуро з точки зору обчислювальних затрат та точності розрахунку інтенсивностей кольору пікселів.

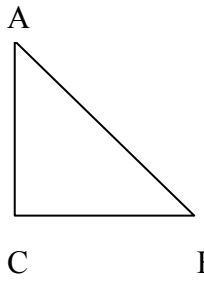


Рисунок 9.11

Попереднім етапом рендерингу Гуро є тріангуляція ділянки, обмеженою полігоном. Згідно з рендерингом зафарбування виконується від ребер АВ та АС.

Якщо відрізок АС паралельний осі абсцис, то відпадає необхідність в рендерингу нижнього ребра.

Приріст інтенсивності кольору вздовж ребра АВ визначається за виразом $\Delta I_{AB} = \frac{I_A - I_B}{\acute{A}\ddot{I}_{\grave{A}\hat{A}}}$, де БП $_{AB}$ – більший з приростів ребра АВ.

$БП_{AB}$ = АВ за умови, що відрізок АВ паралельний осі ординат.

Таким чином, найбільш доцільним для рендерингу Гуро є прямокутний трикутник, катети якого паралельні осям координат.

19. Виконати опис полігональної сітки, наведеної на рис. 9.12, за методом явного задання ребер.

$V = (V_1, V_2, V_3, V_4) = ((X_1, Y_1, Z_1), (X_2, Y_2, Z_2), (X_3, Y_3, Z_3), (X_4, Y_4, Z_4))$

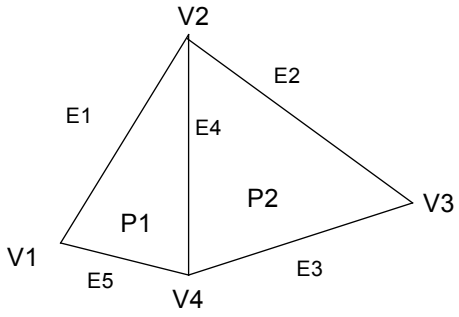


Рисунок 9.12.

$E_1 = (V_1, V_2, P_1, L)$ $\quad E_3 = (V_3, V_4, P_2, L)$

$E_2 = (V_2, V_3, P_2, L)$ $\quad E_4 = (V_1, V_2, P_1, P_2)$

$P_1 = (E_1, E_4, E_5)$ $\quad P_2 = (E_2, E_4, E_3)$

Метод забезпечує швидкий пошук спільних ребер, унеможливлюючи дублювання вершин при заданні багатокутників.

20. Виконати опис полігональної сітки, зображеної на рисунку 9.13, шляхом явного задання багатокутників і з використанням покажчиків.

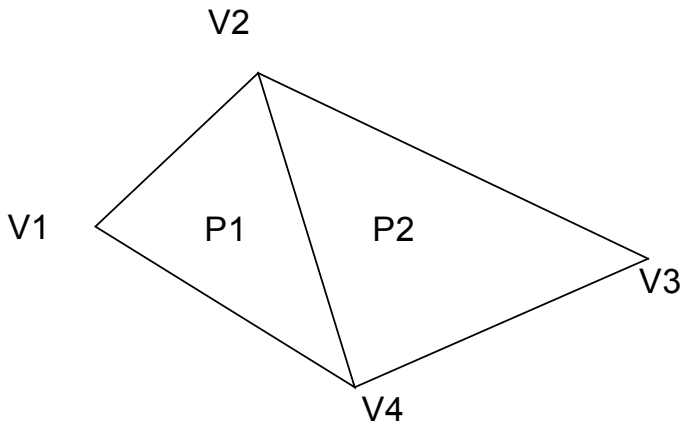


Рисунок 9.13

Опис полігональної сітки при явному завданні багатокутників має вигляд:

$$P_1 = (V_1, V_2, V_4) = ((X_1, Y_1, Z_1), \ldots, (X_2, Y_2, Z_2), \ldots, (X_4, Y_4, Z_4))$$
$$P_2 = (V_2, V_3, V_4) = ((X_2, Y_2, Z_2), (X_3, Y_3, Z_3), \ldots, (X_4, Y_4, Z_4))$$

Явне задання багатокутників характеризується надлишковістю, оскільки для вершин V_2 та V_4 опис проведений двічі.

Опис логічної сітки при використанні покажчиків має вигляд:

$$P_1 = (V_1, V_2, V_3, V_4) = ((X_1, Y_1, Z_1), \ldots, (X_2, Y_2, Z_2), (X_3, Y_3, Z_3), (X_4, Y_4, Z_4))$$
$$P = (1, 2, 4) \quad P = (4, 2, 3)$$

При заданні P цифри в дужках є покажчиками вершин в описі V.

21. Виконати відсікання відрізка прямої, показаного на рис. 9.14, за алгоритмом ділення відрізка навпіл.

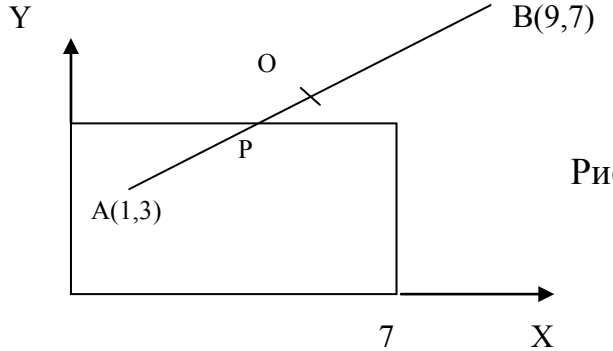


Рисунок 9.14.

Знаходимо координати точки 0

$$X_0 = X_A + \frac{X_B - X_A}{2} = 1 + \frac{9-1}{2} = 5.$$

$$Y_0 = \frac{Y_B - Y_A}{2} = 3 + \frac{7-3}{2} = 5.$$

Відрізок ОВ, згідно з алгоритмом Коена – Сазерленда, відкидається.
Знаходимо координати точки Р

$$Xp = X_A + (X_0 - X_A) / 2 = 1 + (5 - 1) / 2 = 3,$$
$$Yp = Y_A + (Y_0 - Y_A) / 2 = 3 + (5 - 3) / 2 = 4.$$

Відрізок ОР, згідно з алгоритмом Коена – Сазерленда, відкидається.
Відрізок АР належить вікну.

22. Виконати відсікання відрізків прямих, зображених на рисунку 9.15, згідно з алгоритмом Коена – Сазерленда.

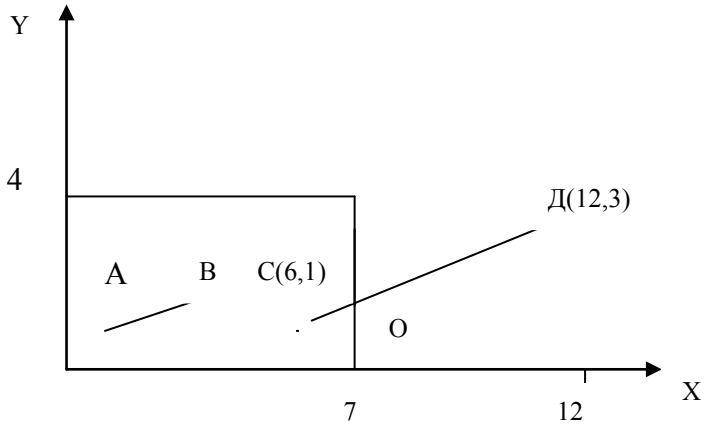


Рисунок 9.15

Визначимо чотирирозрядні коди для точок А та В згідно з ознаками:

	Вище	Нижче	Лівіше	Правіше
A	0	0	0	0
B	0	0	0	0

Оскільки чотирирозрядні коди точок А та В нульові, то відрізок знаходиться всередині вікна.

Для точок С та Д чотирирозрядні коди дорівнюють:

точка С – 0000,
точка Д – 0001.

Оскільки операція порозрядного логічного добутку дає нульовий результат, то потрібно виконати відсікання.

Відрізок прямої СД задаємо рівнянням вигляду

$$Y = 1 + 2/6 \times (x - 6) = 1 + 1/6\ (x - 6).$$

Максимальна абсциса межі вікна дорівнює 7. Знаходимо Y для x = 7.

$$Y = 1 + 1/6 = 7/6.$$

Точка О має координати (7, 7/6). Відрізок ОД відкидається.

23. Визначити мінімальний об'єм відеопам'яті режиму True Color, якщо монітор підтримує стандарт Artist 1+.

Режим True Color забезпечує 16777216 кольорів, що вимагає використання $\log_2 16777216 = 24$ розрядів відеопам'яті на один піксел.

Розмір адресного простору для режиму Artist 1+ дорівнює 1024×768, а загальна кількість точок на екрані в цьому випадку дорівнює $1024 \times 768 = 786432$.

Потрібний обсяг відеопам'яті для режиму True Color буде дорівнювати $786432 \times 24 = 18874368$ бітів.

24. Визначити мінімальний обсяг відеопам'яті за умови, що підтримується режим SVGA і формується 256 кольорів.

Режиму SVGA відповідає екран з адресним простором 1024×768 точок.

Для кодування 256 кольорів потрібно $\sqrt{256} = 8$ шарів пам'яті. Загальний обсяг пам'яті дорівнюватиме 1024×768×8 = 6 291 456 бітів.

25. Визначити частоту рядкового розгорнення при формуванні телевізійного растра з числом рядків у кадрі Z = 625 при: а) прогресивному б) черезрядковому розгорненні.

Відповідно до формули $F_z = Z \times f_k$ визначаємо частоту для прогресивного розгорнення (f_k = 50 Гц) $f_z = 625 \times 50 = 31250$ Гц; для черезрядкового розгорнення (f_k = 25 Гц) $f_z = 625 \times 25 = 15\ 625$ Гц.

26. Визначити кількість пікселів зображення, що його можна сформувати при асинхронному растровому розгорненні за 50 кадрів за умови, що зворотний хід рядка та кадру відповідно дорівнюють t_{ap}, t_k, час розрахунку піксела мікропроцесором – t, а кількість рядків на екрані – n.

При асинхронному растровому розгорненні відеопам'ять доступна під час зворотних ходів рядка та кадра. За час зворотного ходу рядка можна сформувати $\left\lfloor \frac{t_p}{t} \right\rfloor$ пікселів. Враховуючи, що кількість рядків дорівнює n, то за прямий хід кадра буде сформовано $\left\lfloor \frac{t_p}{t} \right\rfloor$ n пікселів. За час зворотного ходу кадра мікропроцесор сформує $\left\lfloor \frac{t_k}{t} \right\rfloor$ точок. За 50 кадрів загальну кількість точок визначаємо за формулою $50 \times \left(n \left\lfloor \frac{t_p}{t} \right\rfloor + \left\lfloor \frac{t_k}{t} \right\rfloor \right)$.

27. ***Визначити кількість оптоелектроних пар передекранної сенсорної панелі для забезпечення режиму точного позиціювання за умови, що використовується екран стандарту Artist 1+.***

Передекранна сенсорна панель містить пари оптоелектронних елементів, які розміщені по периметру екрана. Кожна з таких пар містить джерело світла та його приймач.

При дотикові оператором до передекранної панелі виконується введення в ЕОМ координат X та У положення об'єкта на екрані.

Режиму Artist 1+ відповідає адресний простір 1024 × 768 точок.

Позначимо через n кількість оптоелектроних пар, яку потрібно забезпечити для однієї зі сторін екрану, яка містить N точок.

Тоді N/n визначає розмір макрозони для режиму позиціювання.

Для забезпечення ідентифікації кожної точки макрозони відношення N/n має дорівнювати n, тобто N/n = n. $N = n^2$.

Звідси $n = \lceil \sqrt{N} \rceil$ при N = 1024, n = 32, якщо N = 68, n = 29.

28. Визначити типи чотирикутників, для яких при заповненні за критерієм зв'язності потрібно дві точки-затравки за умови, що виконується горизонтальна растеризація, а одна з точок-затравок розміщена у верхньому кутку чотирикутника.

Більше 2-х точок-затравок потрібно за умови, що багатокутник не є випуклим.

Можливі варіанти невипуклих чотирикутників наведені на рис. 9.16.

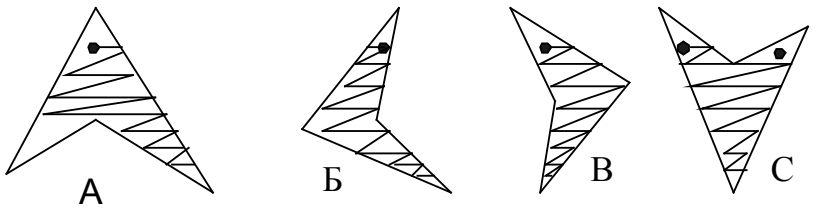


Рисунок 9.16

Як видно з рисунка, точки-затравки при горизонтальній растеризації потрібні для випадків А та С.

29. Знайти вираз для полінома Лагранжа за умови, що базові точки мають координати (7, 10), (9, 30), (11, 50).

Інтерполяційний поліном Лагранжа n-го степеня для даної множини точок має вигляд

$$y = \sum_{s=0}^{n} \left(\frac{(x-x_0)...(x-x_{s-1})(x-x_{s+1})...(x-x_n)}{(x_s-x_0)...(x_s-x_{s-1})(x_s-x_{s+1})...(x_s-n)}\right) y_s .$$

У даному випадку маємо

$$y = \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} y_1 + \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} y_2 + \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)} y_3 =$$

$$= \frac{(x-9)(x-11)}{(7-9)(7-11)}10 + \frac{(x-7)(x-11)}{(9-7)(9-11)}30 + \frac{(x-7)(x-9)}{(11-7)(11-9)}50 =$$

$$= \frac{(x-9)(x-11)}{(-2)(-4)}10 + \frac{(x-7)(x-11)}{2(-2)}30 + \frac{(x-7)(x-9)}{4(2)}50 =$$

$$= (x^2 - 11x - 9x + 99)\frac{10}{8} + (x^2 - 11x - 7x + 77)(-\frac{30}{4}) + (x^2 - 7x - 9x + 63)\frac{50}{8} =$$

$$= 1,25x^2 - 25x + 123,75 - 7,5x^2 + 135x - 577,5 + 6,25x^2 - 100x + 393,75 =$$

$$= 10x - 60$$

30. Записати рівняння характеристик сім'ї поверхонь $x+c^2y+ z - 2c = 0$.

Диференціюємо за параметром c $F_c = 2cy - 2 = 0$. При фіксованому $c = c_0$ два рівняння $x + c_0^2 y + z - 2c_0 = 0$ і $2c_0y - 2 = 0$, які є рівняннями площин, визначать характеристику – пряму їх перетину. Її напрямний вектор має координати $(-c_0, 0, c_0)$, або (-1, 0, 1).

31. Визначити відстань між точками екрана, при якій афект аліайзингу невидимий для спостерігача.

При формуванні зображення на стадії растеризації виникають спотворення, зумовлені дискретним характером формування зображень і недостатньою роздільною здатністю пристроїв відображення. На краях об'єктів з'являються яскраво виражені сходинки або зубці. Даний артефакт отримав назву ступінчастого ефекту чи ефекту аліайзингу. Ефект аліайзингу погіршує якість сформованого зображення, що передбачає розробку спеціальних методів і засобів його усунення. Дослідження показали, що при використанні $17^{//}$ монітора та розміщенні спостерігача на відстані 65 см від екрана для повного усунення ефекту аліайзингу потрібен монітор з роздільною здатністю як мінімум 4000 × 4000 пікселів.

Діагональ монітора 17 дюймов. 1 дюйм = 2,54 сантиметра. Отже, діагональ монітора 43,18 сантиметра.

Знайдемо розмір сторін монітора за формулою $a^2 + a^2 = b^2$, де a – сторона монітора, а b – його діагональ.

$$a = \sqrt{\frac{b^2}{2}} = \sqrt{\frac{(43,18)^2}{2}} = 30,52\,см$$

Відстань. між точками екрана дорівнює $\frac{30,52\,см}{4000} = 0,00736\,см$.

32. Визначити необхідний обсяг відеопам'яті для різних графічних режимів екрана монітора, якщо відома глибина кольору на одну точку.

Режим екрана	Глибина кольору (бітів на точку)				
	4	8	16	24	32
640 на 480					
800 на 600					
1024 на 768					
1280 на 1024					

Для зразка розглянемо екран 640 × 480. Усього на екрані 640 × 480 = 307200 точок.

Необхідний обсяг відеопам'яті

V = 4 біти × 307200 = 1228800 бітів = 153600 байтів = 150 Кбайтів.

Аналогічно розраховується необхідний обсяг відеопам’яті інших графічних режимів.

Режим екрана	Глибина кольору (бітів на крапку)				
	4	8	16	24	32
640 на 480	150 Кб	300 Кб	600 Кб	900 Кб	1,2 Мб
800 на 600	234 Кб	469 Кб	938 Кб	1,4 Мб	1,8 Мб
1024 на 768	384 Кб	768 Кб	1,5 Мб	2,25 Мб	3 Мб
1280 на 1024	640 Кб	1,25 Мб	2,5 Мб	3,75 Мб	5 Мб

Рекомендована література

Базова

1. Naty Hoffman, Tomas Akenine-Moller, Real-Time Rendering. Publisher : A K Peters/CRC Press; 4th edition (August 6, 2018), 1178 pages.
2. P. Godse, Dr. D. A. Godse. Computer Graphics and Multimedia Publisher : 9789333223362 (November 3, 2020), 686 pages.
3. R. Stuart Ferguson. Practical Algorithms for 3D Computer Graphics, Publisher : A K Peters/CRC Press; 2nd edition (October 6, 2019), 520 pages.
4. Пічугін М. Ф., Канкін І. О., Воротніков В. В. Комп'ютерна графіка : навч. пос. К. : Центр навчальної літератури. 2019. 346 с.
5. Романюк О. Н. Комп'ютерна графіка. Електронний навчальний посібник. Режим доступу : http://posibnyky.vntu.edu.ua/k_g/index.html.
6. Романюк О. Н., Майданюк В. П. Практикум для самостійної роботи студентів з навчальної дисципліни «Комп'ютерна графіка». Вінниця : ВНТУ, 2022. 86 с.
7. Інженерна та комп'ютерна графіка: практикум для навчання в умовах інформаційно-освітнього середовища : навч. посіб. / [Д. В. Бабенко та ін.] ; за ред. професора Д. В. Бабенка. Миколаїв : МНАУ, 2020. 256 с.
8. Веселовська Г. В., Ходаков В. Є., Веселовський В. М. Комп'ютерна графіка : навч. посіб. для студентів ВНЗ ; за ред. В. Є. Ходакова. Херсон : Олді-Плюс, 2018. 581 с.
9. Журавчак Л. М., Левченко О. М. Програмування комп'ютерної графіки та мультимедійні засоби : навч. посіб. Львів : Вид-во Львівської політехніки, 2019. 276 с.
10. Пєтух А. М., Обідник Д. Т., Романюк О. Н. Інтерполяція в задачах контурного формоутворення : моногр. Вінниця : ВНТУ, 2007. 103 с.
11. Романюк О. Н., Чорний А. В. Високопродуктивні методи та засоби зафарбовування тривимірних графічних об'єктів : монографія. Вінниця : УНІВЕСУМ-Вінниця, 2006. 190 с.
12. Романюк О. Н., Катєльніков Д. І., Косовець О. П. Веб-дизайн і комп'ютерна графіка : навч. посіб. Вінниця : ВНТУ, 2020. 103 с.
13. Журавчак Л. М., Левченко О. М. Програмування комп'ютерної графіки та мультимедійні засоби : навч. посіб. Львів : Вид-во Львівської політехніки, 2019. 276 с.
14. Комп'ютерна графіка: конспект лекцій для студентів усіх форм навчання спеціальностей 122 «Комп'ютерні науки» та 123 «Комп'ютерна інженерія» з курсу «Комп'ютерна графіка». / Укладач О. П. Скиба, Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2019. 88 с.

Додаткова

1. Власій О. О., Дудка О. М. Комп'ютерна графіка. Обробка растрових зображень : навч.-метод. посіб. Івано-Франківськ : ДВНЗ «Прикарпатський національний університет імені Василя Стефаника», 2015. 72 с.
2. Співак С. М. Теоретичні основи комп'ютерної графіки та дизайну : навч. посіб. К, : Київський університет імені Бориса Грінченка, 2013.
3. 3DS MAX для початківців. / Романюк О. Н. та ін. Вінниця : Едельвейс, 2015. 100 с..
4. Романюк О. Н., Курінний М. С. Методи та засоби антиаліайзингу контурів об'єктів у системах комп'ютерної графіки : монографія. Вінниця : УНІВЕСУМ-Вінниця, 2006. 163 с.
5. Дудка О. М. Комп'ютерна графіка : навч. пос. Вид. 7-ме. Івано-Франківськ : Прикарпатський національний університет імені Василя Стефаника: ЦІТ, 2010. 55 с.

Ресурси Інтернет

1. Машинна графіка. Режим доступу : https://uk.wikipedia.org/wiki/Машинна графіка
2. Комп'ютерна графіка : навчальний посібник: в 2-х кн. Кн. 1. / Укладачі: Тотосько О. В., Микитишин А. Г., Стухляк П. Д. Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2017. 304 с. URL: http://elartu.tntu.edu.ua/ bitstream/lib/22337/1/Komp_graf_knyga_1.pdf.
3. Романюк О. Н. Комп'ютерна графіка. Електронний навчальний посібник. Режим доступу : http://posibnyky.vntu.edu.ua/k_g/index.html
4. Заїка В. Ф., Твердохліб М. Г., Тарбаєв С. І., Чумак Н. С. Основи інженерної та комп'ютерної графіки. 2017. Режим доступу до ресурсу : http://www.dut.edu.ua/uploads/l_1622_31814633.pdf.
5. Посібник користувача Photoshop. [Електронний ресурс] Режим доступу до ресурсу : https://helpx.adobe.com/ua/photoshop/user-guide.html
6. Посібник користувача Illustrator. [Електронний ресурс] Режим доступу до ресурсу : https://helpx.adobe.com/ua/illustrator/user-guide.html
7. Комп'ютерна графіка. Режим доступу : https://wikiless.org/wiki/ Комп'ютерна графіка
8. D. Eck. Introduction to computer graphics. Режим доступу до ресурсу: http://math.hws.edu/graphicsbook/ 9. Joey de Vries. Welcome to OpenGL. – Режим доступу до ресурсу : https://learnopengl.com/

Електронне навчальне видання

Романюк Олександр Никифорович
Романюк Оксана Володимирович
Чехместрук Роман Юрійович

Комп’ютерна графіка

Рукопис оформлено *О. Романюком*

Редактор *В. Дружиніна*

Оригінал-макет виготовлено у *РВВ ВНТУ*

Підписано до видання 14.06.2023 р.
Гарнітура Times New Roman.
Зам. № Р2023-073.

Видавець та виготовлювач
Вінницький національний технічний університет,
Редакційно-видавничий відділ.
ВНТУ, ГНК, к. 114.
Хмельницьке шосе, 95, м. Вінниця, 21021.
press.vntu.edu.ua;
Email: irvc.vntu@gmail.com.
Свідоцтво суб’єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.